Approximation Algorithms for Disjoint Paths Problems

by

Jon Michael Kleinberg

S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology
(1994)
A.B., Computer Science and Mathematics
Cornell University
(1993)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996

Signature of Author____
Department of Electrical Engineering and Computer Science
May 3, 1996

Certified by____
Michel X. Goemans
Assistant Professor of Applied Mathematics
Thesis Supervisor

Accepted by____
F.R. Morgenthaler
Chairman, Department Committee on Graduate Students

Approximation Algorithms for Disjoint Paths Problems

by

Jon Michael Kleinberg

Submitted to the Department of Electrical Engineering and Computer Science

on May 3, 1996, in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy in Computer Science

Abstract

The construction of disjoint paths in a network is a basic issue in combinatorial optimization: given a network, and specified pairs of nodes in it, we are interested in finding disjoint paths between as many of these pairs as possible. This leads to a variety of classical NP-complete problems for which very little is known from the point of view of approximation algorithms. It has recently been brought into focus in work on problems such as VLSI layout and routing in high-speed networks; in these settings, the current lack of understanding of the disjoint paths problem is often an obstacle to the design of practical heuristics.

We develop a number of general techniques for designing approximation algorithms for disjoint paths problems. Our approach leads to the first constant-factor approximation algorithm for the maximum disjoint paths problem on the two-dimensional mesh, as well as on a class of planar networks generalizing the mesh. We obtain a number of related algorithms by this approach, including a routing algorithm for the mesh that is optimal in the *on-line* model considered in much of the recent work on this problem; here connection requests arrive over time and must be processed immediately. Underlying a number of our results is a new approximation algorithm for a basic, NP-hard variant of the maximum flow problem.

Thesis Supervisor: Michel X. Goemans

Title: Assistant Professor of Applied Mathematics

Keywords: combinatorial optimization, approximation algorithms, network flow

Acknowledgements

It has been my privilege to have worked in a number of exciting and intellectually rich environments, and with a wonderful array of mentors and colleagues. I thank my thesis advisor Michel Goemans for his support and collegiality over the past three years. I have benefited enormously from working with him, and from his perspective on algorithms and combinatorics. I began my work in the area that is the topic of this thesis with Éva Tardos, and the research contained in three of the chapters of this thesis was done in collaboration with her. I am grateful to have had the opportunity to work so closely with her — a process that has significantly shaped my approach to research — and for the considered guidance she has given me in many different regards.

Early in my undergraduate career at Cornell, I had the good fortune to begin working with Dan Huttenlocher; and this collaboration has also had a great impact on my development as a computer scientist. I have worked with Dan both as a fellow researcher and as his teaching assistant, and I thank him for imparting to me his broad and balanced view of research in computer science. The environment at Cornell was both exciting and immensely supportive — I have benefited from interactions with many of the faculty and visitors there, including Paul Chew, Bruce Donald, Rod Downey, Klara Kedem, Keith Marzullo, Paul Pedersen, Ronitt Rubinfeld, David Shmoys, Bernd Sturmfels, Sam Toueg, and Nick Trefethen. The community of friends and fellow students in which I was immersed there was a wonderful one as well; I will try to avoid listing too many names here, but want to mention Lillian Lee, Tony Yan, Anne Kist, Sendhil Mullainathan, Jen Smith, Barry Isralewitz, Rachel Byard, Charlotte Lee, Aravind Srinivasan, and Thomas Yan.

The environment in the Lab for Computer Science at MIT has been an exciting one to be a part of. I thank Tom Leighton for his advice and for the opportunity to work with him on a variety of problems. I have also been fortunate to have been able to work with Bonnie Berger, David Karger, and Nancy Lynch; I additionally thank all of them for their continuing advice as my job search progressed this year, and David for his thorough reading of this thesis. I am also grateful to have had the chance to work with several of the visitors to the theory group at MIT — in particular, Allan Borodin and Hagit Attiya during my first year here, and Ronitt Rubinfeld during my last. I thank Marcos Kiwi and Dan Spielman for a wonderful year as apartmentmates; and I also thank Matt Andrews, András Benczúr, Michael Bender, Stan Chen, Ric Crabbe, Rosario Gennaro, Joshua Goodman, Shai Halevi, Rebecca Hwa, Mike Leventon, Liana Lorigo, Carol Sandstrom and Chris Small (and Emily and Sophie), Ravi Sundaram, and Lisa Zhang. Finally, I thank Bruce Dale, Be Hubbard, David Jones, and Joanne Talbot for all their help.

I spent much of the summer of 1995 working in the Theory of Computation Group at the IBM Watson Research Center — a very lively and welcoming research environment. Portions of the work in this thesis, as well as current aspects of my research, emerged from my time there; and I look forward to continued collaboration with the researchers and other summer visitors that I met at IBM. In particular, I thank Pankaj Agarwal, Alok Aggarwal, Allan Borodin, Don Coppersmith, Sanjeev Khanna, Bill Pulleyblank, Prabhakar Raghavan, Baruch Schieber, Madhu Sudan, and David Williamson. I additionally would like to thank Alok, Allan, and Prabhakar for their advice over the past year.

Finally: I thank my parents, my brother Bobby, and Lillian, for all their support. It is because of them that my years spent on this work have been such a delight.

For five of my six semesters at MIT, I have been supported by an ONR Graduate Fellowship. Portions of this thesis represent joint work with several of the people mentioned above: Chapters 6 and 7 are based on joint work with Éva Tardos, Chapter 8 on joint work with Éva Tardos and Satish Rao, Chapter 9 on joint work with Alok Aggarwal and David Williamson, and portions of Chapter 5 on joint work with Ronitt Rubinfeld.

Contents

Chapter 1

Introduction

> *People, he said, ... they use the telephone. Phil calls Philadelphia. Pat calls Patagonia. ...*
>
> — *Vladimir Nabokov,* Lolita.

A basic problem that arises in large-scale communication networks is that of assigning paths to *connection requests*. Each connection request is a pair of physically separated nodes that wish to communicate over a path through the network; given a list of such requests, one wants to assign paths to as many as possible in such a way that no two paths "interfere" with each other. Thus, for a given list of requests, we can ask a number of natural questions. How many requests are simultaneously *realizable* using paths that are mutually edge-disjoint? How many *rounds of communication* are required to satisfy all requests, when all paths assigned in a single round must be edge-disjoint? These turn out to be classical problems of combinatorial optimization and operations research; they are also among the most well–known NP-complete graph problems.

The intractability of these disjoint paths problems does not appear to be simply a theoretical phenomenon. A number of recent papers (e.g. [12, 46]) have observed that much of the difficulty in admission control and routing of virtual circuits in high–performance communication networks comes from the lack of good heuristics for constructing disjoint paths. These issues also arise within the context of VLSI, where the current lack of understanding of these problems is often an obstacle in the design of effective layout and wire routing algorithms for printed circuits.

In essentially all of the optimization versions of these problems, finding the best solution is NP-complete. In this work, therefore, we are interested in developing efficient algorithms that produce *near–optimal* solutions. Such *approximation algorithms* have recently been the focus of an intense amount of work in the theoretical computer science community; in the setting of disjoint paths problems, as in many similar settings, the development of approximation algorithms is of value both in suggesting robust heuristic approaches to computationally intractable problems, and in providing further understanding of the elusive structure of their optima.

Let us make all this more precise, as follows. Given an undirected graph $G = (V, E)$, each connection request is specified by a pair of *terminals* s_i and t_i, where $s_i, t_i \in V$. Let $\mathcal{T}$ be the set of all terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$. We say that $\mathcal{T}$ is *realizable* in G if there exist mutually edge-disjoint paths $P_1, \ldots, P_k$ such that P_i has endpoints s_i and t_i. Given G, k, and $\mathcal{T}$, determining whether $\mathcal{T}$ is realizable in G is one of Karp's original NP-complete problems [56] — we will refer to this as the basic *disjoint paths problem.*

Thus, the first of the optimization problems mentioned in the opening paragraph is simply to find a maximum–size realizable subset of a set $\mathcal{T}$ of terminal pairs in a graph G. We will refer to this as the *maximum disjoint paths problem* (MDP), and denote the maximum size of a realizable subset of $\mathcal{T}$ in G by $\alpha(\mathcal{T})$.

Theoretical work on the disjoint paths problem has its roots in the early development of graph connectivity and network flow. The special case in which all (s_i, t_i) pairs are the same — so that one wants to find the maximum number of disjoint paths between two fixed nodes in a graph — is the essence of Menger's classical theorem [82], generalized into the max-flow min-cut theorem of Ford and Fulkerson [34, 35]. This work also led to efficient algorithms for this special case; see Chapter 2 for technical background on this and others of the results mentioned in this introduction.

Following Karp's NP-completeness result, it became clear that similar clean structural results and efficient algorithms were not likely to be obtainable for the disjoint paths problem in its full generality. Thus, most of the graph–theoretic work in this area has turned to tractable special cases of the problem; this is given extensive coverage in a survey by Frank [39].

Much less attention has been devoted to the problem of designing approximation algorithms for intractable cases of the MDP, the subject of our work here. To be concrete, by a *c–approximation algorithm* for the MDP we mean an algorithm that, on input G and $\mathcal{T}$, produces a realizable set $\mathcal{T}' \subseteq \mathcal{T}$ of size at least $\alpha(\mathcal{T})/c$. All the approximation algorithms we will discuss run in polynomial time. We will survey previous work on approximation in Chapter 2. It is worth noting at the outset that many of the algorithms we discuss here are randomized; their approximation guarantees hold with constant probability, and hence can be made to hold with probability $1 - 2^{-k}$ by running them $O(k)$ times independently. Thus this notion of a probabilistic performance guarantee does not rely on any assumptions about the distribution of the input; it holds for all input instances, and the probability is taken only over the random choices made by the algorithm. In some of our results randomization is used mainly for the sake of simplicity and can be eliminated; in some we do not know how to avoid the use of randomization; and in some it is possible to prove that no deterministic algorithm can perform as well as the randomized algorithm we present.

Let us discuss a natural generalization of the MDP, obtained by incorporating an additional consideration from the network routing applications discussed above. Again, consider a graph $G = (V, E)$ with a set $\mathcal{T}$ of terminal pairs. Now, however, each pair (s_i, t_i) has associated with it a given *demand* ρ_i; and each edge e is given a *capacity* c_e, which indicates the total amount of traffic it can carry. We say that $\mathcal{T}$ is *realizable* in G if there exist paths $P_1, \ldots, P_k$ such that P_i has endpoints s_i and t_i, and the following *capacity constraint* is met for every edge:

$$\sum_{i:e \in P_i} \rho_i \leq c_e. \tag{1.1}$$

That is, the total demand contained in paths using edge e does not exceed the capacity of e. Throughout this work, we will make the following *balance assumption*:

$$\forall i, e \ : \ \rho_i \leq c_e \tag{1.2}$$

That is, every demand on its own is capable of "fitting" across every edge.

For $\mathcal{T}' \subseteq \mathcal{T}$, define $\rho(\mathcal{T}')$ to be the total demand of terminal pairs in $\mathcal{T}'$:

$$\rho(\mathcal{T}') = \sum_{i:(s_i,t_i) \in \mathcal{T}'} \rho_i. \tag{1.3}$$

The *unsplittable flow problem* (UFP), with input G and $\mathcal{T}$, asks for a realizable subset $\mathcal{T}'$ of $\mathcal{T}$ that maximizes $\rho(\mathcal{T}')$.[1] Note that in the case when all ρ_i and all c_e are equal to 1, this coincides with the maximum disjoint paths problem; and hence designing approximation algorithms for the UFP can only be more difficult than for the MDP. By analogy with the MDP, let us use $\alpha(\mathcal{T})$ to denote the maximum value of $\rho(\mathcal{T}')$, over all realizable $\mathcal{T}' \subseteq \mathcal{T}$.

There is, however, a special case of the UFP in which strong approximation results have been obtained by Raghavan and Thompson [94, 93]. This is the case in which all capacities c_e exceed all demands ρ_i by a factor of at least $\Omega(\log |E|)$ — we will refer to this as the *high-capacity* UFP. Here, the algorithms of [94, 93] provide a constant–factor approximation for any graph G. While these results do not relate directly to the disjoint paths problem itself, we will make use of them in designing some of our algorithms. Of course the strength of this result is such that the UFP can be considered largely settled in this "high–capacity" case. However, in many network routing applications, each communication path can consume a large fraction of the available bandwidth on a link [12, 90]; for such situations, one must be able to deal with edges whose capacity is small relative to the individual demands.

1.1 Related Optimization Problems

The two problems discussed above will serve as our main focus; however, there are a number of closely related optimization problems that our techniques can be adapated to handle. We introduce these here.

On–Line Routing. Above, we observed that the routing of connections in communication networks is a motivation for the disjoint paths and unsplittable flow problems. In many of these applications, one is faced with the additional constraint of being presented with terminal pairs gradually over time; as each pair is presented, it must immediately be routed on a free path through G, or rejected, without knowledge of future requests. The goal remains to route a large number of terminal pairs relative to the optimum. This (not surprisingly) results in

[1]This problem is often referred to as the *integer multicommodity flow problem* in the operations research literature. We have adopted the unorthodox name used here because of our understanding that the term *integer multicommodity flow* does not have a single standardized meaning within this area; for example, it frequently refers only to the special case in which all ρ_i are equal to 1. See Section 2.4.

a number of additional complications that the designer of an approximation algorithm must contend with; for example, it rules out approaches that incrementally develop a solution by *re-routing* old paths as new ones are added.

Algorithms that must operate in this "one–pass" fashion are typically termed *on-line algorithms.* The design of on–line algorithms is an issue that has been actively studied over the last ten years; the work of Sleator and Tarjan [114] was a major influence in popularizing this area. Routing in networks is clearly a natural setting in which to analyze on–line algorithms, and this has been undertaken at a theoretical level by a number of researchers; some of the earliest papers here were those of Garay, Gopal, et al. [41, 42] and Aspnes et al. [8].

Interestingly, good approximations exist for the high–capacity UFP even in the on-line model. An algorithm of Awerbuch, Azar, and Plotkin [10] provides an on–line $O(\log |E|)$-approximation when no demand requires more than an $O(\log^{-1} |E|)$ fraction of the minimum edge capacity. In the on–line setting, as in the classical setting, much less is known about approximation when the demands ρ_i are allowed to be comparable in size to the capacities c_e. Moreover, some lower bounds have been established here, showing that no on-line algorithm can provide a good approximation in certain families of graphs [10, 17, 15]. Again, see Chapter 2 for a discussion of prior work here.

When it is necessary to distinguish on–line algorithms from the traditional algorithms that will occupy our attention for most of the time, we will refer to the latter as *off-line algorithms.*

Routing in Rounds. Let us return to the second of the questions raised in the opening paragraph — what is minimum number of *rounds of communication* required to route a set $\mathcal{T}$ of terminal pairs, when all paths assigned in a single round must be edge-disjoint? More succinctly, given a graph G and set $\mathcal{T}$ of terminal pairs, we are asking for the smallest number of realizable sets into which $\mathcal{T}$ can be partitioned. Let us denote this minimum by $\chi(\mathcal{T})$. Of course, $\chi(\mathcal{T}) = 1$ if and only if $\mathcal{T}$ is realizable in G, so determining $\chi(\mathcal{T})$ is NP-complete. We also note that the same question applies to the more general setting of unsplittable flow with arbitrary demands, where again we seek a minimum partition into realizable sets.

In addition to the above scheduling interpretation, one can view this problem in a number of other ways. As a salient example, the objective function $\chi(\mathcal{T})$ arises in the design of algorithm for routing in all–optical networks. The model here, considered in [1, 95, 9] and based on current proposals for network design using optical communication technology, is the following. Each connection request (s_i, t_i) must be assigned an s_i-t_i path and a *wavelength*, so that if two paths share an edge then they must use different wavelengths. The goal is to minimize the number of wavelengths used to route all the connections. There is clearly a one–to–one correspondence between wavelengths here, and the "rounds" of the previous paragraph.

Congestion. Finally, there is a third natural objective function (in addition to α and χ) that one can consider when given a graph G and terminal pairs $\mathcal{T}$. As above, we suppose that each terminal pair (s_i, t_i) has an associated demand ρ_i. Let us say that a *routing* of $\mathcal{T}$ in G is simply a choice of an s_i–t_i path P_i, for each $(s_i, t_i) \in \mathcal{T}$. The *congestion* of a routing is then defined to be the maximum amount of demand carried by the routing over any edge in G; i.e., it is given by the expression

$$\max_e \sum_{i:e\in P_i} \rho_i. \tag{1.4}$$

We use $\nu(\mathcal{T})$ to denote the minimum congestion achievable by a routing of $\mathcal{T}$ in G. Of course, when all demands are equal to 1, $\nu(\mathcal{T}) = 1$ if and only if $\mathcal{T}$ is realizable in G, so determining $\nu(\mathcal{T})$ is NP-complete.

The notion of congestion does not take into account the capacities on the edges of G (implicitly, all are equal to 1). When dealing with arbitrary capacities, we will also be interested in the *relative congestion* of a routing — the maximum fraction by which the demand carried by the routing over any edge e exceeds its capacity c_e. That is, the relative congestion of a routing specified by paths $\{P_i\}$ is equal to

$$\max_e \frac{\sum_{i:e\in P_i} \rho_i}{c_e}. \tag{1.5}$$

We use $\nu_0(\mathcal{T})$ to denote the minimum relative congestion achievable by a routing of $\mathcal{T}$ in G.

Note that $\nu(\mathcal{T}) = \nu_0(\mathcal{T})$ in any graph in which all capacities are equal to 1. Note also that $\nu_0(\mathcal{T}) \leq \chi(\mathcal{T})$, since one can obtain a routing of relative congestion $\chi(\mathcal{T})$ by simply "superimposing" all the paths used in a routing of $\mathcal{T}$ in $\chi(\mathcal{T})$ rounds. But there are graphs G and sets $\mathcal{T}$ of terminal pairs for which $\chi(\mathcal{T})$ can be much larger than $\nu_0(\mathcal{T})$; we will see examples of this in Chapter 2.

Karp, Leighton, et al. [57] and Raghavan and Thompson [94] use congestion as a measure of how "densely" wires must be packed on a VLSI chip. And though the UFP is perhaps a more accurate reflection of the types of packing constraints that predominate in virtual circuit routing in high-speed networks, congestion also makes sense as a measure in this domain. In particular, the *statistical multiplexing* capabilities of Asynchronous Transfer Mode (ATM) allow one, in some cases, to trade off the congestion on a link against the expected rate of cell loss. We will not discuss this further here; we refer the reader to a reference such as [90] for a general introduction to ATM.

An Example. It is worth discussing an example of a graph G and a set $\mathcal{T}$ of terminal pairs, in order to illustrate the relationship between the three objective functions α, χ, and ν. Perhaps surprisingly, the case in which G is a graph consisting of one unit-capacity edge (s,t) is already interesting. Let us assume the set $\mathcal{T}$ of terminal pairs consists of k copies of the pair (s,t), with demands $\rho_1, \ldots, \rho_k$. Then

- $\alpha(\mathcal{T})$ corresponds to the NP-complete *knapsack problem*: given real–valued weights $\rho_1, \ldots, \rho_k \in (0,1)$, find a subset of the $\{\rho_i\}$ of maximum total weight, subject to the constraint that their total weight not exceed 1.

- $\chi(\mathcal{T})$ corresponds to the NP-complete *bin–packing problem*: given real–valued weights $\rho_1, \ldots, \rho_k \in (0,1)$, place them into as few "bins" as possible, so that the total amount of weight placed in any bin does not exceed 1.

- $\nu(\mathcal{T})$ is trivial in this case; it is equal to $\sum_i \rho_i$.

1.2 Overview of the Present Work

We now provide a brief overview of the material in Chapters 3–9, which constitutes the main technical portion of this work. The results described here are part of the overall context set forth in Chapter 2, and they extend some of the earlier work discussed there.

The focus of this work is on approximation algorithms for the unsplittable flow problem and its variants. The two main results that we build up to are a general constant–factor approximation algorithm for the *single–source* unsplittable flow problem (Chapters 3 and 4), when all terminal pairs share a common vertex; and a constant–factor approximation algorithm for the unsplittable flow problem on a class of locally planar graphs that generalizes the two–dimensional mesh (Chapter 7). The latter algorithm requires, as one component, one of the single–source algorithms developed in Chapter 3. Variants of the latter algorithm provide, in the same general class of graphs, a constant–factor approximation algorithm for the problem of minimizing congestion (Chapter 8), and an (optimal) on–line $O(\log n)$–approximation for the UFP. This settles open questions of Karp, Leighton, Rivest, Thompson, Vazirani, Vazirani [57], and of Awerbuch, Gawlick, Leighton, Rabani [12] respectively. We also develop algorithms for constructing *node–disjoint* paths on the two–dimensional mesh (Chapter 9); the performance guarantee of our algorithm essentially settles (to within a logarithmic factor) an open question of Aggarwal, Klawe, Lichtenstein, Linial, and Wigderson [2] that was raised in the context of VLSI layout.

A central theme in this work is the connection between the unsplittable flow problem and the closely related *fractional multicommodity flow problem*. We will discuss the latter in detail in Chapter 2; for now, it is sufficient to think of it as a variant of the UFP in which the "unsplittability" constraint is relaxed to allow the ρ_i units of demand associated with the pair (s_i, t_i) to be routed only partially, and to be split up over several paths. The basic optimization problems involving fractional flow can be solved in polynomial time, and thus it makes sense to investigate ways in which optimal fractional flows can be used to approximate the UFP. This general approach is typically referred to as *rounding*: we wish to "round off" the fractional solution we obtain to a 0/1-valued solution that produces an unsplittable routing. For example, the high–capacity UFP approximation of Raghavan and Thompson [94] is based on a direct rounding procedure, which we will sketch in Chapter 2.

Our approximation algorithm for the general single–source problem is based on a new rounding procedure for single–source fractional flow — this latter problem corresponds to the well–known *maximum flow problem*. For general multi–terminal problems, one can show that the optimal value of a fractional flow can far exceed the optimal value of an unsplittable flow — thus, a direct comparison to the optimum fractional flow will not be of use in designing a general approximation algorithm in this case. However, on the two–dimensional mesh and its generalization discussed above, we show that one can make use of the optimal fractional solution in designing an approximation algorithm. Our algorithm here uses the Raghavan–Thompson method on a high–capacity "simulated network" that we construct on top of the underlying mesh. To generalize this construction of a simulated network beyond the mesh itself, we require a multicommodity flow result of Okamura and Seymour [84], as well as some topological results on the disjoint paths problem due to Schrijver [104]. One interesting feature of the node–disjoint paths problem, from the perspective of these techniques, is that the gap between the fractional and unsplittable solutions can be very large, and thus the fractional solution provides very little information in designing our algorithm for the node–disjoint case (Chapter 9).

Certain of the results described below make use of the following definition. We say that a set of terminal pairs is a *permutation* if every vertex of the graph appears in exactly one pair. We say that it is a *partial permutation* if every vertex appears in at most one pair.

Single–Source Unsplittable Flow. In Chapter 3, we consider a very basic case of the unsplittable flow problem, which previous approximation techniques were unable to handle. This is the *single–source* UFP, in which all s_i are the same. When all ρ_i and c_e are equal, this problem can be reduced to the case of edge–disjoint s-t paths, the subject of Menger's theorem [82] and the max-flow min-cut theorem [34, 35]. But when we simply remove this condition, the single–source UFP becomes NP-complete (as indicated above, it trivially contains the *knapsack problem*); and in fact previous techniques are capable of providing an approximation algorithm only when every c_e is at least $\Omega(\log |E|)$ times every ρ_i.

Here we provide a constant–factor approximation for the general case of this problem. We also investigate what can be said about the case in which $\rho^* = \max_{i,e} \rho_i / c_e$ is small, as this is relevant to situations such as those covered by [94, 93, 10], when every connection consumes only a small fraction of the bandwidth on a link. We provide an approximation algorithm with a performance guarantee of $1 + O(\sqrt{\rho^*})$ — the crucial point being that this converges to 1 as ρ^* becomes smaller and smaller. We obtain similar approximation ratios for the problems of minimizing congestion and routing in rounds. For the latter problem, we make use of a matroid structure inherent in the single–source disjoint paths problem in order to develop the approximation algorithm. The connection with classical matroid algorithms appears to be interesting in its own right; its first application in approximation problems of this type was found by the author and Éva Tardos [63], for a related problem that we will not be discussing in this work.

As indicated above, the results are based on a new *rounding* procedure for the fractional version of the problem, which is precisely the classical *maximum flow problem.* The nature of the algorithms developed here makes them useful in a number of subsequent chapters, as they allow many algorithms designed with the MDP in mind to be converted into algorithms for the more general UFP.

Single–Source Problems in Directed Graphs. We also extend our algorithms for the single–source UFP to the case of directed graphs. Although the final results we obtain are essentially the same (with somewhat larger constants) as for the undirected case, the techniques required are more elaborate.

This is the only place in this work in which we deal with directed graphs, and we have two main motivations for doing so. First, much less is known about the disjoint paths problem in the directed case, and thus it is of interest to develop techniques that apply to directed graphs. Second, it is possible to reduce a general family of NP-complete "load–balancing" problems to the directed case of the single–source UFP. In this way our constant–factor approximation for this problem implies a general constant–factor approximation for any load–balancing problem of this type. Thus, as is the case with the classical max-flow min-cut theorem, we obtain applications of the single–source UFP that have essentially nothing to do with the issue of routing in networks.

The Greedy Algorithm. Chapter 5 is a short chapter which analyzes the performance of perhaps the simplest algorithm for the MDP — the *greedy algorithm*, which always routes a connection on the shortest available path. For the most part, we consider a slight variant of the greedy algorithm called the *bounded greedy algorithm* (BGA), which only routes a connection if there is a free path that is sufficiently short. We show that in a bounded–degree expander graph, the BGA provides an $O(\log n(\log\log n)^2)$–approximation for the MDP. The fact that this performance guarantee can be achieved by an algorithm that is both on-line and deterministic presents a somewhat striking contrast with lower bounds due to Bartal et al. [15] and Awerbuch et al. [10], which apply to classes of graphs other than expanders. For permutations on expanders, we show that the BGA is in fact an $O(\log n)$–approximation. This improves on the bound of $O(\log^2 n)$ that follows from the techniques of Aumann–Rabani [9] and Raghavan–Upfal [95]; these papers consider the routing of permutations on expanders in the off-line and randomized on–line settings respectively. The latter part of this chapter is based on joint work with Ronitt Rubinfeld [62].

Approximations on the Two–Dimensional Mesh. In Chapter 6, we develop approximation algorithms for the two–dimensional mesh graph. The two–dimensional mesh is a fundamental graph to consider both in the context of VLSI and the context of virtual circuit routing. The $n \times n$ two–dimensional mesh is the adjacency graph of the $n \times n$ chessboard; it has vertex set $\{(i, j) : 1 \leq i, j \leq n\}$, with (i, j) and (k, ℓ) joined by an edge iff $|k - i| + |\ell - j| = 1$. In some sense, it forms the basis of the study of VLSI layout, since most models of a VLSI chip represent it as an $n \times n$ array of square cells [57]. It is one of the most common parallel architectures; and many large–scale networks for high–speed communication are configured as mesh–like, nearly planar graphs [12, 22, 90, 66, 81, 117]. Finally, we note that the MDP remains NP-complete when the underlying graph is restricted to be the mesh [65].

We provide a constant–factor approximation for the UFP on the (uniform–capacity) two–dimensional mesh. This is the first constant–factor approximation for the UFP, or even the maximum disjoint paths problem, in any class of graphs other than trees. Using the same underlying technique, we obtain an on–line $O(\log n)$–approximation algorithm, which is optimal for on–line algorithms on the mesh. This settles an open question of Awerbuch, Gawlick, Leighton, and Rabani [12]. The underlying technique for both these algorithms is, roughly, the following. We impose a high–capacity "simulated network" on the mesh; the paths are then constructed in two stages: (i) using an algorithm for the high–capacity UFP (e.g. [94, 93, 10]), we choose "global routes" in the simulated networks; (ii) these global routes are then converted into paths in the mesh itself.

We also note that a straightforward application of the greedy set cover technique [24, 53, 77], in conjunction with the constant–factor approximation for the MDP, allows us to obtain an $O(\log n)$ approximation for the minimum partition of $\mathcal{T}$ into rounds. This improves on a bound of Aumann and Rabani [9].

Approximations on a Generalization of the Mesh. Much of the motivation for studying the mesh in the context of communication networks stems from the fact that it is a bounded–degree planar graph typical of many of the types of networks that arise in practice. Thus, it is important to develop routing algorithms here that are robust in the sense that they do not rely heavily on the fixed row/column structure of the mesh itself. With this in mind, we define in Chapter 7 the class of *densely embedded graphs*, which generalizes the mesh. In particular, it includes any class of even–degree graphs that can be embedded in the plane with a fixed boundary and uniformly bounded face size. We extend our constant–factor approximation algorithm to this class of graphs; much of the difficulty here is in adapting the construction of the underlying "simulated network," indicated above, to the setting of an arbitrary densely embedded graph. To do this, we make use of classical results of Okamura and Seymour [84] and Schrijver [104] on the disjoint paths problem. Our requirement that the vertex degrees be even arises throughout previous work on this problem, as will be seen in Chapter 2; it is not particularly limiting in practice, as it can be achieved by doubling every edge. The material in Chapters 6 and 7 is based on joint work with Éva Tardos [63, 64].

Minimizing Congestion. In Chapter 8, we consider the problem of minimizing congestion. Using the simulated network construction of Chapter 7, we obtain a constant–factor approximation for congestion in densely embedded graphs. Combined with the algorithm of Raghavan and Thompson [94], this settles an open question of Karp, Leighton et. al [57] for the special case of the mesh. This is based on joint work with Satish Rao and Éva Tardos [61].

Node–Disjoint Paths. Finally, all of the above has been concerned with *edge–disjointness* constraints. But a number of very basic and interesting questions arise when one requires paths that are *node–disjoint*. At the level of arbitrary graphs, these problems are very closely related; but when one is interested, for example, in the case of planar graphs, the problems are very different. In particular, one begins to encounter "topological" obstacles to the existence of paths that did not arise in the edge–disjoint case. Moreover, such problems arise very naturally in VLSI layout problems, which are often fraught with node–disjointness constraints.

In Chapter 9, we develop algorithms for finding node–disjoint paths on the two–dimensional mesh. These will not be proper approximation algorithms in the sense defined above; but they will be guaranteed to produce realizable sets within a polylogarithmic factor of the *bisection bound*. Specifically, we will show that every permutation on the $n \times n$ mesh contains a set of size $\Omega(\frac{n}{\log n})$ that is realizable by node–disjoint paths. This essentially (to within this logarithmic factor) closes a gap left open by Aggarwal, Klawe, Lichtenstein, Linial, and Wigderson [2, 3]. We also develop extensions of this algorithm for the case of "sparse" sets of terminal pairs on the mesh. The material in this chapter is based on joint work with Alok Aggarwal and David Williamson [5].

Figure 1-1 gives an approximate guide to the interdependence among Chapters 2–9. Chapter 2 can be used mainly as a reference; most of the subsequent chapters rely only on selected portions of it.

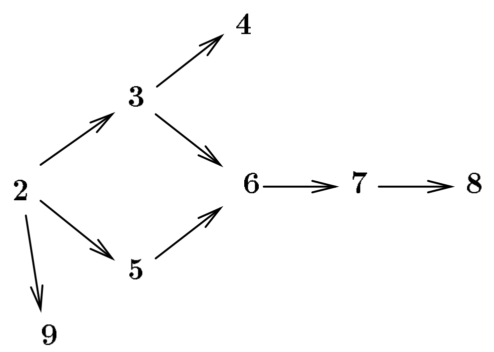

Figure 1-1: Interdependence among chapters

1.3 Some General Notation

Much of the notation that we require will be developed as it is needed; here we introduce some basic definitions that will be used throughout.

As before, let $G = (V, E)$ be an undirected graph, possibly with parallel edges. If $u, v \in V$, let $d(u, v)$ denote the least number of edges in a u-v path. For $v \in V$, and $r \geq 0$, let $B_r(v)$ denote the set of all $u \in V$ for which $d(v, u) \leq r$.

For $X \subset V$, let $G[X]$ denote the subgraph of G induced on the vertices in X. Let $\delta(X)$ denote the set of all edges with exactly one end in X. Let $\pi(X)$ denote the set of all vertices of X that are an end of an edge in $\delta(X)$. Let $X^o = X \setminus \pi(X)$, the "interior" of X. For two disjoint sets $X, Y \subset V$, let $\delta(X, Y)$ denote the set of edges with one end in X and the other in Y.

An *Eulerian graph* is one in which the degree of every vertex is even; its name commemorates the fact that the development of graph theory would have been postponed by a century had the street map of Königsberg been Eulerian. A *planar graph* is one that can be drawn in the plane without edge crossings; viewing the drawing of a planar graph as a subset of the plane, we observe that its complement consists of a finite number of connected components, called *faces*, of which the unbounded one is called the *outer face*.

Finally, we confess to using the terms "node" and "vertex" completely interchangeably.

Chapter 2

Background

Paths, paths, everywhere; a stamped–in network of paths spreading over the empty land, through long grass, through burnt grass, through thickets, down and up chilly ravines, up and down stony hills ablaze with heat . . .
— *Joseph Conrad,* Heart of Darkness.

None of the theorems discussed in this chapter are our own; we give attributions in almost all cases, unless the result is somehow part of the "folklore." Our development of this background material is not meant to be comprehensive; in particular, we devote the most attention to the results that we will use in subsequent chapters.

2.1 Menger's Theorem

As mentioned in Chapter 1, one of the most basic cases of the disjoint paths problem is that in which all (s_i, t_i) pairs are the same; that is, all connections requests are between two vertices $s, t \in V$. In this, the setting of Menger's classical theorem [82], one asks: What is the maximum number of mutually edge–disjoint s-t paths in G?

One way to provide evidence that there do not exist k edge–disjoint s-t paths is to exhibit a set $X \subset V$ such that

(i) $s \in X$ and $t \notin X$.

(ii) $|\delta(X)| < k$.

For if there were k edge–disjoint s–t paths, then by (i) each would have to pass through a different edge in $\delta(X)$; but this would contradict (ii). We will call such a set X an *s-t cut*, and define its *value* to be $|\delta(X)|$.

Menger's theorem is the non–trivial converse of the above observation.

Theorem 2.1.1 (Menger) *There exist k edge–disjoint s-t paths in G if and only if there does not exist an s-t cut of value less than k.*

There are many proofs known for Menger's theorem; when Ford and Fulkerson proved their generalization, the max-flow min-cut theorem [34, 35] (see Section 2.4), they provided a polynomial–time algorithm for finding the maximum number of edge–disjoint s–t paths in a graph G.

The formulation of Menger's theorem exposes a very elegant necessary condition for the existence of a solution to the disjoint paths problem, with arbitrary input G and $\mathcal{T}$.

> There does not exist a set $X \subset V$ such that
>
> (i) There is a set of j terminal pairs in $\mathcal{T}$, each with one end in X and the other in $V \setminus X$.
>
> (ii) $|\delta(X)| < j$.

We refer to this general statement as the *cut condition*, and the set X as a *cut*. X will be called a *non–trivial cut* if neither X nor $V \setminus X$ are empty. Note that in this general formulation of the cut condition, we do not require the entire set $\mathcal{T}$ of terminal pairs to be split by the cut. Also, it is a simple matter to show

Lemma 2.1.2 *The cut condition holds if and only if it holds for all sets X for which both $G[X]$ and $G[V \setminus X]$ are connected.*

(See e.g. [39].)

The cut condition is not sufficient for the solvability of the disjoint paths problem in general. For example, let us consider the *brick wall graph* of Figure 2-1. Speaking somewhat informally, we say that the general $n \times n$ brick wall graph G_n consists of n layers of vertices, connected as in the figure. Let $\mathcal{T}_n$ be a set of terminal pairs in G_n, such that all s_i appear from top to

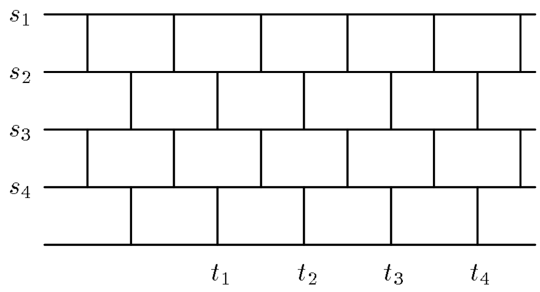

Figure 2-1: A bad example

bottom at the leftmost edge, and all t_i appear from left to right on the bottom layer. ($\mathcal{T}_4$ is pictured in the figure.) Then one can easily verify that the cut condition holds for $(G_n, \mathcal{T}_n)$; but if $1 \leq i < j \leq k$, then any s_i-t_i path must share an edge with any s_j-t_j path, and hence $\alpha(\mathcal{T}_n)$ is equal to 1. This in fact an interesting example on which to compare the objective functions α, χ, and ν — we have $\alpha(\mathcal{T}_n) = 1$, $\chi(\mathcal{T}) = n$, and $\nu(\mathcal{T}) = 2$, indicating among other things that the gap between ν and χ can be very large.

The special case of the MDP encompassed by Menger's theorem is quite robust in the sense that a number of seemingly more general cases reduce easily to it. As a primary example, consider the *single–source* MDP, in which all s_i are the same vertex $t \in G$ (though the t_i, for $i = 1, \ldots, k$, may be distinct). If we construct a new graph G' by joining an additional node t^* to each of $t_1, \ldots, t_k$, then we see that there exist k edge–disjoint paths from s to t^* if and only if the original problem has a solution. Thus, by Menger's theorem

Theorem 2.1.3 *An instance of the single–source disjoint paths problem has a solution if and only if the cut condition holds.*

Moreover, since there is an efficient algorithm for determining the maximum number of edge–disjoint s-t^* paths in the newly constructed graph, there is an efficient algorithm for the single–source maximum disjoint paths problem. On the other hand, it is interesting to recall that by the example in Chapter 1, the single–source unsplittable flow problem is NP-complete. We develop approximation algorithms for it in Chapter 3.

In later chapters, we will also make use of another general problem that can be reduced to Menger's theorem — that of computing *linkages* between sets. Let X and Y be disjoint subsets of V. By an X-Y *linkage* we mean a collection of edge–disjoint paths, each with one end in X and the other in Y. An X-Y linkage will be called *simple* if the endpoints of all the paths are distinct. A simple X–Y linkage of maximum size can be computed efficiently by attaching an additional vertex s to all the vertices in X, an additional vertex t to all the vertices in Y, and computing a maximum size set of edge–disjoint s-t paths in the newly constructed graph. To compute a (non–simple) X–Y linkage of maximum size, one defines Δ to be the maximum degree of the original graph, attaches s and t to their respective sets X and Y via Δ parallel edges to each vertex, and again computes a maximum size set of edge–disjoint s-t paths in the newly constructed graph. We will use $f(X,Y)$ (resp. $f_s(X,Y)$) to denote the size of a maximum (simple) X–Y linkage.

Karp's proof that the disjoint paths problem is NP-complete [56] gave compelling evidence that there does not exist a simple criterion for the existence of disjoint paths of the type suggested by the cut condition. In the next two sections, we discuss some special cases of the disjoint paths problem in which feasibility can be decided in polynomial time (and for which simple criteria are both necessary and sufficient); following this, we discuss a *fractional* version of the disjoint paths problem that can be solved in general in polynomial time.

2.2 Tractable Special Cases

One of the first natural special cases of the disjoint paths problem to be considered was that of a fixed number of terminal pairs. Even the case $|\mathcal{T}| = 2$ turned out to be quite non–trivial, and it was not until 1980 that three independent solutions emerged, due to Seymour [111], Shiloach [113], and Thomassen [116].

Theorem 2.2.1 (Seymour, Shiloach, Thomassen) *There is a polynomial-time algorithm for the disjoint paths problem in the case* $|\mathcal{T}| = 2$.

These algorithms also came with a fairly simple structural criterion for the existence of the paths, which we will not go into here; for perhaps the shortest proof of this criterion, the reader is referred to Section 2 of [100].

Naturally the next question was whether the problem is in polynomial time for any fixed value of $|\mathcal{T}|$. A polynomial time algorithm for this case was discovered by Robertson and

Seymour, as one of the main consequences of their Graph Minors series. The algorithm is given in [102], but the proof of its correctness requires much of the machinery developed throughout the series of papers.

Theorem 2.2.2 (Robertson–Seymour) *For every fixed k, there is a polynomial-time algorithm for the disjoint paths problem in the case $|\mathcal{T}| = k$.*

Indeed, for every k the algorithm runs in time $O(n^3)$. However, the $O()$ hides a very high dependence on k and very large constants. Finding a reasonably simple algorithm even for the case $k = 3$ remains an interesting open question. We will not be making use of Theorem 2.2.2, though later in this work we will relate some of the concepts we introduce to definitions from the Graph Minors series.

For the case of a variable number of terminal pairs, recall that the problem is NP-complete even for planar graphs. Thus the emphasis here, in developing special cases, is in finding some structured "arrangement" of the set of terminals that makes the problem more tractable. Historically one of the first such results was developed by Okamura and Seymour [84], in the following setting.

Let G be a planar graph, and Φ the outer face of G. Also, let us define the *augmented graph* G^+ to be the graph obtained from G by adding all (s_i, t_i) pairs as edges. Then Okamura and Seymour show

Theorem 2.2.3 (Okamura–Seymour) *If G^+ is Eulerian and all terminals lie on Φ, the outer face of G, then the set $\mathcal{T}$ is realizable in G if and only if the cut condition holds.*

The proof also yields a polynomial–time algorithm for finding the paths. A number of more efficient algorithms have been developed for this case, culminating in a recent linear time (and quite practical) algorithm of Wagner and Weihe [118].

The requirement that G^+ be Eulerian is called the *parity condition*, and it appears in much of the work on exactly solvable special cases. Though it seems somewhat mysterious, the reader can appreciate its importance by considering again the brick wall graph of Figure 2-1; it satisfies all the hypotheses of the Okamura–Seymour theorem aside from the parity condition, and yet the disjoint paths do not exist.

A result due to Frank [37] allows us to weaken the parity condition a little. For a set X, let $d(X)$ denote the set of terminal pairs with one end in X and the other in $V \setminus X$. We say that the cut condition holds *strictly* if $|\delta(X)| > |d(X)|$ for all non–trivial cuts X. (The cut condition itself requires only non–strict inequality.)

Theorem 2.2.4 (Frank) *Let G be planar, and Eulerian everywhere but its outer face Φ. Let $\mathcal{T}$ be a set of terminal pairs all lying on Φ. Then if the cut condition holds strictly, the set $\mathcal{T}$ is realizable.*

We will make use of Frank's extension of the Okamura–Seymour theorem in Chapter 7.

There are a number of subsequent results on the solvability of the disjoint paths problem in cases when the underlying graph is planar and the terminals have some "nice" arrangement. In particular, see the papers of Okamura [85] and Schrijver [103], both of which are discussed in Frank's survey [39].

There is a further elegant theorem along these lines due to Seymour [112], which we state here although we will not have occasion to use it.

Theorem 2.2.5 (Seymour) *Let G be planar, with terminal pairs $\mathcal{T}$, and suppose that G^+ is both planar and Eulerian. Then $\mathcal{T}$ is realizable in G if and only if the cut condition holds.*

Again, there is a polynomial–time algorithm accompanying this theorem.

2.3 Disjoint Paths on a Surface

We now discuss some special cases of the *node–disjoint* paths problem, when the underlying graph is embedded on a fixed surface. The techniques here will be used in Chapter 7, as part of our approximation for (edge–)disjoint paths in densely embedded graphs.

First we need some preliminary topological definitions; the reader is referred to an introductory textbook such as [80] for more detail. Let S^1 denote the unit circle, and S^2 the surface of the unit sphere in $\mathbf{R}^3$, as topological spaces. Let Σ denote a compact orientable surface; it is well-known that Σ may be obtained from S^2 by attaching a finite number of handles. By a *disc* we mean a set homeomorphic to the closed unit ball in $\mathbf{R}^2$, and by Σ-*disc*, we mean a subset of Σ homeomorphic to a disc. We also use the terms *curve* (continuous image of $[0,1]$) and *closed* *curve* (continuous image of S^1). Our definition of graph embedding is standard; a *face* of an embedded graph G is a connected component of $\Sigma \setminus G$, and we say G is *strongly embedded* if the closure of each face is a Σ-disc, and each face is bounded by a simple cycle of G.

(It is not difficult to produce an example of a graph embedding that is not a strong embedding; consider the following construction. Let C be a copy of K_3, with vertices v_1, v_2, v_3. Let C_i $(i = 1, 2, 3)$ also be a copy of K_3. We define a 9-node graph G by identifying v_i with a node of C_i, for $i = 1, 2, 3$. Then it is possible to embed G on the torus in such a way that the for each edge e of C, there is a face Φ in the embedding such that the sequence of edges bounding Φ contains e twice.)

In the results to follow, one can take all curves and closed curves to be polygonal, and all surfaces to be *triangulated*; in this way, we avoid various topological exotica.

Two curves $\mathcal{R}$ and $\mathcal{R}'$ are said to be *homotopic* on Σ if, informally speaking, one can be "continuously deformed" into the other without moving the endpoints. Technically, $\mathcal{R}$ and $\mathcal{R}'$ are homotopic, written $\mathcal{R} \sim \mathcal{R}'$, if there exists a continuous function $\Psi : [0,1] \times [0,1] \to \Sigma$ such that for all $t \in [0,1]$,

(i) $\Psi(t, 0) = \mathcal{R}(t)$, $\Psi(t, 1) = \mathcal{R}'(t)$.

(ii) $\Psi(0, t) = \mathcal{R}(0)$, $\Psi(1, t) = \mathcal{R}(1)$.

We define homotopy for closed curves using a function $\Psi : S^1 \times [0,1] \to \Sigma$ that satisfies condition (i). A closed curve is said to be *null-homotopic* on Σ if it is homotopic to a point; it is well–known (see e.g. [99]) that a closed curve is null-homotopic if and only if it is contained in a Σ-disc.

Robertson and Seymour proved the following theorem [98]. Let G be a planar graph, and $\mathcal{T}$ a set of terminal pairs on the outer face of G. If we are looking for a realization of $\mathcal{T}$ by *node–disjoint paths*, then we observe that the following are obstacles to the existence of such paths.

(i) Two pairs of terminals *cross*; that is, they appear in the cyclic order s_i, s_j, t_i, t_j on the outer face.

(ii) (A type of cut condition.) One can draw a simple closed curve $\mathcal{R}$ in the plane, meeting the graph only at vertices, so that the number of terminals pairs with one end inside $\mathcal{R}$ and the other end outside $\mathcal{R}$ is greater than the number of times that $\mathcal{R}$ meets G.

For in case (ii), each path in a realization of G would have to pass through a different vertex in $\mathcal{R} \cap G$; and there are not enough such vertices.

In [98] it is shown that these conditions form a criterion for the (non-)existence of paths. The proof is not very difficult.

Theorem 2.3.1 (Robertson–Seymour) *If G is planar and $\mathcal{T}$ lies on the outer face, then $\mathcal{T}$ is not realizable by node–disjoint paths if and only if one of (i) or (ii) above holds.*

In [98], a similar result is also proved for graphs embedded on a cylinder.

Schrijver proved what is in some sense the ultimate generalization of this result, for graphs embedded on an arbitrary surface [104]. Say that a curve $\mathcal{R}$ on Σ is *G-normal* if it meets the drawing of G only at vertices. Let $\mathrm{cr}(\mathcal{R}, G)$ denote the number of times $\mathcal{R}$ meets G, and $\mathrm{mincr}(\mathcal{R}, G)$ the minimum of $\mathrm{cr}(\mathcal{R}', G)$ over all $\mathcal{R}' \sim \mathcal{R}$. For two curves $\mathcal{R}_0$ and $\mathcal{R}_1$, we define $\mathrm{cr}(\mathcal{R}_0, \mathcal{R}_1)$ to be the number of crossings of $\mathcal{R}_0$ and $\mathcal{R}_1$, and $\mathrm{mincr}(\mathcal{R}_0, \mathcal{R}_1)$ to be the minimum of $\mathrm{cr}(\mathcal{R}_0', \mathcal{R}_1')$, over all $\mathcal{R}_0' \sim \mathcal{R}_0$ and $\mathcal{R}_1' \sim \mathcal{R}_1$.

Let $A_1, \ldots, A_k$ be closed curves on Σ. We are interested in finding node–disjoint cycles $C_1, \ldots, C_k$ on Σ such that $C_i \sim A_i$ for $i = 1, \ldots, k$. Consider the following two necessary conditions for the existence of such cycles, analogous to obstacles (i) and (ii) above.

(i) There exist closed curves $A_1', \ldots, A_k'$ on Σ such that $A_i \sim A_i'$, and the A_i' are mutually disjoint.

(ii) For every closed curve $\mathcal{R}$ on Σ we have

$$\mathrm{cr}(\mathcal{R}, G) \geq \sum_{i=1}^{k} \mathrm{mincr}(\mathcal{R}, A_i).$$

It turns out that these are not sufficient for the existence of the disjoint cycles in G [105]; to obtain a set of sufficient conditions, we must consider the following type of parity condition.

Say that a closed curve $\mathcal{R}$ is *odd* with respect to G and $\{A_i\}$ if

$$\mathrm{cr}(\mathcal{R}, G) \not\equiv \sum_i \mathrm{cr}(\mathcal{R}, A_i) \pmod 2.$$

Note that the parity of the number of crossings is invariant under homotopy; so if $\mathcal{R}$ is odd and $\mathcal{R} \sim \mathcal{R}'$, then $\mathcal{R}'$ is odd as well. Thus for every odd curve, a necessary condition for the existence of the disjoint cycles is that we have strict inequality in (ii). But this does *not* necessitate changing the statement of (ii); for if we know that $\mathrm{cr}(\mathcal{R}, G) \geq \sum_{i=1}^{k} \mathrm{mincr}(\mathcal{R}, A_i)$ for an odd curve $\mathcal{R}$, then strict inequality already holds.

The problem arises when we concatenate two odd curves with a common point. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two odd closed curves, with a common point $x \in \Sigma \setminus G$. Define $\mathcal{R}$ to be the closed curve obtained by concatenating the parametrizations of $\mathcal{R}_1$ and $\mathcal{R}_2$ at the common point x, in the natural way. We say that $\mathcal{R}$ is the *doubly odd* closed curve composed of $\mathcal{R}_1$ and $\mathcal{R}_2$. Now, $\mathrm{cr}(\mathcal{R}, G)$ and $\sum_i \mathrm{mincr}(\mathcal{R}, A_i)$ *do* have the same parity, and so it is possible for these quantities to be equal. But because of the parity constraints, we know that if $C_1, \ldots, C_k$ are node–disjoint cycles in G homotopic to $A_1, \ldots, A_k$, then we must have

$$\mathrm{cr}(\mathcal{R}_1, G) > \sum_{i=1}^{k} \mathrm{cr}(\mathcal{R}_1, C_i),$$

$$\mathrm{cr}(\mathcal{R}_2, G) > \sum_{i=1}^{k} \mathrm{cr}(\mathcal{R}_2, C_i),$$

whence

$$\begin{aligned}
\mathrm{cr}(\mathcal{R}, G) &= \mathrm{cr}(\mathcal{R}_1, G) + \mathrm{cr}(\mathcal{R}_2, G) \\
&> \sum_{i=1}^{k} \mathrm{cr}(\mathcal{R}_1, C_i) + \sum_{i=1}^{k} \mathrm{cr}(\mathcal{R}_2, C_i) \\
&= \sum_{i} \mathrm{cr}(\mathcal{R}, C_i) \\
&\geq \sum_{i} \mathrm{mincr}(\mathcal{R}, A_i).
\end{aligned}$$

Thus, we obtain our third necessary condition.

(iii) For every doubly odd closed curve $\mathcal{R}$ on Σ we have

$$\mathrm{cr}(\mathcal{R}, G) > \sum_{i=1}^{k} \mathrm{mincr}(\mathcal{R}, A_i).$$

Schrijver proves that (i), (ii), and (iii) are indeed sufficient [104].

Theorem 2.3.2 (Schrijver) *Given closed curves $A_1, \ldots, A_k$ on Σ, there exist node–disjoint cycles $C_1, \ldots, C_k$ in G such that $C_i \sim A_i$ if and only if (i), (ii), and (iii) hold.*

For us, it will be useful to have a version of this theorem for surfaces with boundary. By a *surface Σ with boundary* we mean a compact surface Σ from which the interiors of one or more disjoint closed Σ-discs have been removed. The boundaries of these closed Σ-discs will be called the *boundary components* of Σ. In this setting, we are interested in the following related problem. We are given curves $A_1, \ldots, A_k$ whose endpoints coincide with vertices of G on boundary components of Σ, and we want to find node–disjoint paths $P_1, \ldots, P_k$ in G such that $P_i \sim A_i$ for $i = 1, \ldots, k$.

Here, our three necessary conditions remain almost exactly the same, except that we must incorporate the fact that Σ has a boundary. So call a curve $\mathcal{R}$ *essential* if it is either a closed curve or a curve with both endpoints on boundary components of Σ. Our necessary conditions now become

(i) There exist curves $A'_1, \ldots, A'_k$ on Σ such that $A_i \sim A'_i$, and the A'_i are mutually disjoint.

(ii) For every *essential* curve $\mathcal{R}$ on Σ we have

$$\mathrm{cr}(\mathcal{R}, G) \geq \sum_{i=1}^{k} \mathrm{mincr}(\mathcal{R}, A_i).$$

(iii) For every doubly odd closed curve $\mathcal{R}$ on Σ we have

$$\mathrm{cr}(\mathcal{R}, G) > \sum_{i=1}^{k} \mathrm{mincr}(\mathcal{R}, A_i).$$

Now the following is known [107], and can in fact be shown by a simple reduction to Theorem 2.3.2

Theorem 2.3.3 (Schrijver) *Given curves $A_1, \ldots, A_k$ on Σ, whose endpoints coincide with vertices of G on boundary components of Σ, there exist node–disjoint paths $P_1, \ldots, P_k$ in G such that $P_i \sim A_i$ if and only if conditions (i), (ii), and (iii) above hold.*

2.4 Fractional Flow

As we begin to turn to the issue of approximation, we introduce the notion of *fractional flow*, which due to its tractability is of great help in the design and analysis of approximation algorithms for the MDP.

As before, we are given a graph G and a set $\mathcal{T}$ of k terminal pairs. Let us consider the case in which each terminal pair (s_i, t_i) has an associated demand ρ_i. A *fractional flow* for $\mathcal{T}$ in G is defined as follows. For each pair (s_i, t_i), we choose

(i) a set of s_i-t_i paths $P_i^1, \ldots, P_i^{q_i}$, and

(ii) associated non-negative weights $y_i^1, \ldots, y_i^{q_i} \in \mathbf{R}$ such that $z_i = \sum_j y_i^j \leq \rho_i$.

Of the ρ_i units of flow from s_i to t_i, a total of y_i^j is routed on path P_i^j (and so $\rho_i - z_i$ units of flow are not routed at all). We say that such a flow is a *fractional routing* if $z_i = \rho_i$ for each $i = 1, \ldots, k$; that is, the total demand for each terminal pair is routed on the associated flow paths. A fractional flow is said to be *feasible* if it satisfies the capacity constraints

$$\sum_{(i,j):e \in P_i^j} y_i^j \leq c_e. \tag{2.1}$$

We can now ask whether $\mathcal{T}$ is *fractionally realizable* (or *fractionally feasible*) — by this we mean that there exists a feasible fractional routing. We can also consider fractional version of some of the optimization problems considered in the introduction:

(i) We can ask — what is the maximum amount of flow that can be feasibly shipped from the sources to the sinks in G? For this latter problem, we will denote the maximum in question by $\alpha^f(\mathcal{T})$. Of course, using the above notation, the unsplittable flow problem is obtained by simply adding the requirement that each q_i be equal to 0 or 1; and for those terminal pairs with $q_i = 1$, we must have $y_i^1 = \rho_i$.

(ii) The *congestion* of a fractional routing is defined to be the maximum amount of flow sent across any edge; using the above notation, we can write it as

$$\max_e \sum_{(i,j):e \in P_i^j} y_i^j.$$

Similarly, the *relative congestion* of a fractional routing is defined to be

$$\max_e \frac{\sum_{(i,j):e\in P_i^j} y_i^j}{c_e}.$$

We use $\nu^f(\mathcal{T})$ (resp. $\nu_0^f(\mathcal{T})$) to denote the minimum congestion (resp. relative congestion) of a fractional routing of $\mathcal{T}$.

These are often termed *fractional multicommodity flow problems*; the terminal pairs (s_i, t_i) are viewed as different commodities that must be simultaneously transported on the same underlying network. Both here and in what follows, when we speak of "maximum flow" (question (i) of the previous paragraph), we do not allow more than ρ_i units of flow to be shipped from s_i to t_i. That is, we do not consider the version of the problem in which one can ship more flow than the total demand. This is mainly for the sake of a uniform presentation, and it can be finessed by allowing $\rho_i = \infty$ for some of the pairs (s_i, t_i).

There is also a notion of a cut condition in this setting; we can define it as follows. For a set $X \subset V$, recall that $\delta(X)$ denotes the set of all edges with exactly one end in X, and $d(X)$ denotes the set of all terminal pairs with exactly one end in X. Define

$$\begin{aligned} c(X) &= \sum_{e\in\delta(X)} c_e \\ \rho(X) &= \sum_{i:(s_i,t_i)\in d(X)} \rho_i \end{aligned}$$

Then the cut condition is the following.

For all $X \subset V$, we have $c(X) \geq \rho(X)$.

Although the cut condition is necessary for fractional realizability, it is not in general sufficient. A fairly large amount of work has been done on showing the sufficiency of "approximate" cut conditions; we will discuss this further in Chapter 10, and also refer the reader to [69, 60, 44, 89, 74] for more details.

The well–studied maximum s-t flow problem deals with fractional flow — it is simply the above problem in the case when all (s_i, t_i) pairs in $\mathcal{T}$ are the same. This is the case considered in the fundamental work of Ford and Fulkerson [34, 35], who showed that in this setting the cut condition is necessary and sufficient for fractional realizability. Note that since we are assuming that all (s_i, t_i) pairs are the same, the definition of fractional flow allows us to assume further that $|\mathcal{T}| = 1$ — by merging all the s-t demand into one single value ρ. (It can be split up fractionally again when it is routed.)

Thus we obtain the *max-flow min-cut theorem.*

Theorem 2.4.1 (Ford–Fulkerson) *Let G be an arbitrary graph, and $\mathcal{T}$ consist of a single pair (s,t) with demand ρ. Then $\mathcal{T}$ is fractionally realizable in G if and only if $c(X) \geq \rho$ for all sets $X \subset V$ for which $s \in X$, $t \notin X$.*

The original proof of Ford and Fulkerson yielded a polynomial time algorithm for deciding realizability for the case in which ρ and all c_e are bounded integers. A polynomial–time algorithm for the general case was first obtained independently by Dinic [30] and Edmonds and Karp [32]. The efficiency of maximum flow algorithms has become a major issue in algorithm design; see for example the survey of Goldberg et al. [48]. All these algorithms solve the more general *maximization* version of this problem, in which one wishes to ship the maximum amount of flow possible from s to t.

The max-flow min-cut theorem appears to be simply a fractional analogue of Menger's theorem; but in fact it generalizes it, due to the following second result of Ford and Fulkerson.

Theorem 2.4.2 (Ford–Fulkerson) *Let G be an arbitrary graph, and $\mathcal{T}$ consist of a single pair (s,t) with demand ρ. Suppose further that ρ is an integer, and all capacities are integers. Then $\mathcal{T}$ is fractionally realizable if and only if it is realizable by ρ flow paths each carrying one unit of flow.*

Due to this theorem, we can feel justified in using the term "s-t flow" in a graph with only unit capacities to denote a set of edge–disjoint s-t paths.

From the s-t version of these results, it is again possible to infer more general results, by analogy with the method we saw in Section 2.1. For example, the single–source maximum flow problem, when all s_i are the same vertex, can be reduced to the s-t maximum flow problem as before. It is worth stating this as an explicit corollary of Theorems 2.4.1 and 2.4.2.

Corollary 2.4.3 *Let G be an arbitrary graph, and $\mathcal{T}$ a set of k terminal pairs such that $s_i = s \in V$ for $i = 1, \ldots, k$. Then* *(i) $\mathcal{T}$ is fractionally realizable if and only if the cut condition holds.*

(ii) A maximum fractional flow can be found in polynomial time.

(iii) If all demands ρ_i are equal to a common value ρ, and all capacities are multiples of ρ, then there is a maximum fractional flow that is an unsplittable flow. In particular, in this case, if $\mathcal{T}$ is fractionally realizable then it is realizable in the unsplittable sense.

Now let us consider the fractional multicommodity flow problem in its full generality. It turns out that unlike the MDP and UFP, the fractional multicommodity flow problem can be solved in polynomial time. The most direct way to see this is to verify that it can be written as a polynomial–size linear program, and thus solved optimally using a polynomial–time linear programming algorithm [58, 55, 49]. Recently, faster and more combinatorial algorithms have been developed to provide $(1+\varepsilon)$–approximations for this problem (see e.g. [88, 68, 13]).

Since maximum fractional flows can be computed in polynomial time, we will be making use of them in designing approximation algorithms. As with many fundamental problems of combinatorial optimization, a question that naturally arises is the extent to which the existence of a fractional solution to a problem implies the existence of an *integer* solution. And, in a related vein, one can ask how different the optimum over fractional solutions can be in comparison to the optimum over (the more restricted set of) integer solutions.

In our case, at least when all ρ_i are equal to 1, integer solutions correspond to feasible *unsplittable* flows, and so this comes down to a comparison of $\alpha(\mathcal{T})$ and $\alpha^f(\mathcal{T})$. Following the example of Figure 2-1, consider an $n \times n$ brick wall graph with n terminal pairs arranged as shown, and all capacities and demands equal to 1. There is a fractional flow of total value $\frac{1}{2}n$, while the maximum unsplittable flow has value 1, since disjoint paths cannot be found between any two pairs simultaneously. Thus, we have the following fact (noted in [44] and elsewhere).

Theorem 2.4.4 *Even when all demands and capacities are equal to* 1*, the gap between the maximum fractional flow value $\alpha^f(\mathcal{T})$ and the maximum unsplittable flow value $\alpha(\mathcal{T})$ can be as large as a factor of $\frac{1}{2}k$ for k terminal pairs.*

Of course, the size of this gap is extremely sensitive to the nature of the underlying graph; if we double every edge in Figure 2-1, then Frank's extension of the Okamura–Seymour theorem applies and there is no gap at all. Much of our work here is motivated by the investigation of the relationship between fractional and unsplittable flow, both as a means to provide approximation algorithms and as an end in itself.

Seymour formulated the following conjecture [112], which can easily be motivated by examples such as Figure 2-1 and theorems such as 2.2.3 and 2.2.5.

> *Seymour's Conjecture: If all capacities and demands are equal to* 1*, and a set* $\mathcal{T}$ *of terminals is fractionally realizable, then it is realizable in half–integers. That is, there is a fractional flow shipping all the demand, in which all path weights are equal to* $\frac{1}{2}$*.*

This conjecture is false, disproved by Lomonosov [76]; in fact, it is now known that the "half–integer" flow problem is NP-complete [83]. However, it seems quite possible that structural results of this type may indeed hold for the unsplittable flow problem in general graphs, and this is clearly an interesting issue for future work.

2.5 The High–Capacity UFP

We can now introduce the *randomized rounding* technique of Raghavan and Thompson [94], which leads to Raghavan's algorithm for the high–capacity UFP [93]. Recall that in this special case of the UFP, we have an n-node, m–edge graph with minimum edge capacity at least $\Omega(\log m)$ times the maximum demand. Re–scaling, let assume that the maximum demand ρ^* is at most 1, and the minimum edge capacity c_* is at least $\varepsilon \log m$.

Raghavan's algorithm is now simply the following.

(i) Solve the fractional maximization problem; for the pair (s_i, t_i) we obtain paths $P_i^1, \ldots, P_i^{q_i}$, and weights $y_i^1, \ldots, y_i^{q_i} \in \mathbf{R}$ such that $z_i = \sum_j y_i^j \leq \rho_i$.

(ii) Define $\overline{y}_i^j = y_i^j / \rho_i$ and $\overline{z}_i = z_i / \rho_i$.

(iii) Choose a constant *scaling factor* μ. (The dependence of μ on ε will determined below.)

(iv) Finally, route the pair (s_i, t_i) on path P_i^j with probability $\mu \overline{y}_i^j$, and don't route (s_i, t_i) at all with probability $1 - \mu \overline{z}_i$.

In analyzing this algorithm, we must verify two things. First, we argue that with high probability, no capacity constraint is violated by the resulting paths. And second, we argue that the total amount of flow routed is within a constant factor of the optimum.

To verify the first of these, we need the following result from [93]; it is a bound on the sum of independent random variables in the spirit of results of Chernoff [23] and Hoeffding [51].

Theorem 2.5.1 (Raghavan) *Let $a_1, \ldots, a_r \in [0,1]$. Let $X_1, X_2, \ldots, X_r$ be independent Bernoulli trials with $EX_j = p_j$ and $\Psi = \sum_i a_i X_i$; so $E\Psi = m = \sum_i a_i p_i$. Then for $\delta > 0$ we have*

$$Pr[\Psi > (1+\delta)m] < \left[\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right]^m.$$

We specialize this to the form in which we will use it as follows.

Corollary 2.5.2 *Let $a_1, \ldots, a_r \in [0,1]$. Let $0 < \mu < \alpha \leq 1$, and $p_1, \ldots, p_r \in [0,1]$. Let $X_1', X_2', \ldots, X_r'$ be independent Bernoulli trials with $EX_j' = \mu p_j$ and $\Psi' = \sum_i a_i X_i'$. Let $m = \sum_i a_i p_i$. Then*

$$Pr[\Psi' > \alpha m] < (e\alpha^{-1}\mu)^{\alpha m}.$$

This follows by applying the bound of Theorem 2.5.1 with m set to μm and δ set to $\alpha\mu^{-1} - 1$.

We apply the corollary to prove that the capacity of edge e is not violated after rounding, with high probability. First, to make the computation easier, note that on any edge e which is not fully saturated by the fractional solution, we may add artificial demands (and corresponding amounts of flow) across e in order to saturate it. Thus, we will assume that all edges are saturated. Let p_i denote the fraction of demand of (s_i, t_i) that is routed across edge e by the fractional solution, and X_i' the indicator random variable for the event that (s_i, t_i) is routed on a path that uses edge e. Note that with $a_i = \rho_i$, $m = \sum_i a_i p_i$ is simply the capacity c_e (by our saturation assumption), and Ψ' is now a random variable equal to the amount of flow crossing edge e in the solution produced by Raghavan's algorithm. The "bad event" that the capacity of edge e is violated has probability at most $(e\mu)^{c_e}$, by Corollary 2.5.2; since $c_e \geq \varepsilon \log m$, we can set $\mu = e^{-1}4^{-1/\varepsilon}$ and obtain a probability bounded by m^{-2}. Thus, invoking the union bound, the probability that the capacity of any edge is violated is at most m^{-1}.

Now, the expected amount of flow routed by the approximate solution is exactly μ times the fractional optimum; since it cannot exceed the fractional optimum, Markov's inequality implies that with probability at least $\frac{\mu}{2-\mu}$, the approximate solution has value at least $\frac{1}{2}\mu$ times the fractional (and hence the unsplittable) optimum.

Since each randomized rounding experiment is an independent trial that succeeds with constant probability, we obtain

Theorem 2.5.3 (Raghavan) *Iterating the above algorithm $\Theta(k)$ times provides a constant–factor approximation to the high–capacity* UFP *with probability $1 - 2^{-k}$.*

There are a number of things worth noting about this algorithm. First, the requirement that all capacities be large relative to the demands was crucial for the experiment to succeed. For consider a graph G consisting of two nodes s and t, with m parallel edges of unit capacity. There are m terminal pairs, each equal to (s,t) with unit demand, and our fractional solution (unfortunately) splits the flow for each pair evenly across the m edges. Now, our randomized rounding experiment corresponds to the well–known combinatorial problem of throwing m objects into m containers with equal probability; if we scale down the flow by only a constant factor μ before performing the rounding, then a standard argument shows that $\Theta(\frac{\log m}{\log\log m})$ paths will be routed across a single edge with high probability.

In fact, the requirement that all capacities be large was crucial for the nature of the result we proved. The randomized rounding algorithm in this case was able to produce an unsplittable solution whose value was within a constant factor of the fractional optimum. But we saw in the previous section, via Theorem 2.4.4, that no such result is possible for the unsplittable flow problem in general. Thus, the high–capacity assumption played a significant role here; approximation techniques for the UFP with small capacities cannot work so directly from the fractional solution.

Recently, Awerbuch, Azar, and Plotkin [10] gave an *on–line* $O(\log n)$–approximation for the high–capacity UFP in general graphs; see Section 2.7.

2.6 Approximations for the MDP

As noted above, approximation results are much more limited for the MDP, and for the UFP when one drops the "high–capacity" assumption. Prior to this work, the only class of graphs for which a constant–factor approximation algorithm had been obtained was the class of capacitated trees, by Garg, Vazirani, and Yannakakis [44]. Of course, deciding the realizability of a set of terminals $\mathcal{T}$ is easy, since there is a unique path for every (s_i, t_i) pair. But if all demands are equal to 1, and capacities can take on the values 1 and 2, the maximization problem becomes NP-complete. Garg et al. consider the case of trees in which all demands are equal to 1, and capacities are arbitrary. They show that in this setting, the ratio between the fractional and integral optima is at most 2, and using this they obtain a 2–approximation algorithm.

Essentially the only other (off–line) approximation algorithms for the MDP are an $O(\log n)$–approximation for the two–dimensional mesh, obtained independently of this work by Aumann and Rabani [9], and the approximations for expander graphs discussed in Section 2.8.

These above results concern the MDP, and not the more general unsplittable flow problem. There has been work on the problem of minimizing congestion, with arbitrary unsplittable demands, when the underlying graph G is a cycle. (So the problem is, for each demand, to decide whether to send it clockwise or counter–clockwise around the cycle.) This question was raised in the context of SONET ring routing by Cosares and Saniee [26]. Schrijver, Seymour, and Winkler showed that on the cycle, one can always find (in polynomial time) an unsplittable routing for the demands such that the amount of flow crossing any edge is at most $\frac{3}{2}\rho^*$ units more than is required for the fractional optimum [108]. Thus, the approximation guarantee they obtain here is additive, rather than multiplicative.

2.7 On–Line Routing Algorithms

Recall that an on–line algorithm for the MDP (or UFP) is one that is presented with a sequence of terminal pairs in order; when each arrives, it must be irrevocably rejected or routed on a path through G without knowledge of future input. In analyzing such algorithms, especially when randomization is involved, one must be careful that things are defined correctly; thus, we will define our on–line model in slightly more detail.

As is standard in this area, we can picture the execution of an on–line algorithm as a game played between the algorithm and an *adversary* that generates the input. First the underlying graph G is fixed. Then the adversary (using knowledge of the algorithm) creates an ordered set $\mathcal{T}$ of terminal pairs and presents them one at a time. For a deterministic algorithm, the performance is measured as the ratio of the optimum $\alpha(\mathcal{T})$ to the number of pairs routed by the algorithm.

Now, for a randomized on–line algorithm, we allow the adversary to have knowledge of the algorithm, but *not* of the random choices that will be made during the algorithm's execution. The algorithm's performance is the ratio of the optimum $\alpha(\mathcal{T})$ to the *expected* number of pairs routed by the algorithm. It will turn out that randomized on–line algorithms are often capable of achieving significantly better performance guarantees than deterministic ones; this is perhaps not surprising, given that randomization on the part of the algorithm to some extent mitigates the power of adversarially chosen input. We note that there are other ways of modeling randomized on–line algorithms, and a number of subtle issues arise [16]; however, we will not get into these here.

As mentioned in Section 2.5, Awerbuch, Azar, and Plotkin [10] have given a deterministic $O(\log n)$–approximation for the high–capacity UFP in general graphs. They also prove that $\Omega(\log n)$ is a lower bound for the performance ratio of any deterministic on–line algorithm for this problem. Although Awerbuch et al. only prove their performance guarantee in terms of the optimum unsplittable flow, it holds as well in terms of the optimum fractional flow. We prove this in Chapter 6, where we need a slightly stronger analysis of their algorithm for use in our on–line algorithm for the UFP in densely embedded graphs.

For the MDP, on the other hand, a number of fairly straightforward lower bounds show that no deterministic on–line algorithm can achieve a polylogarithmic performance ratio in trees or meshes [10, 17]. Thus, the on–line algorithms developed here have essentially all been randomized.

An $O(\log n)$–approximation for the case of trees was obtained by Awerbuch, Bartal et al. [11]; this approximation ratio was improved to $O(\log d)$ for trees of diameter d by Awerbuch, Gawlick, Leighton, and Rabani [12]. Awerbuch, Gawlick, et al. [12] also obtained an on–line $O(\log n \log\log n)$–approximation for the two–dimensional mesh. Finally, the technique of

Raghavan and Upfal [95] discussed in the following section provides an $O(\log^2 n)$–approximation for bounded–degree expander graphs.

Now it is natural to ask whether there could be a randomized on–line algorithm with a performance ratio of $O(\log^{O(1)} n)$ for *all* graphs, and this was recently answered in the negative by Bartal, Fiat, and Leonardi [15]. They showed that there is a family of graphs $\{G_n\}$ and an $\varepsilon > 0$ such that no on–line algorithm can be better than an $O(n^\varepsilon)$–approximation on G_n. In fact, the graphs G_n used by Bartal et al. are simply the $n \times n$ brick wall graphs of Figure 2-1. Their on–line lower bound is highly sensitive to the structure of these graphs; and it must be so, since in Chapter 7 we provide an on–line $O(\log n)$–approximation for any Eulerian planar graph with (loosely speaking) the structure of the brick wall graph. As our class includes, for example, the brick wall graph in which every edge is given capacity 2, a natural next question is the following — is there any $\Omega(n^\varepsilon)$ lower bound for on–line algorithms in any family of graphs with edge capacity 2, instead of 1?

2.8 Expander Graphs

Finally, we discuss a line of work related to constructing disjoint paths in bounded–degree expander graphs. We will make use of this in Chapter 5, when we analyze the performance of the greedy algorithm for constructing disjoint paths in expanders.

First, we define an expander graph. We say that an n–node graph G is an *α–expander* if, for a constant $\alpha > 0$, every set $X \subset V$ of cardinality at most $\frac{1}{2}n$ satisfies the following *expansion condition*:

$$|\delta(X)| \geq \alpha |X|. \tag{2.2}$$

Expander graphs must be highly non–planar; in particular, any class of graphs embeddable on a fixed surface cannot contain arbitrarily large expanders [47, 6]. All the expanders we consider here will have bounded maximum degree Δ. At times below we will use the term "sufficiently strong expanders"; we intend here to avoid lengthy technical details, and essentially mean expander graphs with an implicit lower bound on the constant α.

It turns out that in any (sufficiently strong) expander graph, *any* set of terminals of up to a certain size is realizable. The first result of this type was proved by Peleg and Upfal [86];

their bound on the size of the allowable terminal set was considerably strengthened by Broder, Frieze, and Upfal [20]. In order to obtain paths that are edge–disjoint, both these papers require very strong expansion guarantees. However, for more moderate expanders, [20] also proves the following. Recall that by the *congestion* of a set of paths, we mean the maximum of number of paths containing any fixed edge.

Theorem 2.8.1 (Broder–Frieze–Upfal) *For every $\alpha \geq 1$, there exists a constant θ such that the following holds. In any Δ–regular n-node α-expander graph G, any partial permutation containing at most $O(n/\log^{\theta} n)$ terminal pairs can be routed with congestion at most 2, using paths of length $O(\log n)$.*

The proof also provides a randomized polynomial time algorithm for constructing the paths.

If one allows super–constant congestion, then even larger terminal sets can be guaranteed to be realizable; in particular Leighton and Rao prove the following [70]. (The result in [70] is stated only for strong expanders, but the proof as given there applies to α-expanders for arbitrary $\alpha > 0$.)

Theorem 2.8.2 (Leighton–Rao) *For every $\alpha > 0$, every bounded–degree α–expander G has the following property: every partial permutation $\mathcal{T}$ containing at most $O(n/(\log n \log\log n))$ terminals pairs can be routed in G with congestion $O(\log\log n)$, using paths of length $O(\log n)$.*

Recent unpublished work of Broder, Frieze, and Upfal also addresses this trade–off between the size of the terminal set and the allowed amount of congestion [21].

As a by-product of their earlier fundamental work on multicommodity flow, Leighton and Rao [69] established the following result.

Theorem 2.8.3 (Leighton–Rao) *Let G be a bounded–degree, n–node α-expander. Then any partial permutation in G can be routed with congestion $O(\log n)$ using paths of length $O(\log n)$.*

Using this result, Aumann and Rabani observed the following [9]. Say that we route a partial permutation in such a graph G, and then build a *conflict graph* $\mathcal{K}$ on the set of paths used in the routing. Now, each path meets $O(\log n)$ other paths on each of its $O(\log n)$ edges, so the conflict graph $\mathcal{K}$ has maximum degree $O(\log^2 n)$. Hence we can color it with $O(\log^2 n)$ colors. But each color class is a set of edge–disjoint paths; thus they show

Theorem 2.8.4 (Aumann–Rabani) *Let G be a bounded–degree, n–node α-expander. Then any partial permutation in G can be partitioned into $O(\log^2 n)$ realizable sets.*

It follows that there is an $O(\log^2 n)$–approximation for the MDP in bounded–degree expanders: one simply computes the partition above and takes the largest realizable set produced.

Raghavan and Upfal [95] arrive at Theorems 2.8.3 and 2.8.4 via a completely different technique, more closely related to the methods of Broder, Frieze, and Upfal [20]. Given a permutation on a bounded-degree expander G, they produce a routing for it iteratively as follows. For terminal pair (s_i, t_i), a point $x_i \in V$ is chosen at random from the stationary distribution of the random walk on G. The path for (s_i, t_i) is then constructed from an s_i-x_i random walk of length $\Theta(\log n)$, followed by an x_i-t_i random walk of length $\Theta(\log n)$. Raghavan and Upfal show that the set of paths produced this way, for all pairs, has congestion $O(\log n)$ with high probability; thus Theorem 2.8.3 follows.

A variant of this randomized algorithm produces an *on–line* $O(\log^2 n)$–approximation for the MDP in bounded–degree expander graphs. Specifically, when a terminal pair arrives, we construct the above type of random walk; we then route the pair if the path constructed does not intersect any previously routed path. It is not hard to show, given that the conflict graph $\mathcal{K}$ on the paths will have maximum degree $O(\log^2 n)$ with high probability, that this is an $O(\log^2 n)$–approximation.

In Chapter 5, we develop an on–line $O(\log n \log\log^2 n)$–approximation for expanders that is deterministic.

2.9 Directed Graphs

All of the above discussion has been in terms of *undirected graphs*, and essentially all of the present work is concerned with the case of undirected graphs. However, in Chapter 4, we will extend our single–source unsplittable flow algorithm to the case of *directed graphs*. Thus, in this section, we discuss some of the known results on disjoint paths in directed graphs that we will be using.

First of all, by a *directed graph*, we mean an undirected graph $G = (V, E)$ together with a choice of a *direction* for every edge. Thus, if $e = (u, v) \in E$, this choice of direction fixes one of u or v to be the *head* of e, and the other to be the *tail*. A *path* in a directed graph must now respect the directions of its edges, in the obvious way. So in the context of the MDP, if (s_i, t_i) is a terminal pair in G, we are now seeking a directed path *from* s_i *to* t_i. This introduces a substantial new complication to the disjoint paths problem, and changes the nature of what can be done algorithmically. As one striking example, the problem of deciding whether $(G, \mathcal{T})$ is realizable, when G is directed and $|T| = 2$, was shown to be NP-complete by Fortune, Hopcroft, and Wyllie [36]. This is in contrast to the algorithmic results of [111, 113, 116, 102] discussed in Section 2.2.

Indeed, the disjoint paths problem is much less well understood in general in the directed setting than it is in the undirected setting. Let us indicate here which of the results of the previous sections have analogues in the directed case, and which do not. First, we formulate a directed cut condition as follows. If $X \subset V$, we define $\delta^-(X)$ to be the set of edges entering X, and $\delta^+(X)$ to be the set of edges leaving X. In the capacitated case, we define $c^-(X)$ and $c^+(X)$ to be the total capacity of edges entering and leaving X, respectively. Finally, we define $\rho^-(X)$ to be the total demand of terminal pairs (s_i, t_i) for which $s_i \notin X$ and $t_i \in X$; we define $\rho^+(X)$ analogously. Thus, a natural phrasing of the cut condition is as follows.

For all $X \subset V$, we have $c^-(X) \geq \rho^-(X)$.

(The cut condition for c^+ and ρ^+ is equivalent, by considering $V \setminus X$ in place of X.)

Menger's theorem and the max-flow min-cut theorem still hold in the directed case (and are, in fact, perhaps easier to prove for directed graphs). That is, the cut condition is necessary and sufficient for realizability when all terminal pairs are the same. By the reduction discussed in Sections 2.1 and 2.4, we obtain results for more general "single–source" problems as well. We state the directed analogue of Corollary 2.4.3 here, as we will be using it in Chapter 4.

Corollary 2.9.1 *Let G be an arbitrary directed graph, and $\mathcal{T}$ a set of k terminal pairs such that $s_i = s \in V$ for $i = 1, \ldots, k$. Then*

(i) $\mathcal{T}$ is fractionally realizable if and only if the cut condition holds.

(ii) A maximum fractional flow can be found in polynomial time.

(iii) If all demands ρ_i are equal to a common value ρ, and all capacities are multiples of ρ, then there is a maximum fractional flow that is an unsplittable flow. In particular, in this case, if $\mathcal{T}$ is fractionally realizable then it is realizable in the unsplittable sense.

The general fractional multicommodity flow problem in directed graphs can still be formulated as a linear program of polynomial size, and hence solved in polynomial time. It follows that the high–capacity UFP approximation still works in the directed case. In fact, the on–line algorithm due to Awerbuch et al. [10] also works in the directed case.

Essentially none of the other results of the previous sections hold for the directed case. A limited number of results of the flavor of those in Section 2.2 have been proved; see for example [29, 38, 110, 106]. Similarly, nothing is known for directed graphs embedded on general surfaces that is as general as Schrijver's theorem 2.3.2. Thus it remains an interesting general problem to try identifying further special cases of the disjoint paths problem that are solvable in polynomial time in directed graphs. Finally, the approximation algorithms described in Sections 2.6 and 2.7 that do not assume large edge capacities do not apply in the directed case.

Chapter 3

Single–Source Unsplittable Flow

The less I seek my source for some definitive
The closer I am to fine ...
— *Indigo Girls,* Indigo Girls.

Let us consider the unsplittable flow problem in the case in which all terminal pairs share a common vertex. That is, the set of terminal pairs is $\mathcal{T} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$, and all s_i are equal to a common vertex s. For notational simplicity in this case, we will sometimes use $\mathcal{D}$ to denote the set $\{t_1, \ldots, t_k\}$; we will refer to these as the *terminals*, the vertex s as the *source*, and specify the problem using the triple $(G, \mathcal{D}, s)$.

We recall from Section 2.4 that fractionally, this is precisely the classical maximum flow problem. Moreover, the *integrality theorem* stated there, Theorem 2.4.2, shows that when all demands are the same value ρ, and all capacities are multiples of ρ, then the maximum unsplittable flow value is equal to the maximum fractional flow value. It follows that in this case, the unsplittable optimum is characterized by the cut condition, and one can find this optimum in polynomial time.

But this is far from the case for the general single–source UFP. In particular, the following result is immediate; it is similar to one of the examples used in Chapter 1. (Very similar hardness results can also be found in [26, 108].) We say that a problem is *strongly NP-complete* if it remains NP-complete when all numbers in the input are written in unary; i.e., it is NP-complete even when the input numbers are polynomially bounded.

Proposition 3.0.1 *The single–source* UFP *is NP-complete, even for a graph* G_0 *consisting of two vertices with two parallel edges joining them. It is strongly NP-complete, even for a graph* G_1 *consisting of two vertices with* m *parallel edges, where* m *is part of the input.*

Proof. The following *partition problem* is NP-complete [56]: given a set S of positive integers $a_1, \ldots, a_n$, with $D = \sum_i a_i$, decide whether there is a set $S' \subset S$ with

$$\sum_{j : a_j \in S'} a_j = \frac{1}{2} D.$$

Given an instance of the partition problem, with numbers $a_1, \ldots, a_n$ and $D = \sum_i a_i$, we reduce it to the following single–source UFP $(G_0, \mathcal{D}, s)$. The underlying graph G_0 has node set $\{u, v\}$, with two parallel edges of capacity $\frac{1}{2}D$ each. We define $s = v$, $t_i = u$, and $\rho_i = a_i$ for $i = 1, \ldots, n$. This problem has a feasible solution if and only if the partition problem has a solution.

To prove strong NP-completeness, we note that the following *bin packing problem* is strongly NP-complete [43]: given given a set S of positive integers $a_1, \ldots, a_n$, and numbers B and D, decide whether there is a partition of S into at most B sets, each of which sums to at most D. Given such an instance of the bin–packing problem, we reduce it to the following single–source UFP $(G_1, \mathcal{D}, s)$. The underlying graph G_0 has node set $\{u, v\}$, with B parallel edges of capacity D each. We define $s = v$, $t_i = u$, and $\rho_i = a_i$ for $i = 1, \ldots, n$. This problem has a feasible solution if and only if the bin packing problem has a solution. ■

Since the problem is NP-complete, we are interested in providing approximation algorithms for it. We will consider both the maximization and congestion versions of the problem, as introduced in Chapter 1. We obtain constant–factor approximations for both these versions. We also consider the extent to which tighter approximation guarantees can be obtained when the maximum demand is small compared to the minimum edge capacity; here we show how our approximation ratios converge to 1 as this demand–to–capacity ratio goes to 0. In all cases, our results follow from a relation between the fractional and unsplittable optima for single–source problems.

We also obtain a constant–factor approximation algorithm for the problem of routing in rounds, in the single–source case. Moreover, if the ratio of the maximum demand to the minimum capacity is sufficiently small, then our approximation ratio for this problem is 2.

Since we saw above that it is NP-hard to distinguish a set of terminals routable in a single round from one that is routable in two, one cannot hope in this case for an approximation ratio better than 2. However, it is possible that one could be able to obtain an approximation to the minimum number of rounds that differs from the optimum by an *additive* error, and we leave this as an interesting open question.

For notational convenience, we will use $\alpha(G, \mathcal{D}, s)$ to denote the quantity $\alpha(\{(s, t_i) : t_i \in \mathcal{D})$ defined in Chapter 1 — that is, the maximum amount of realizable demand for the problem $(G, \mathcal{D}, s)$. We define $\alpha^f(G, \mathcal{D}, s)$, $\nu(G, \mathcal{D}, s)$, $\nu^f(G, \mathcal{D}, s)$, and $\chi(G, \mathcal{D}, s)$ analogously.

As mentioned in Section 2.6, results of a similar flavor have been developed by Schrijver, Seymour, and Winkler for the case of multi–terminal problems on a uniform–capacity cycle [108]. For the problem of minimizing congestion, their results also provide an approximate solution that approaches the optimum value as the demand–to–capacity ratio goes to 0. Our basic approach is quite different, since we are able to use the fact that all terminal pairs have a common source, but must deal with an arbitrary underlying graph.

Techniques

To highlight some of the complications that arise in designing approximation algorithms for the single–source UFP, it is worth considering the obstacles that arise in trying to adapt standard maximum flow techniques to our setting. Two of the main methods used to prove the max-flow min-cut theorem are *augmentation* — iteratively construct a maximum flow by increasing flow on some edges and re-routing across others — and *induction* — find a cut whose incident edges are saturated, contract the side of the cut that does not contain the source, and proceed by induction. However, it seems very difficult to define suitable "approximate" analogues of these methods in the setting of unsplittable flow. There are a number of basic reasons for this — in particular, both methods rely very heavily on the fact that in constructing a fractional flow, it is enough to maintain knowledge only of the amount of flow f_e crossing each edge e. In our setting, on the other hand, one must keep track of the *set of demand values* ρ_i crossing each edge; thus, for example, it matters whether $f_e = 1$ because e is carrying one unsplittable unit of flow, or because it is carrying two unsplittable half–units of flow.

In terms of other methods, we saw in the previous chapter that the randomized rounding method of Raghavan and Thompson [94] requires the minimum capacity to exceed the maximum demand by a factor of at least $\Omega(\log |E|)$; thus it is inapplicable to the general setting we are considering here. Finally, it seems that an approach based on setting all demands to the same value, invoking Corollary 2.4.3, and then restoring all demands to their original values is inherently incapable of providing a constant–factor approximation, since it introduces terms that depend on the ratio between the largest and smallest demands in $\mathcal{D}$.

Our basic method in this paper is closest to the last of these ideas — we reduce the problem to one in which all demands are approximately the same by a *grouping* technique. Specifically, we cover the graph G with small trees in such a way that (a) no edge is contained in too many trees, and (b) the amount of demand contained in each tree is approximately the same. We then build the unsplittable flow in two parts: we first use Corollary 2.4.3 to find unsplittable flow paths for designated *leader nodes* in each tree; and we concatenate these flow paths with paths connecting the terminals in $\mathcal{D}$ to the leaders of their trees.

It turns out that in the undirected case, essentially any such "tree cover" of the graph that satisfies a few simple conditions is sufficient to make this work. Things are much more complicated in the directed case, which we consider in Chapter 4. Although there is a directed analogue of Corollary 2.4.3 to work with, one must choose the tree cover very carefully now to make sure that the source has a flow of large enough value to the leaders of the resulting trees. It is this careful construction of a "good" tree cover that will require most of the work in the algorithm for directed graphs.

3.1 Coalitions

Recall our *balance assumption* (1.2) that $\rho_i \leq c_e$ for all i, e. By re-scaling, we will assume that the minimum edge capacity is equal to 1, and define $\rho^* = \max_i \rho_i < 1$.

By a *tree in G*, we mean a subset of the vertices of G that induces a connected subgraph, together with a spanning tree for these vertices. If T is a tree, in which each node $v \in T$ has an associated non–negative *weight* ω_v, we define a *center node* of (T, ω) to be a node $u \in T$ such that $T \setminus \{u\}$ has no component X for which

$$\sum_{v \in X} \omega_v > \frac{1}{2} \sum_{v \in T} \omega_v.$$

It is well–known that every weighted tree contains a center node. (One nice proof of this, which can be found in [97], is the following. Suppose that (T, ω) has no center node; then for each node u of T there is a unique component X_u of $T \setminus \{u\}$ that contains more than half the total weight in T. Since T has fewer edges than nodes, the pigeonhole principle implies that there exist nodes u and v for which the (unique) edge e in $\delta(X_u)$ and $\delta(X_v)$ is the same. It follows that e has ends u and v, and that $X_u \cap X_v = \phi$. But this is a contradiction, since each of X_u and X_v contains more than half the total weight in T.)

We now introduce the notion of a *coalition*, which will serve as our way of grouping demand in G. A coalition will be a set of trees, no more than two of which contain any edge, with each terminal assigned to some tree that contains it.

Definition 3.1.1 *A* (weak) coalition *in $(G, \mathcal{D}, s)$ is a pair $(\{T_1, \ldots, T_r\}, \tau)$, where each T_i is a tree in G, $\tau : \mathcal{D} \to \{T_1, \ldots, T_r\}$ is a function, and the following properties are satisfed.*

(i) The trees are half–disjoint*, by which we mean that each edge of G belongs to at most two of the T_i.*

(ii) For each $i = 1, \ldots, k$, $t_i \in \tau(t_i)$.

We will be particularly interested in coalitions for which the total demand assigned to each tree is approximately the same:

Definition 3.1.2 *A* strong coalition *in $(G, \mathcal{D}, s)$ of order σ is a pair $(\{T_1, \ldots, T_r\}, \tau)$, such that the following properties are satisfed.*

(i) $(\{T_1, \ldots, T_r\}, \tau)$ is a coalition in G.

(ii) For each $j = 1, \ldots, r$, either $s \in T_j$ or we have

$$\sigma \leq \sum_{i : \tau(t_i) = T_j} \rho_i \leq \sigma + \rho^*. \tag{3.1}$$

The trees of a strong coalition for which inequality (3.1) does not hold will be called the *exceptional trees.* Note that all exceptional trees, by property (iii), contain s.

First we show that strong coalitions always exist.

Lemma 3.1.3 *For every $\sigma \geq 0$, there exists a strong coalition in $(G, \mathcal{D}, s)$ of order σ.*

Proof. We double every edge of G so that it becomes Eulerian, and let C be the edge sequence of an Eulerian circuit in G beginning at s. Let $t_{\pi(1)}, \ldots, t_{\pi(k)}$ denote the order in which the terminals are first encountered while following C, where π is a permutation on $\{1, \ldots, k\}$. We partition the sequence $t_{\pi(1)}, \ldots, t_{\pi(k)}$ into r intervals, such that the sum of the values of ρ_{i_j} in each interval but the last is between σ and $\sigma + \rho^*$. This is possible since each ρ_{i_j} is at most ρ^*. Let $\mathcal{D}_j$ denote the set of terminals in the j^{th} such interval, and let C_j denote the set of edges of C that were traversed between the first visit to a vertex in $\mathcal{D}_j$ and the last visit to a vertex in $\mathcal{D}_j$. In the case $j = r$, define C_r to be the set of edges of C that were traversed between the first visit to a vertex in $\mathcal{D}_r$ and the end of the Eulerian circuit (at s).

Now, C_j contains the edge set of a spanning tree on a superset of $\mathcal{D}_j$; let T_j be any such spanning tree. Moreover, since $G[C_r]$ contains s, we choose T_r so that it contains s. Define the function $\tau : \mathcal{D} \to \{T_1, \ldots, T_r\}$ by setting $\tau(t_i) = T_j$ if $t_i \in \mathcal{D}_j$.

We claim first that $(\{T_1, \ldots, T_r\}, \tau)$ is a coalition. Property (ii) of Definition 3.1.1 is immediate from the construction; to see (i), we observe that each edge e of G appears only twice in C; so it appears in at most two of the C_j and thus in at most two of the trees T_j. Now, the total demand assigned to each tree by τ is between σ and $\sigma + \rho^*$, except possibly for the tree T_r, which contains s. Thus $(\{T_1, \ldots, T_r\}, \tau)$ is a strong coalition of order σ. ■

If Γ is a (weak) coalition in $(G, \mathcal{D}, s)$, with trees $T_1, \ldots, T_r$, we define $U(\Gamma)$ to be any set of vertices $\{u_1, \ldots, u_r\}$ such that

(i) if T_i is not exceptional, then u_i is a center node of T_i, and

(ii) if T_i is exceptional, then $u_i = s$.

We will refer to the node $u_i \in U(\Gamma)$ as the *leader* of the tree T_i. u_i will be called *exceptional* if the associated tree T_i is exceptional, and *non–exceptional* otherwise. We define a single–source UFP $(G, U(\Gamma), s)$ by associating with each $u_i \in U(\Gamma)$ any demand that does not exceed the total demand assigned by Γ to the tree T_i. We now show that one can route this set of nodes $U(\Gamma)$ without much more relative congestion than was possible for the original set of terminals.

Lemma 3.1.4 *Let Γ be a coalition in $(G, \mathcal{D}, s)$, and suppose that $(G, \mathcal{D}, s)$ has a feasible fractional routing. Consider the problem $(G, U(\Gamma), s)$ as defined in the previous paragraph, and suppose that the maximum demand associated with any $u_i \in U(\Gamma)$ is at most ε. Then $(G, U(\Gamma), s)$ has a fractional routing of relative congestion at most $1 + \varepsilon$.*

Proof. The feasible fractional routing for $\mathcal{D}$ in G assigns to each $t_j \in \mathcal{D}$ a set of t_j-s paths $Q_j^1, \ldots, Q_j^{q_j}$ and non-negative weights $y_j^1, \ldots, y_j^{q_j}$ such that $\sum_\ell y_j^\ell = \rho_j$. We use these paths to define a fractional flow for $U(\Gamma)$ in G.

For $u_i \in U(\Gamma)$, let ε_i denote the amount of demand associated with u_i in the problem $(G, U(\Gamma), s)$, and let ε_i' denote the total amount of demand associated with terminals of $\mathcal{D}$ that are assigned to the tree T_i:

$$\varepsilon_i' = \sum_{j:\tau(t_j)=T_i} \rho_j.$$

Recall that by the definition of the problem $(G, U(\Gamma), s)$, we have $\varepsilon_i \leq \varepsilon_i'$. Now if $u_i \in U(\Gamma)$ is the leader of an exceptional tree, then $u_i = s$ and hence we do not need to construct any paths at all. Otherwise, for each $t_j \in \tau^{-1}(T_i)$, let R_{ij} denote the unique u_i-t_j path in T_i. We now define the fractional flow paths for u_i as follows. For those j such that $t_j \in \tau^{-1}(T_i)$, and $\ell = 1, \ldots, q_j$, we define P_{ij}^ℓ to be the concatenation of R_{ij} and Q_j^ℓ. The set of paths for u_i is then

$$\{P_{ij}^\ell : t_j \in \tau^{-1}(T_i),\ 1 \leq \ell \leq q_j\}.$$

We set the weight associated with the path P_{ij}^ℓ to be

$$y_j^\ell \cdot (\varepsilon_i / \varepsilon_i') \leq y_j^\ell. \tag{3.2}$$

We define the *Q-part* of a path P_{ij}^ℓ to be the portion consisting of edges of Q_j^ℓ, and the *R-part* of P_{ij}^ℓ to be the portion consisting of edges of R_{ij}.

We must verify that this set of flow paths, for all non–exceptional u_i, has relative congestion at most $1+\varepsilon$. First observe that the set of all Q-parts of these flow paths contributes a relative congestion of at most 1: these Q-parts are in one-to-one correspondence with the original (feasible) fractional flow paths for $(G, \mathcal{D}, s)$, and the weight assigned to each Q-part is less than or equal to the weight assigned to the corresponding fractional flow path for $(G, \mathcal{D}, s)$, by Equation 3.2. The set of all R-parts of these paths contributes a congestion of at most ε to each edge. To see this, observe that since each non–expectional u_i is a center node of its associated tree, the total demand across any edge in G due to R-parts of paths for u_i is at most $\frac{1}{2}\varepsilon_i \leq \frac{1}{2}\varepsilon$. Since no edge of G is contained in more than two trees of Γ, the total demand across any edge in G due to the R-parts of all flow paths is at most ε. Since all edges in G have capacity at least 1, it follows that the total *relative* congestion due to the R-parts of all paths is also at most ε. Adding the bounds for the two parts of each path, one obtains the statement of the lemma. ∎

One can also prove this lemma by multiplying the capacities in G by $1+\varepsilon$ and then verifying the cut condition. We thank Michel Goemans for suggesting the proof given above.

3.2 Congestion

In this section, we use strong coalitions to obtain three related results, in the single–source case: a bound on the ratio between the optimum relative congestion for fractional and unsplittable flows, an approximate max-flow min-cut theorem for unsplittable flow, and an approximation algorithm for the minimum congestion of an unsplittable flow.

Theorem 3.2.1 *Let $(G, \mathcal{D}, s)$ be fractionally feasible. Then it has an unsplittable routing of relative congestion at most* $1 + 4\sqrt{\rho^*} + 4\rho^* = (1 + 2\sqrt{\rho^*})^2$.

Proof. We set $\sigma_0 = \frac{1}{2}\sqrt{\rho^*} \geq \frac{1}{2}\rho^*$, and let Γ be a strong coalition in $(G, \mathcal{D}, s)$ of order σ_0. As before, choose a set of nodes $U(\Gamma)$. To each non–exceptional node $u_i \in U(\Gamma)$, we associate a demand of σ_0; to each exceptional u_i we associate a demand of 0.

Let G' denote the graph obtained from G by multiplying all capacties by $1 + \sigma_0$, and then rounding each capacity up to the nearest multiple of σ_0. Note that each resulting capacity c'_e satisfies

$$c'_e \leq (1 + \sigma_0)c_e + \sigma_0 \leq (1 + 2\sigma_0)c_e.$$

Now, since $(G, \mathcal{D}, s)$ is assumed to be fractionally feasible, it follows from Lemma 3.1.4 that $(G', U(\Gamma), s)$ is fractionally feasible. But all demands in $(G', U(\Gamma), s)$ are equal to 0 or σ_0, and all capacities are multiples of σ_0, so by Corollary 2.4.3 there is a feasible unsplittable flow for $(G', U(\Gamma), s)$.

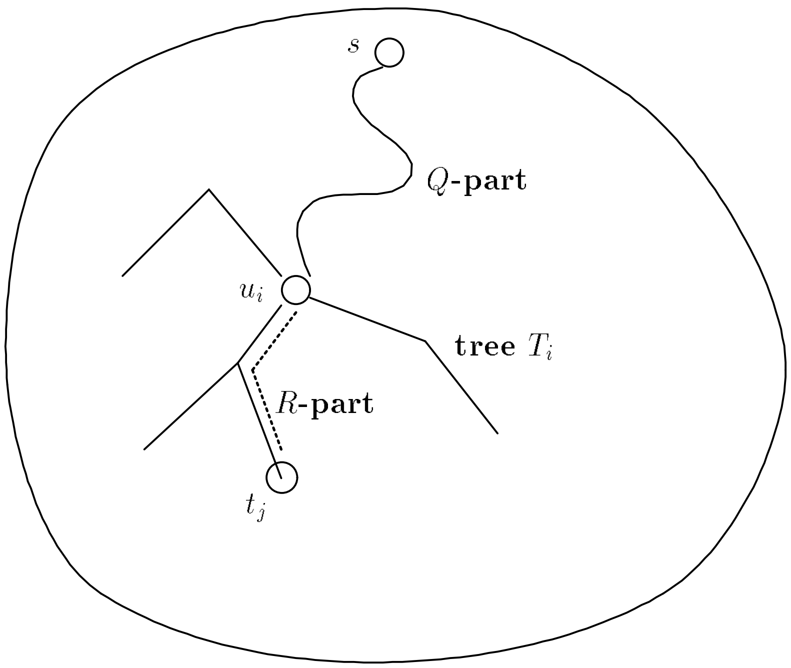

Figure 3-1: Constructing the unsplittable flow

Let Q_i, for $i = 1, \ldots, r$, denote the u_i-s path associated with this feasible unsplittable flow. We now describe the construction of the unsplittable flow in $(G, \mathcal{D}, t)$, by specifying a path for each terminal t_i. So let $t_j \in \mathcal{D}$, and suppose $\tau(t_j) = T_i$. We define R_j to be the unique t_j-u_i path in T_i. We then define P_j, the flow path for t_j, to be the concatenation of R_j and Q_i. We define the *Q-part* of P_j to be the portion consisting of edges of Q_i, and the *R-part* of P_j to be the portion consisting of edges of R_j.

It remains only to compute the relative congestion of the flow defined by the paths $\{P_j\}$. The above arguments showed that if each Q_i were to carry at most σ_0 units of flow, then the relative congestion on edge e due to the Q-parts of all flow paths in G would be at most $1+2\sigma_0$. But by part (ii) of Definition 3.1.2, each Q_i may carry up to $\sigma_0 + \rho^*$ units of flow, and thus the relative congestion on e due the Q-parts of all paths is at most

$$\frac{\sigma_0 + \rho^*}{\sigma_0} \cdot (1 + 2\sigma_0) = (1 + 2\sqrt{\rho^*})(1 + \sqrt{\rho^*}) = 1 + 3\sqrt{\rho^*} + 2\rho^*. \tag{3.3}$$

Now, by part (i) of Definition 3.1.1, e belongs to at most two of the trees in Γ. Thus by part (ii) of Definition 3.1.2, the total amount of flow across e due to the R-parts of all paths is at most $2\sigma_0 + 2\rho^*$. Since $c_e \geq 1$, the total relative congestion on e due to the R-parts of all flow paths is at most

$$2\sigma_0 + 2\rho^* = \sqrt{\rho^*} + 2\rho^* \tag{3.4}$$

Adding (3.3) and (3.4), the total relative congestion on e is at most

$$1 + 4\sqrt{\rho^*} + 4\rho^*,$$

from which the theorem follows. ■

When $\rho^* = 1$, so that we are making no assumptions about having "small" demands (aside from Equation (1.2)), the congestion bound of Theorem 3.2.1 is equal to 9. We note that with a less clean proof, one can slightly improve the constant in Theorem 3.2.1, as follows. Our construction of strong coalitions in Lemma 3.1.3 results in at most one exceptional tree. As a consequence, if e is contained in two trees, then one is non–exceptional, and hence the node u_i for that tree will be a center node. From this it follows that the total flow across edge e due to the R-parts of all flow paths is at most $\sigma_0 + \rho^*$ (from the one exceptional tree that might contain e) plus $\frac{1}{2}\sigma_0 + \frac{1}{2}\rho^*$ (from the other tree containing e, which must be non–exceptional), and hence is at most $\frac{3}{2}\sigma_0 + \frac{3}{2}\rho^*$. Setting $\sigma_0 = \sqrt{\frac{2\rho^*}{7}}$, we obtain a bound of $1 + \sqrt{14\rho^*} + \frac{7}{2}\rho^*$, which is approximately 8.25 when $\rho^* = 1$. For the sake of simplicity, we will continue to use the bound stated in Theorem 3.2.1.

Of course, a natural goal is to determine the tightest possible bound for the minimum relative congestion of an unsplittable routing of $(G, \mathcal{D}, s)$, when it is fractionally feasible. The following example shows that it must be at least $2 - o(1)$. Let G consist of vertices s and t, with k parallel edges of unit capacity between them. Let $\mathcal{D}$ consist of $k+1$ copies of the vertex t, with each terminal in $\mathcal{D}$ having demand $1 - \frac{1}{k+1}$. Then $(G, \mathcal{D}, s)$ is fractionally feasible, but the minimum congestion of an unsplittable routing of $(G, \mathcal{D}, s)$ is $2 - \frac{2}{k+1}$. Whether this example gives the worst possible gap between the fractional and unsplittable congestions is an interesting open question.

We say that the cut condition is *met by a factor of* β if $c(X) \geq \beta\rho(X)$ for all sets X not containing s. We now use Theorem 3.2.1 to show that if the cut condition is met by a large enough factor, then there is a feasible unsplittable flow. This result will be useful in Chapter 6.

Corollary 3.2.2 *Suppose that in* $(G, \mathcal{D}, s)$ *we have* $\rho^* < \frac{1}{4}$, *and the cut condition is met by a factor of at least* $\beta = (1 - 2\sqrt{\rho^*})^{-2}$. *Then* $(G, \mathcal{D}, s)$ *has a feasible unsplittable solution.*

Proof. We define G' to be the graph obtained from G' by dividing down all capacities by a factor of β. In the resulting problem $(G', \mathcal{D}, s)$, the ratio of the maximum demand to the minimum capacity is at most $\beta\rho^*$, and so by Theorem 3.2.1 there is an unsplittable flow for $(G', \mathcal{D}, s)$ of relative congestion at most $(1 + 2\sqrt{\beta\rho^*})^2 = \beta$. This is a feasible unsplittable flow in $(G, \mathcal{D}, s)$. (Note that we required $\rho^* < \frac{1}{4}$ so that β would be well–defined.) ■

Finally, we give an approximation algorithm for the minimum congestion of a single–source unsplittable flow on an arbitrary graph G.

Corollary 3.2.3 *There is a* 9*–approximation to* $\nu(G, \mathcal{D}, s)$ *in an arbitrary graph* G.

Proof. We can assume that all edges have capacity 1, and that the maximum demand is $\rho^* < 1$. In polynomial time, we first compute a fractional flow in $(G, \mathcal{D}, s)$ of minimum congestion ν^*. We know that ν^* is a lower bound on the optimal unsplittable congestion, and we know that ρ^* is also a lower bound on this optimum (since some edge must carry ρ^* units of flow). Thus, we set $c = \max(\nu^*, \rho^*)$, and assign all edges a capacity of c. Since there is a feasible fractional flow with this choice of capacity, we know by Theorem 3.2.1 that there is an unsplittable flow of congestion at most $(1 + 2\sqrt{\rho^* c^{-1}})^2 \leq 9$, from which the approximation ratio follows. ■

3.3 Maximization

We now turn from the problem of minimizing congestion to the basic unsplittable flow problem as defined in Chapter 1: we wish to maximize total amount of flow that can be sent subject to the capacity constraints. Recall that the optimum for this problem, in the single–source case, is denoted $\alpha(G, \mathcal{D}, s)$. We obtain a constant–factor approximation for $\alpha(G, \mathcal{D}, s)$; and since our performance guarantee will in fact be in terms of the fractional optimum, it will imply a constant–factor gap between the values of the fractional and unsplittable optima.

Again, we let $(G, \mathcal{D}, s)$ denote a single–source UFPwith minimum edge capacity 1. If all demands have roughly the same value, then it is not difficult to obtain a constant–factor approximation algorithm by simply assigning the same demand value to all terminals and applying Corollary 2.4.3. This is the content of Lemmas 3.3.1 and 3.3.2.

We use these lemmas to take care of demands that are "sufficiently large" — in particular, those larger than a constant ρ_0 that we will define below. For the set of all demands smaller than ρ_0, we require a technique involving strong coalitions (Theorem 3.3.3). Specifically, we first compute a maximum flow on the set of leaders of a strong coalition; we then construct an unsplittable flow on the terminals in $\mathcal{D}$ so that for each leader routed by the initial maximum flow, we are able to route a constant fraction of the demand in its tree. This will provide us with a constant–factor approximation ratio.

Recall that $\rho^* = \max_i \rho_i$. We define $\rho_* = \min_i \rho_i$.

Lemma 3.3.1 *There is a polynomial–time algorithm producing an unsplittable flow of value at least $\frac{1}{2}\rho_*/\rho^*$ times $\alpha^f(G, \mathcal{D}, s)$.*

Proof. Write $y = \rho_*/\rho^*$. Let $(G, \mathcal{D}, s)$ be an instance of the single–source UFP. Let G' denote the graph obtained from G by first rounding all capacities down to the nearest multiple of ρ^*, and then multiplying them by y. Thus, each capacity in G' is between $\frac{1}{2}y$ and y times what it is in G. Let $\mathcal{D}'$ denote the set of terminals obtained from $\mathcal{D}$ by assigning each a demand of ρ_*. We first observe that $\alpha^f(G', \mathcal{D}', s)$ is at least $\frac{1}{2}y$ times $\alpha^f(G, \mathcal{D}, s)$. To see this, take any fractional flow in $(G, \mathcal{D}, s)$ and multiply all the flow values by a factor of $\frac{1}{2}y$. The resulting flow satisfies all capacity constraints in G', and it does not route more than ρ_* units of flow to any terminal in $\mathcal{D}'$; thus it is a feasible fractional flow in $(G', \mathcal{D}', s)$.

But in $(G', \mathcal{D}', s)$, all demands are equal to ρ_*, and all capacities are multiples of ρ_*; thus by Corollary 2.4.3, we can find an unsplittable flow of value equal to $\alpha^f(G', \mathcal{D}', s)$. The flow paths from this unsplittable solution are in turn feasible in $(G, \mathcal{D}, s)$, since the demands in $(G, \mathcal{D}, s)$

are at most a factor of y^{-1} greater, while the capacities are at least y^{-1} times greater. Thus, the value of this unsplittable flow is within a factor of $\frac{1}{2}y$ of $\alpha^f(G, \mathcal{D}, s)$. ■

Lemma 3.3.2 *There is a polynomial–time algorithm producing an unsplittable flow of value at least* $[2e\lceil\ln(\rho^*/\rho_*)\rceil]^{-1}$ *times* $\alpha^f(G, \mathcal{D}, s)$.

Proof. Write $q = \lceil\ln(\rho^*/\rho_*)\rceil$. We partition the terminals into q subsets $\mathcal{D}_1, \ldots, \mathcal{D}_q$, so that the demands of all terminals in $\mathcal{D}_i$ are in the interval $[\rho_* e^{i-1}, \rho_* e^i]$. In each of the problems $(G, \mathcal{D}_i, s)$, we can produce an unsplittable flow of value at least $(2e)^{-1}$ times $\alpha^f(G, \mathcal{D}_i, s)$, using Lemma 3.3.1. Since $\alpha^f(G, \mathcal{D}, s) \leq \sum_{i=1}^{q} \alpha^f(G, \mathcal{D}_i, s)$, we can obtain the claimed approximation ratio by taking, from among the q unsplittable flows that we compute, the one of the greatest value. ■

Theorem 3.3.3 *Let* $\rho^* < \frac{3}{2} - \sqrt{2}$. *There is a polynomial–time algorithm for the single–source* UFP *producing a flow of value at least* $1 - 4\sqrt{\rho^*} + 2\rho^*$ *times* $\alpha^f(G, \mathcal{D}, s)$.

Proof. We require $\rho^* < \frac{3}{2} - \sqrt{2}$ so that the expression $1 - 4\sqrt{\rho^*} + 2\rho^*$ is positive. Let $(G, \mathcal{D}, s)$ be an instance of the single–source UFP in which $\rho^* < \frac{3}{2} - \sqrt{2}$. We set $\sigma_0 = \frac{\sqrt{\rho^*}}{2-2\sqrt{\rho^*}}$; note that

$$\sigma_0 - \rho^* = \frac{\sqrt{\rho^*} - 2\rho^* + 2\rho^{*3/2}}{2 - 2\sqrt{\rho^*}} = \frac{(\rho^{*1/4} - \rho^{*3/4})^2 + \rho^{*3/2}}{2 - 2\sqrt{\rho^*}} > 0.$$

Let Γ be a strong coalition in $(G, \mathcal{D}, s)$ of order $\sigma_0 - \rho^*$, let $T_1, \ldots, T_r$ denote the trees in Γ, and for $i = 1, \ldots, r$ let $\mathcal{D}_i$ denote the set of terminals assigned to the tree T_i.

Let G' denote the graph obtained by multiplying all capacities of G by $1 + \sigma_0$ and then rounding up to the nearest multiple of σ_0. We assign each non–exceptional vertex in $U(\Gamma)$ a demand of σ_0, and each exceptional vertex in $U(\Gamma)$ a demand equal to the amount assigned to its tree. (Because of this latter point, we are in fact placing demand on s itself; this is simply to be able to make a comparison to the optimum, below.) We then compute a maximum unsplittable flow in $(G', U(\Gamma), s)$. Since the only demands not equal to σ_0 are situated at the source s, Corollary 2.4.3 implies that this maximum unsplittable flow is equal to the fractional optimum in $(G', U(\Gamma), s)$ and can be computed in polynomial time.

We claim that the value of this unsplittable flow is at least as large as the value of the maximum fractional flow in $(G, \mathcal{D}, s)$. For consider any fractional flow in $(G, \mathcal{D}, s)$, and let ε_i

denote the demand contained in tree T_i of Γ that was routed by this flow. Then by Lemma 3.1.4, there is a feasible fractional flow in $(G', U(\Gamma), s)$ that routes ε_i demand from $u_i \in U(\Gamma)$. But the unsplittable flow that we compute in $(G', U(\Gamma), s)$ has value equal to that of the *maximum* fractional flow, proving the claim.

Let U' denote the set of non–exceptional vertices of $U(\Gamma)$ that are routed by the maximum unsplittable flow constructed in the previous paragraph, and I' the set of indices of such vertices in $\{1, \ldots, r\}$. For $i \in I'$, we define $\mathcal{D}'_i \subset \mathcal{D}_i$ to be any set which is maximal subject to the condition that

$$\rho(\mathcal{D}'_i) \leq \left(\frac{1 - 2\sigma_0}{1 + 2\sigma_0}\right) \sigma_0. \tag{3.5}$$

Note that by the maximality of $\mathcal{D}'_i$, we have

$$\rho(\mathcal{D}'_i) \geq \left(\frac{1 - 2\sigma_0}{1 + 2\sigma_0}\right) \sigma_0 - \rho^*. \tag{3.6}$$

Let $\mathcal{D}' = \cup_{i \in I'} \mathcal{D}'_i$. Finally, let I'' denote the set of indices of the exceptional trees, and $\mathcal{D}'' = \cup_{i \in I''} \mathcal{D}_i$.

We will show that $\mathcal{D}' \cup \mathcal{D}''$ is realizable in the graph G. We argued above that $\sigma_0 \cdot |I'| + \rho(\mathcal{D}'')$ is at least as large as the maximum fractional flow in $(G, \mathcal{D}, s)$; by Equation (3.6), the unsplittable flow that we produce has value at least

$$\left[\left(\frac{1 - 2\sigma_0}{1 + 2\sigma_0}\right) \sigma_0 - \rho^*\right] |I'| + \rho(\mathcal{D}'').$$

This value is thus within a factor of at least

$$\frac{\left(\frac{1-2\sigma_0}{1+2\sigma_0}\right) \sigma_0 - \rho^*}{\sigma_0} = 1 - \frac{\frac{2\sqrt{\rho^*}}{1-\sqrt{\rho^*}}}{\frac{1}{1-\sqrt{\rho^*}}} - \frac{2\rho^*}{\frac{\sqrt{\rho^*}}{1-\sqrt{\rho^*}}} = 1 - 4\sqrt{\rho^*} + 2\rho^*. \tag{3.7}$$

times that of the fractional optimum.

Thus it remains only to show that $\mathcal{D}' \cup \mathcal{D}''$ is realizable in G. For $u_i \in U'$, let Q_i denote the u_i-s flow path in the maximum unsplittable flow we constructed in $(G', U(\Gamma), s)$. Now, consider any $t_j \in \mathcal{D}' \cup \mathcal{D}''$, with $\tau(t_j) = T_i$. As in the proof of Theorem 3.2.1, we define R_j to be the unique t_j-u_i path in T_i. We then define P_j to be the concatenation of Q_i and R_j. P_j will serve as the flow path for the terminal t_j.

We claim that this set $\{P_j\}$ of flow paths does not violate any capacity constraint. To show this, we consider any edge e. We know that if each flow path Q_i were to carry σ_0 units of flow, then at most $(1+2\sigma_0)c_e$ units of flow would pass across e due to the Q-parts of all paths in $\{P_j\}$. But now, each flow path Q_i carries at most

$$\left(\frac{1-2\sigma_0}{1+2\sigma_0}\right)\sigma_0$$

units of flow, and hence at most $(1-2\sigma_0)c_e$ units of flow pass across edge e due to the Q-parts of all paths in $\{P_j\}$. Also, since e is contained in at most two trees of Γ, and $c_e \geq 1$, at most $2\sigma_0 \leq 2\sigma_0 c_e$ units of flow pass across e due to the R-parts of all paths in $\{P_j\}$. Hence, the capacity constraint for edge e is satisfed.

Thus the set $\mathcal{D}' \cup \mathcal{D}''$ is realizable in G, using the flow paths $\{P_j\}$. The theorem follows. ■

From this result, we can obtain a constant–factor approximation without any assumptions about the size of the largest demand (aside from Equation 1.2). Let $\mathcal{D}_0$ denote the set of terminals of demand less than e^{-3}, and $\mathcal{D}_1$ the set of terminals of demand at least e^{-3}. We can obtain a constant–factor approximation algorithm for $\alpha(G, \mathcal{D}_1, s)$ using Lemma 3.3.2, and a constant–factor approximation algorithm for $\alpha(G, \mathcal{D}_0, s)$ using Theorem 3.3.3. Of the two unsplittable flows produced this way, taking the one of greater value gives us

Corollary 3.3.4 *There is a constant–factor approximation algorithm for* $\alpha(G, \mathcal{D}, s)$.

3.4 Routing in Rounds

In order to obtain a constant–factor approximation for the problem of routing in rounds, we make use of a classical connection between flows and matroids [87, 119]. In the following section we discuss this connection and its consequences; we then present the approximation algorithm.

3.4.1 Tools from Matroid Theory

A *matroid* M is a pair $(U, \mathcal{M})$, where U is a finite ground set, and $\mathcal{M}$ is a collection of subsets of U that satisfies that following two conditions.

(i) *(Closure with respect to inclusion.)* If $Y \in \mathcal{M}$ and $X \subset Y$, then $X \in \mathcal{M}$.

(ii) *(The exchange property.)* If $X, Y \in \mathcal{M}$ and $|X| < |Y|$, then there exists an element $y \in Y$ such that $X \cup \{y\} \in \mathcal{M}$.

A subset of U is called *independent* if it belongs to $\mathcal{M}$; it is called a *basis* if it is a maximal independent set, with respect to inclusion. Note that the exchange property ensures that all bases have the same cardinality. If $X \subset U$, we define the *rank* of X, denoted $r(X)$, to be the maximum cardinality of an independent subset of X. We refer the reader to a book such as [119] for further background on matroids.

There are a number of basic problems that are computationally intractable for set systems in general, but which are tractable when the set system is a matroid. We will be primarily interested in the following *covering problem*: for a matroid $M = (U, \mathcal{M})$, what is the minimum cardinality cover of U by members of $\mathcal{M}$? This question is addressed by the following result of Edmonds [31].

Theorem 3.4.1 (Edmonds) *There is a polynomial–time algorithm to find a minimum cardinality cover of the ground set of a matroid $M = (U, \mathcal{M})$ by independent sets. Moreover, if this minimum is greater than j, then there is some $X \subset U$ for which $j \cdot r(X) < |X|$.*

We say that a single–source UFP is *integral* if all demands are equal to some common value ρ, and all capacities are multiples of ρ. Recall that by Corollary 2.4.3, such integral unsplittable problems are well–characterized by the max–flow min–cut theorem. The relevance of matroids to such problems is a consequence of the following result, due to Perfect [87].

Theorem 3.4.2 (Perfect) *Let $(G, \mathcal{D}, s)$ be an integral single–source UFP. The collection of realizable subsets of $\mathcal{D}$ forms a matroid over the ground set $\mathcal{D}$.*

The following fact is immediate.

Lemma 3.4.3 $\nu_0^f(G, \mathcal{D}, s) \leq \nu_0(G, \mathcal{D}, s) \leq \chi(G, \mathcal{D}, s)$.

We now show that when $(G, \mathcal{D}, s)$ is integral, the matroid structure of the problem implies that the minimum fractional relative congestion, rounded up to the nearest integer, is in fact equal to the minimum number of rounds required. This is far from the case for general unsplittable flow problems.

Lemma 3.4.4 *If $(G, \mathcal{D}, s)$ is integral, then $\lceil \nu_0^f(G, \mathcal{D}, s) \rceil = \chi(G, \mathcal{D}, s)$ (and hence the latter quantity can be computed in polynomial time).*

Proof. In view of the previous lemma, it is enough to show that $\chi(G, \mathcal{D}, s) \leq \lceil \nu_0^f(G, \mathcal{D}, s) \rceil$. Let us denote the former quantity by j. If $j = 1$, then we are done, so let us assume $j > 1$. By Theorem 3.4.2, we can view j as the minimum cardinality of a cover of $\mathcal{D}$ by members of the matroid of realizable subsets. Thus by Theorem 3.4.1, there is a set X such that $\rho(X)$ is more than $j - 1$ times as large as the maximum total demand of a realizable set of terminals in X. Thus, by Corollary 2.4.3, we have $c(X) < \frac{1}{j-1}\rho(X)$. It follows that $\nu_0^f(G, \mathcal{D}, s) > j - 1$, and so $\lceil \nu_0^f(G, \mathcal{D}, s) \rceil \geq j$. ■

3.4.2 The Approximation Algorithm

We are given a single–source problem $(G, \mathcal{D}, s)$; we assume by re-scaling that the minimum edge capacity is 1, and all demands are between 0 and 1. Our approximation algorithm will be similar to that of the previous section; we choose a threshold value, and treat demands larger than the threshold separately from those smaller than the threshold. As above, it is straightforward to handle the large demands (Lemmas 3.4.5 and 3.4.6).

Lemma 3.4.5 *There is a polynomial–time algorithm that routes $(G, \mathcal{D}, s)$ in at most $\lceil 2\frac{\rho^*}{\rho_*}\nu_0^f(G, \mathcal{D}, s) \rceil$ rounds.*

Proof. Let G' denote the graph obtained from G by rounding all capacities down to the nearest multiple of ρ^*, and $\mathcal{D}'$ denote the set of terminals obtained by assigning each terminal in $\mathcal{D}$ a demand of ρ^*. The problem $(G', \mathcal{D}', s)$ is integral; we compute a routing for this problem in $\chi(G', \mathcal{D}', s) = \lceil \nu_0^f(G', \mathcal{D}', s) \rceil$ rounds, and use the associated paths for the problem $(G, \mathcal{D}, s)$. Clearly the paths used in each round respect the capacity constraints in $(G, \mathcal{D}, s)$.

By an argument strictly analogous to the proof of Lemma 3.3.1, we have $\nu_0^f(G', \mathcal{D}', s) \leq 2\frac{\rho^*}{\rho_*}\nu_0^f(G, \mathcal{D}, s)$, from which the result follows. ■

Similarly, by an argument strictly analogous to the proof of Lemma 3.3.2, we have

Lemma 3.4.6 *There is a polynomial–time algorithm that routes $(G, \mathcal{D}, s)$ in at most*

$$\lceil \ln(\rho^*/\rho_*) \rceil \cdot \lceil 2e\nu_0^f(G, \mathcal{D}, s) \rceil$$

rounds.

Theorem 3.4.7 *Let $\rho^* < \frac{7}{4} - \sqrt{3}$. There is a polynomial–time algorithm that routes $(G, \mathcal{D}, s)$ in $2\lceil \nu_0^f(G, \mathcal{D}, s)\rceil$ rounds.*

Proof. Let $(G, \mathcal{D}, s)$ be an instance of the single–source UFP in which $\rho^* < \frac{7}{4} - \sqrt{3}$. We set $\sigma_0 = \frac{\sqrt{\rho^*}}{2-2\sqrt{\rho^*}}$ and let Γ be a strong coalition in $(G, \mathcal{D}, s)$ of order $\sigma_0 - \rho^*$. Let $T_1, \ldots, T_r$ denote the trees in Γ, and for $i = 1, \ldots, r$ let $\mathcal{D}_i$ denote the set of terminals assigned to the tree T_i.

Let G' denote the graph obtained by multiplying all capacities of G by $1 + \sigma_0$ and then rounding up to the nearest multiple of σ_0. We assign each non–exceptional vertex in $U(\Gamma)$ a demand of σ_0, and each exceptional vertex in $U(\Gamma)$ a demand of 0. We claim that there is a fractional routing of $(G', U(\Gamma), s)$ with relative congestion at most $\lceil \nu_0^f(G, \mathcal{D}, s)\rceil$. To see this, note that if we were to multiply all capacities of G by a factor of $\lceil \nu_0^f(G, \mathcal{D}, s)\rceil$, then there would be a feasible fractional routing of $\mathcal{D}$, *and* all capacities would be at least 1; thus, by Lemma 3.1.4, there would be a fractional routing of $U(\Gamma)$ of relative congestion at most $1+\sigma_0$. But the capacities in G' are at least $1 + \sigma_0$ times what they are in G, and hence if we were to multiply all capacities of G' by a factor of $\lceil \nu_0^f(G, \mathcal{D}, s)\rceil$, then $U(\Gamma)$ would be fractionally feasible. It follows that $\nu_0^f(G', U(\Gamma), s) \leq \lceil \nu_0^f(G, \mathcal{D}, s)\rceil$.

The algorithm is now as follows. We compute a routing of $(G', U(\Gamma), s)$ in $\lceil \nu_0^f(G', U(\Gamma), s)\rceil \leq \lceil \nu_0^f(G, \mathcal{D}, s)\rceil$ rounds. Now, we break each of these rounds into two rounds: if u_i is routed in a given round j, then we route the terminals in its tree in the pair of rounds associated with j. This will result in the claimed approximation ratio. It therefore remains to show how to take a single round of our routing of $(G', U(\Gamma), s)$, and route the associated terminals in two rounds in $(G, \mathcal{D}, s)$.

Thus, let U' denote the set of vertices in $U(\Gamma)$ routed in a single round of the routing for $(G', U(\Gamma), s)$, and I' the set of indices of such vertices. Let $\mathcal{D}_i$ denote the of terminals associated with a vertex $u_i \in U$. Since the total demand of terminals in $\mathcal{D}_i$ is at most σ_0, and the maximum demand is ρ^*, we can partition $\mathcal{D}_i$ into two sets, $\mathcal{D}_i^1$ and $\mathcal{D}_i^2$ so that the demand in each is at most $\frac{1}{2}\sigma_0 + \rho^*$. Recall from Equation (3.7) that

$$\frac{\left(\frac{1-2\sigma_0}{1+2\sigma_0}\right)\sigma_0 - \rho^*}{\sigma_0} = 1 - 4\sqrt{\rho^*} + 2\rho^*. \tag{3.8}$$

Now since $\rho^* < \frac{7}{4} - \sqrt{3}$, we have $1 - 4\sqrt{\rho^*} + 2\rho^* < \frac{1}{2}$, and hence by Equation (3.8) we have

$$\rho(\mathcal{D}_i^1), \rho(\mathcal{D}_i^2) \leq \frac{1}{2}\sigma_0 + \rho^* \leq \left(\frac{1-2\sigma_0}{1+2\sigma_0}\right)\sigma_0.$$

We claim that for $\ell = 1, 2$, the set of terminals in $\mathcal{D}_i^\ell$ $(i \in I')$ can be routed in a single round. To show this, we define the following flow path P_j for $t_j \in \mathcal{D}_i^\ell$: we concatenate the t_j-u_i path in T_i, denoted R_j, with the u_i-s flow path, denoted Q_i. Now for any edge e, the total amount of flow crossing e due to Q-parts of paths is at most

$$(1+2\sigma_0)c_e \cdot \left(\frac{1-2\sigma_0}{1+2\sigma_0}\right) = (1-2\sigma_0)c_e,$$

and the total amount of flow crossing e due to R-parts of paths is at most $2\sigma_0 \leq 2\sigma_0 c_e$. Thus the capacity constraints are satisfied for the set of paths $\{P_j\}$ associated with terminals in $\cup_{i\in I'}\mathcal{D}_i^\ell$.

Finally, since terminals in exceptional trees are routed entirely on paths of the form R_j, they can all be routed in one of the rounds already used, without violating the capacity constraints.

■

The above facts provide us with the main approximation result of this section — namely, an approximation algorithm for $\chi(G, \mathcal{D}, s)$ when ρ_* and ρ^* are arbitrary numbers in $(0, 1]$. The approximation is obtained simply by using separate sets of rounds for the terminals of demand at least $\frac{7}{4} - \sqrt{3}$ and for those of demand less than $\frac{7}{4} - \sqrt{3}$. As we are within a constant factor of optimal for each type of demand, we have

Corollary 3.4.8 *Let $(G, \mathcal{D}, s)$ be an arbitrary single–source unsplittable flow problem. There is a polynomial–time constant–factor approximation for $\chi(G, \mathcal{D}, s)$.*

It is also worth noting another natural corollary of the above results.

Corollary 3.4.9 *If $(G, \mathcal{D}, s)$ is fractionally feasible, then it is routable without splitting in a constant number of rounds. If additionally $\rho^* < \frac{7}{4} - \sqrt{3}$, then it is routable without splitting in two rounds.*

Chapter 4

Directed Single–Source Problems

... With windlasses and with assays of bias,
By indirections find directions out.
— *William Shakespeare,* Hamlet.

We now extend the approximation algorithms of the previous chapter to the case of directed graphs. That is, for a vertex s in G, terminal set $\mathcal{D} = \{t_1, \ldots, t_k\}$, and associated demands $\rho_1, \ldots, \rho_k$, we are seeking directed paths *from* s *to* the terminals in $\mathcal{D}$ such that the capacity constraints are met. We again are able to obtain constant approximation ratios that converge to 1 as the ratio of the maximum demand to the minimum capacity goes to 0. The underlying technique here is still based on the construction of coalitions, now using directed *arborescences* in place of trees. However, one must be more careful in defining the "right" coalition in the directed case; we will discuss this below.

This is the only chapter in which we deal with directed graphs. Our motivation in undertaking this digression into the directed case has several sources. First, as noted in Section 2.9, the disjoint paths problem in directed graphs is much less well understood than it is in the undirected case, and so it is of particular interest to develop approximation techniques that are applicable to directed graphs.

Second, we observe that much of the appeal of the max-flow min-cut theorem has nothing to do with issue of routing in networks, but rather lies in the ease with which one can reduce a wide range of problems in combinatorial optimization to it. It turns out that this notion of reduction is applicable in the setting of single–source unsplittable flow as well; but most of the natural reductions here require that the underlying graph be directed.

As a basic example, consider the following machine scheduling problem. We have a set $J = \{j_1, \ldots, j_p\}$ of *jobs*, such that job j_i has *weight* w_i, and a set $M = \{m_1, \ldots, m_q\}$ of *machines*. Associated with job j_i is a set $S_i \subset M$ of machines on which it can run. The goal is to assign each job to a machine in such a way that the maximum weight assigned to any machine is minimized. In the case in which all S_i are equal to M, this is just the classical *minimum makespan problem*, for which a $(1 + \varepsilon)$–approximation can be found in polynomial time for any fixed $\varepsilon > 0$ [50]. But when the S_i are arbitrary sets, Lenstra, Shmoys, and Tardos show that no polynomial–time algorithm can approximate the optimum load to within a factor better than $\frac{4}{3}$ [72]; and they provide a 2-approximation algorithm for a class of problems that includes this one.

Here, we observe that this load balancing problem can be easily formulated as a single–source unsplittable flow problem on a three–level directed graph, as in Figure 4-1. We introduce a node for every $m_j \in M$ and $j_i \in J$, and we include a directed edge (m_j, j_i) iff $m_j \in S_i$. We then attach a source s by an edge to each $m_j \in M$, and give each j_i a demand equal to w_i. Then the minimum–congestion routing of an unsplittable flow in this graph corresponds to a minimum–load assignment of jobs to machines. (We note that the constant factor of approximation we obtain for the general single–source problem is greater than 2, so we are not claiming to improve on the bound of Lenstra, Shmoys, and Tardos for this special case.)

Of course, much more elaborate reductions are possible. As another example, one could consider a *multi-stage load balancing* problem in which there is a set J of jobs, and sets $S_1, \ldots, S_k$ of *sites*, residing in a metric space M. Each job must be processed by one site from each set S_i, in sequence; but in moving from one site to the next, a job cannot move more than some distance ξ in the metric space M. The goal is to find a way of processing each job, so that the maximum load on any site is minimized. Again, it is straightforward to formulate this as the problem of finding the minimum congestion of a single–source unsplittable flow in an appropriate directed graph.

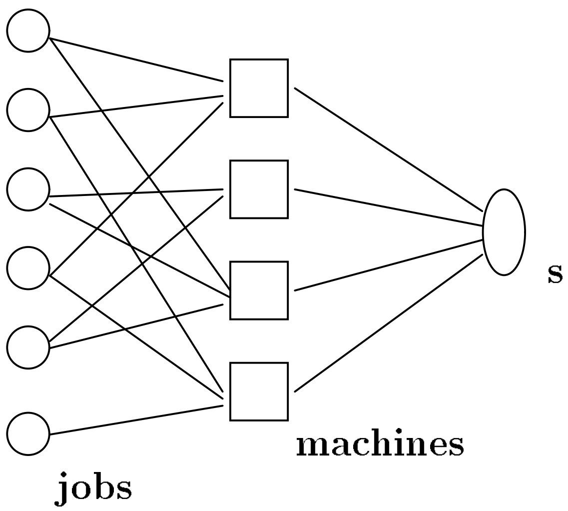

Figure 4-1: A load balancing problem

Definitions and Techniques

We will work here with graphs whose capacity values are integers; by allowing parallel edges, we can assume that every capacity is equal to 1.

As in the undirected case, we make use of the notion of *coalitions* as a way of "grouping" demand in G. However, since G is now a directed graph, we must make sure that the leader of each tree in the coalition has a directed path to each of its terminals — in this way, the demand that has been grouped into this leader node can still reach the terminals in which it resides. Thus, we first define the types of directed trees that we will be dealing with, and then discuss some of the additional complications that arise when using directed coalitions.

Let G be a directed graph, with a distinguished source vertex s. Unless we say otherwise, *paths* and *cycles* in G will be directed. As before, we are interested in finding flow paths from s to the terminals $\mathcal{D} = \{t_1, \ldots, t_k\}$. Thus we will assume that there is a path from s to every other node of G; this is no loss of generality, since no other nodes can participate in a solution to the unsplittable flow problem defined by $(G, \mathcal{D}, s)$. By a *tree* in G, we mean a subgraph T of G whose underlying edge set forms a tree if one ignores the directions of the edges. If T is a tree in G, and v is a vertex in T with a path to every other vertex in T, then we say that v is a *root* of T. A tree can have at most one root; if a tree T has a root, then we will say that

T is an *arborescence*. We say that T is a *spanning arborescence* if it is an arborescence whose vertex set is equal to V. Finally, a *forest* is a set of trees in G that do not share vertices.

Given these definitions, a directed coalition will simply be a set of half–disjoint arborescences, with each terminal assigned to an arborescence that contains it. Strong directed coalitions can be constructed in a straightforward fashion from a depth-first search of any spanning arborescence of G rooted at s (Lemma 4.1.3).

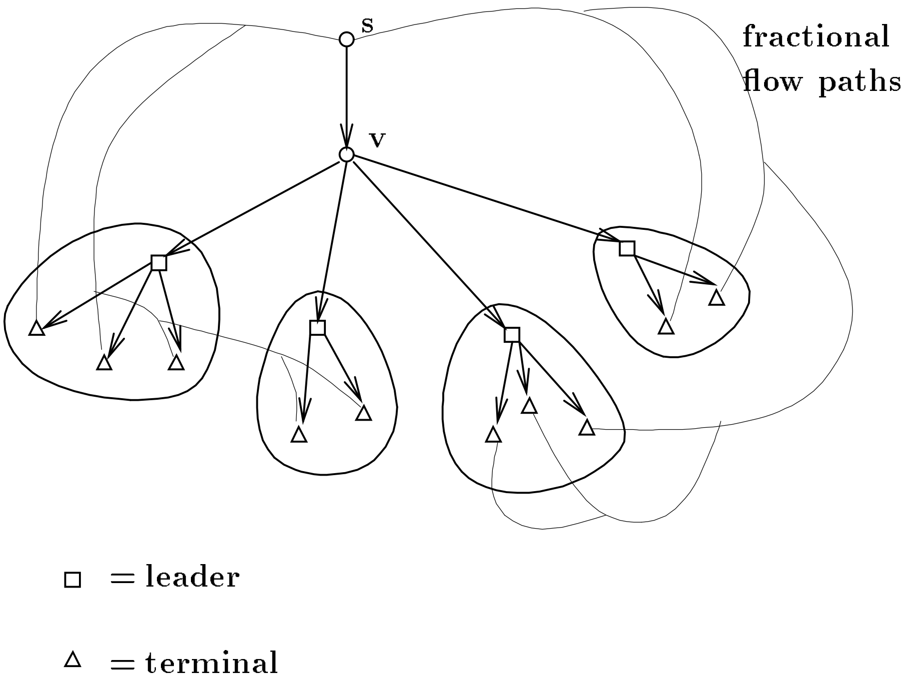

Figure 4-2: A bad choice of coalition

The main technical complication arises from the fact that the obvious analogue of Lemma 3.1.4 does not hold in the directed case — given a fractionally realizable set of terminals, it is possible to construct a coalition so that the resulting set of leaders can only be routed with a very large congestion. Such an example is pictured in Figure 4-2: although there are flow paths to all the terminals, the coalition is built out of edges not participating in this flow, and all the paths from s to the leaders share a single edge (s, v); this results in very high congestion. Note that this difficulty cannot arise in the undirected case, where a set of low–congestion paths to the leaders could be built by taking the fractional flow paths to the terminals in the reverse direction; this is precisely what is taking place in the proof of Lemma 3.1.4.

What we will show, in Lemma 4.1.7, is that one can always choose a "good" coalition, so that the leaders can be routed with approximately the same congestion as the terminal set. Given this lemma, the approximation algorithms will follow almost exactly as in the undirected case.

The construction of a good coalition proceeds roughly as follows. By the max-flow min-cut theorem for directed graphs (Corollary 2.9.1), we have to avoid the following situation: there is a set $S \not\ni s$ containing a large number of leaders, such that very few edges enter S. Thus, given a set S containing a large number of leaders (but not s), we consider the following two cases:

(i) S also contains a correspondingly large number of terminals. In this case, since the terminal set is fractionally routable, there must be many edges entering S.

(ii) S contains very few terminals. Since there are paths from the leaders to their associated terminals, and very few of these terminals lie in S, it must be that there are many edges *leaving* S. In this case, we examine the set of edges leaving S. We say that an edge is *dense* if it carries at least $\frac{1}{2}$ units of flow from s to the terminal set, and *shallow* otherwise. We arrange the construction of our coalition $\vec{\Gamma}$ so that if a node has any entering edge that is dense, such an edge enters it in the arborescences of $\vec{\Gamma}$. We again have two cases to consider.

  (a) If a large fraction of the edges leaving S are dense, then a large amount of flow must be leaving S. It follows that a large amount of flow also enters S (since $s \notin S$), and hence there are many edges entering S.

  (b) Otherwise, suppose that most of the edges leaving S are shallow. This is the most difficult case, since we cannot argue that a lot of flow enters S. However, if we grow the set S by following chains of vertices of in–degree 1, and it remains the case that most of the edges leaving S are shallow, then by the construction of the coalition $\vec{\Gamma}$, we can argue that S now has out–edges to a large number of *shallow branching nodes* — nodes of in–degree greater than 1, all of whose entering edges are shallow. Now, the crucial fact that we establish in Lemma 4.1.7, using Lemmas 4.1.4 and 4.1.6, is that there are edge–disjoint paths linking s to *all* the shallow branching nodes in G. Consequently, we will be able to argue that when most of the edges leaving S are shallow, S must be entered by many of these edge–disjoint paths, and consequently there are many edges entering S, as required.

4.1 Coalitions

We first define coalitions in the directed setting.

Definition 4.1.1 *A* (weak) coalition *in* $(G, \mathcal{D}, s)$ *is a triple* $(\{A_1, \ldots, A_r\}, \{u_1, \ldots, u_r\}, \tau)$, *where each* A_i *is an arborescence in* G *with root* u_i, $\tau : \mathcal{D} \to \{A_1, \ldots, A_r\}$ *is a function, and the following properties are satisfed.*

(i) The arborescences are half–disjoint, *by which we mean that each edge of* G *belongs to at most two of the* A_i.

(ii) For each $i = 1, \ldots, k$, $t_i \in \tau(t_i)$.

Definition 4.1.2 *A* strong coalition *in* $(G, \mathcal{D}, s)$ *of order* σ *is a triple* $(\{A_1, \ldots, A_r\}, \{u_1, \ldots, u_r\}, \tau)$, *such that the following properties are satisfed.*

(i) $(\{A_1, \ldots, A_r\}, \tau)$ *is a coalition in* G.

(ii) For each $j = 1, \ldots, r$, *either* $u_j = s$ *or we have*

$$\sigma \leq \sum_{i : \tau(t_i) = A_j} \rho_i \leq \sigma + \rho^*. \tag{4.1}$$

As before, an arborescence for which inequality (4.1) does not hold will be called *exceptional.* For $i = 1, \ldots, r$, we say that u_i is the *leader* of A_i.

We show that strong coalitions exist in directed graphs. Our construction must be somewhat different from that of Section 3.1, since we need to produce arborescences rather than just trees. Thus, we proceed as follows. If A is an arborescence, and X is a set of vertices of A, consider the collection of subgraphs of A with property that they are arborescences containing X. It is easy to verify that this collection contains a unique minimal arborescence, which we will denote by $A[X]$.

Lemma 4.1.3 *Let A be an spanning arborescence in G with root s, and let $\sigma \geq 0$. Then there exists a strong coalition $\vec{\Gamma}$ in $(G, \mathcal{D}, s)$ of order σ, such that the edge sets of all arborescences in $\vec{\Gamma}$ are subsets of A.*

Proof. We perform a depth–first search of A, starting from the root s, and let $t_{\pi(1)}, \ldots, t_{\pi(k)}$ denote the order in which the terminal are first encountered in this depth–first search. As in the proof of Lemma 3.1.3, we partition this sequence into intervals $\mathcal{D}_1, \ldots, \mathcal{D}_r$, such that the total demand in each interval except possibly the last is between σ and $\sigma + \rho^*$. We add s to the set $\mathcal{D}_r$. We now construct the coalition $\vec{\Gamma}$ by defining, for $i = 1, \ldots, r$, $A_i = A[\mathcal{D}_i]$. We define u_i to be the root of the i^{th} arborescence, and the function τ so that $\tau(t_j) = A_i$ if $t_j \in \mathcal{D}_i$.

We show that the arborescences $\{A_i\}$ are half–disjoint; the other properties of Definitions 4.1.1 and 4.1.2 are straightforward to verify. Let C_i denote the set of edges traversed between the first visit to a vertex in $\mathcal{D}_i$ and the last visit to a vertex in $\mathcal{D}_i$. Now the sets $C_1, \ldots, C_r$ are half–disjoint, since every edge of A is traversed exactly twice in a depth–first search. But C_i contains the edge set of an arborescence on $\mathcal{D}_i$, and thus the edges of $A[\mathcal{D}_i]$ belong to C_i. It follows that the $\{A_i\}$ are half–disjoint. ■

Given any spanning arborescence A rooted at s, and any $\sigma \geq 0$, this lemma gives us a canonical strong coalition of order σ induced by A. Let us use $\vec{\Gamma}(A, \sigma)$ to denote this coalition. Now, in the introduction to this chapter, we indicated that the natural analogue of Lemma 3.1.4 does *not* hold in the directed case, and that we must use information about the fractional flow from s to the terminal set $\mathcal{D}$ in our construction of a directed coalition. In particular, we will need to prove the existence of disjoint paths from s to the *shallow branching nodes* of G, and for this we need Lemmas 4.1.4 and 4.1.6 below.

For a vertex v of a directed graph, let δ_v^+ denote the number of edges for which it is the tail, and δ_v^- denote the number of edges for which it is the head. Let $\delta_v = \delta_v^+ + \delta_v^-$. We say that v is a *leaf* if $\delta_v = 1$; if v is a leaf, we call it an *in-leaf* if $\delta_v^- = 1$, and an *out-leaf* if $\delta_v^+ = 1$. Finally, we say that T is a *reverse arborescence* with root u if reversing the direction of every edge turns it into an arborescence with root u.

Lemma 4.1.4 *Let H be a directed acyclic graph, and u a node of H, such that every node $v \in H \setminus \{u\}$ satisfies $\delta_v^- \geq \delta_v^+$. Then there exists a set of edge disjoint paths in H, such that* *every path has its tail in u, and for every $v \in H \setminus \{u\}$, at least $\delta_v^- - \delta_v^+$ paths have their heads in w.*

Proof. We add a new node u' to H, and for every $v \in H \setminus \{u\}$, we add $\delta_v^- - \delta_v^+$ parallel edges from v to u'. We also add δ_u^+ parallel edges from u' to u. Denote the resulting graph by H'. Note that in H', the in–degree of every node is equal to its out–degree, and so H' is Eulerian. Let C be an Eulerian circuit in this graph. If we delete u' from H', then C decomposes into edge–disjoint paths. It is easy to verify that these paths satisfy the conditions of the lemma. ∎

Lemma 4.1.5 *Let A' be a reverse arborescence on at least two nodes, with root v, and let L be its set of out–leaves. Then there exists a collection of edge–disjoint paths in A' such that each path has its tail in a distinct vertex of L, at least δ_v^- paths have their heads in v, and for every $w \in A' \setminus (Y \cup \{v\})$, at least $\delta_w^- - 1$ of the paths have their heads in w.*

Proof. Attach a new node u to A', by a single edge to each node in L, and apply Lemma 4.1.4. ∎

Lemma 4.1.6 *Let F be a forest, with vertex set W, and let $B = \{w \in W : \delta_w^- = 0\}$. Then there exists a collection of edge–disjoint paths in T such that*

(i) each path has its tail in a distinct vertex of B, and

(ii) for every $x \in W \setminus B$, at least $\delta_x^- - 1$ paths have their heads in x.

Proof. It is enough to prove this for a tree T; one can then apply the lemma in each component of F separately. Assuming T has more than one node, we fix an arbitrary node $u \notin B$. We root the tree at u; by this we mean that w will be defined to be *below* v if either (v, w) or (w, v) is an edge of T, and the undirected distance from u to v is less than the undirected distance from u to w. Say that an edge is *proper* if its tail is below its head, and *improper* otherwise. Let $T_1, \ldots, T_q$ be the set of trees obtained by deleting all improper edges.

We now observe the following facts.

(i) Each T_i is a reverse arborescence, since any node $v \in T_i$ can be the tail of at most one edge in T_i.

(ii) A node is a out–leaf of some T_i if and only if it belongs to the set B.

(iii) The in-degree of a node $v \neq u$ in T_i is equal to $\delta_v^- - 1$ if it is the root of T_i, and equal to δ_v^- otherwise.

(iv) The in–degree of u in its tree T_i is equal to δ_u^-.

From these four facts, it follows that applying Lemma 4.1.5 to each of the T_i produces the desired paths in T. ■

We now use these lemmas to construct an appropriate coalition for $(G, \mathcal{D}, s)$.

Lemma 4.1.7 *Let $(G, \mathcal{D}, s)$ be a single–source fractional flow problem, in which terminal $t_j \in \mathcal{D}_j$ has an associated demand ρ_j. Let us assign to each terminal in $\mathcal{D}$ a demand $\tilde{\rho}_j \leq \rho_j$, in such a way that $(G, \mathcal{D}, s)$ has a feasible fractional routing. Then there exists a spanning arborescence A of G, rooted at s, such that for every $\sigma \geq 0$, the induced coalition $\vec{\Gamma} = \vec{\Gamma}(A, \sigma)$ has the following property. Suppose we define the problem $(G, U(\vec{\Gamma}), s)$ by assigning to each $u_i \in U(\vec{\Gamma})$ any demand ρ'_i not exceeding the total $\tilde{\rho}$–demand of terminals in its arborescence. If for all i we have $\rho'_i \leq \varepsilon$, then $(G, U(\vec{\Gamma}), s)$ has a fractional routing of congestion at most $1 + 6\varepsilon$.*

Proof. By assigning a demand $\tilde{\rho}_j \leq \rho_j$ to each $t_j \in \mathcal{D}$, the resulting problem $(G, \mathcal{D}, s)$ has a fractional routing that respects the capacity constraints. Let f_e denote the total amount of flow crossing edge e in this fractional routing. We call an edge *tight* if $f_e = 1$, and *empty* if $f_e = 0$. We may delete any empty edges from G, as this does not affect the feasibility of $(G, \mathcal{D}, s)$ with demand $\tilde{\rho}$.

We first perform the following reduction steps on the graph G and the associated fractional flow.

(i) We can assume that the set of non–empty edges does not induce any directed cycles, and hence (deleting empty edges if necessary) that the graph G is acyclic and $\delta_s^- = 0$.

(ii) Suppose there exists a set of edges C that induce a simple cycle if one ignores their directions. Let $v_0, \ldots, v_{q-1}$ denote the set of nodes in this cycle, with each $e \in C$ having ends $v_i, v_{(i+1 \bmod q)}$ for some i. We say that the edge $(v_i, v_{(i+1 \bmod q)}) \in C$ is *forward* if its tail is v_i, and *backward* otherwise. We uniformly increase the flow values on forward edges in C while uniformly decreasing the flow values on backward edges, until some edge in C becomes tight or empty.

(iii) If there are two tight edges (u, v) and (v, w), then we may delete both from G and add the tight edge (u, w). If we find the desired coalition in the resulting graph, then it is easy to produce a coalition in the graph G.

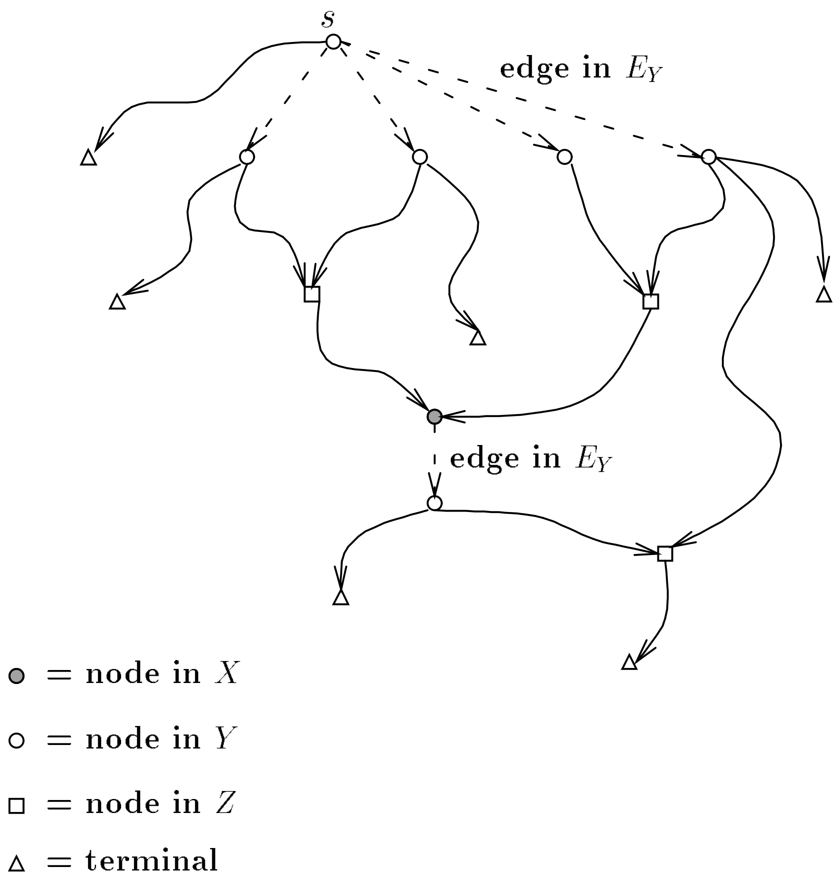

Figure 4-3: Constructing a good coalition

Following these reduction steps, we may assume that G is acyclic, $\delta_s^- = 0$, the set of non-tight edges induce a forest F, and no node is both the head and the tail of a tight edge. Given the last point, let X denote the set of vertices, other than s, that are tails of tight edges. Let Y denote the set containing s and all vertices that are heads of tight edges. Note that X and Y are disjoint sets.

We say that an edge e of G is *dense* if $f_e \geq \frac{1}{2}$, and *shallow* otherwise. We say that a node v is *branching* if $\delta_v^- > 1$, and *shallow* if all edges for which it is the head are shallow. Let Z

denote the set of all shallow, branching nodes.

For $u \in X$, let α_u denote the number of tight edges for which u is the tail. Thus at least α_u units of flow leave u; by flow conservation, at least α_u must enter u. Since this flow must enter on non–tight edges, the in–degree of u is at least $\alpha_u + 1$. In particular, u is branching. Moreover, if $u \in X$ is shallow (and hence in $Z \cap X$), then it must have in–degree at least $2\alpha_u + 1 \geq \alpha_u + 2$.

See Figure 4-3 for an example of all these definitions. To prove Lemma 4.1.7, we first show

(A) Z can be linked to s by edge–disjoint paths.

We prove (A) as follows. Let F denote the forest consisting of the non–tight edges and all nodes incident to them. We apply Lemma 4.1.6 to the forest F, obtaining a set of edge–disjoint paths $\mathcal{P}$. Any node v whose in–degree in F is zero must, by flow conservation, either be the head of a tight edge or be equal to s. Thus such a node belongs to Y, and so by Lemma 4.1.6, each of the paths in $\mathcal{P}$ has its tail in a distinct vertex of Y. Any node $u \in X \cup Z$ is not the head of any tight edge, and hence has the same in–degree in both G and F; thus u is the head of at least $\delta_u^- - 1$ paths in $\mathcal{P}$.

For each node $y \in Y \setminus \{s\}$, we now choose a single tight edge whose head is y, obtaining a collection of tight edges E_Y. Let E' denote the union of E_Y with the edges of $\mathcal{P}$, and let G' denote the subgraph of G consisting of the edges of E' and all nodes incident to them. For a node v of G', let $\delta_{v,G'}^-$ and $\delta_{v,G'}^+$ denote its in–degree and out–degree in the subgraph G'.

We claim that G', with source s, satisfies the hypotheses of Lemma 4.1.4. To see this, let v be any node of $G' \setminus \{s\}$; we show that $\delta_{v,G'}^- \geq \delta_{v,G'}^+$.

(i) If $v \in Y$, then it is at the tail of at most one path in $\mathcal{P}$, and at the head of one edge in E_Y.

(ii) If $v \in X$, then it is at the tail of at most α_v edges in E_Y, and at the head of at least $\delta_v^- - 1 \geq \alpha_v$ paths in $\mathcal{P}$.

(iii) If $v \notin X \cup Y$, then any edge for which it is the tail corresponds to an internal node of some path in $\mathcal{P}$, and can therefore be paired with some edge for which it is the head.

Let $\mathcal{Q}$ denote the set of edge–disjoint paths, anchored in s, that is provided by Lemma 4.1.4. We claim that for every $z \in Z$, the in–degree of z in G' is strictly greater than its out–degree;

this will show that at least one path in $\mathcal{Q}$ has its head in z. To show this, we observe that if $z \in Z \setminus X$, then the edges of G' incident to it correspond to paths in $\mathcal{P}$ for which z is internal, and at least one path in $\mathcal{P}$ for which z is the head; thus $\delta^-_{z,G'} > \delta^+_{z,G'}$. And if $z \in Z \cap X$, then $\delta^+_z \geq \alpha_z + 2$, and so z is the head of at least $\alpha_z + 1$ paths in $\mathcal{P}$. But z is the tail of at most α_z edges in E_Y, and so again we have $\delta^-_{z,G'} > \delta^+_{z,G'}$.

Thus there exists a set of edge–disjoint paths $\mathcal{Q}' \subseteq \mathcal{Q}$, with their tails in s, such that for every $z \in Z$, some path in $\mathcal{Q}'$ has its head in z. This proves Claim (A).

In this set of edge–disjoint s–Z paths, let P_z denote the path with ends s and $z \in Z$. We can now define the arborescence A. Since G is an acyclic graph in which s has a path to every other vertex, it is enough, in defining an arborescence, to specify an in–edge for every node of $V \setminus \{s\}$. For A, we do this as follows. Let $v \in V \setminus \{s\}$.

(i) If v is not a branching node, then we choose the unique edge whose head is equal to v.

(ii) If v is at the head of a dense edge, then we choose one of these dense edges arbitrarily.

(iii) Otherwise, v is a shallow branching node. In this case, we choose the final edge of the path P_v, constructed in the proof of Claim (A).

We now prove the main statement of the lemma. Let ρ'_i denote the demand assigned to $u_i \in U(\vec{\Gamma})$ in the problem $(G, U(\vec{\Gamma}), s)$. By hypothesis,

$$\rho'_i \leq \sum_{j:\tau(t_j)=A_i} \tilde{\rho}_j \leq \varepsilon. \tag{4.2}$$

We claim that if every edge is given a capacity of $1 + 6\varepsilon$, then the problem $(G, U(\vec{\Gamma}), s)$ is fractionally feasible. In order to show this, we must verify the cut condition, by showing that $(1 + 6\varepsilon)|\delta^-(S)| \geq \rho'(S)$ for every set $S \not\ni s$.

Suppose that there is a set violating this cut condition; let us choose such a set S whose cardinality is maximal. Suppose that S contains the leaders $u_{i_1}, \ldots, u_{i_q}$. For each of the associated arborescences A_{i_j}, we say that A_{i_j} is *unsplit* if its node set is entirely contained in S, and *split* otherwise. We consider the following two cases.

(i) At least a $\frac{1}{1+6\varepsilon}$ fraction of the total ρ'-demand of S is associated with leaders of unsplit arborescences. Thus, by Equation (4.2), we have

$$\tilde{\rho}(S) \geq \frac{\rho'(S)}{1+6\varepsilon}.$$

By the feasibility of the problem $(G, \mathcal{D}, s)$ with demands $\tilde{\rho}$, we have $|\delta^-(S)| \geq \tilde{\rho}(S)$, and hence

$$(1+6\varepsilon)|\delta^-(S)| \geq (1+6\varepsilon)\tilde{\rho}(S) \geq \rho'(S).$$

(ii) At least a $\frac{6\varepsilon}{1+6\varepsilon}$ fraction of the total ρ'–demand of S is associated with leaders of split arborescences. Since the maximum demand contained in any single arborescence is at most ε, there are at least

$$\frac{6\varepsilon}{(1+6\varepsilon)} \cdot \frac{\rho'(S)}{\varepsilon} = \frac{6\rho'(S)}{1+6\varepsilon}$$

split arborescences in S. Since these arborescences are half–disjoint, it follows that

$$|A \cap \delta^+(S)| \geq \frac{3\rho'(S)}{1+6\varepsilon}.$$

We now have two sub–cases.

(a) At least a $\frac{2}{3}$ fraction of the edges in $A \cap \delta^+(S)$ are dense. Thus, a total flow value of at least $(1+6\varepsilon)^{-1}\rho'(S)$, in the problem $(G, \mathcal{D}, s)$, is leaving the set S, and consequently at least this much flow is entering the set. But since edges have unit capacity in $(G, \mathcal{D}, s)$, we must have $|\delta^-(S)| \geq (1+6\varepsilon)^{-1}\rho'(S)$. Thus,

$$(1+6\varepsilon)|\delta^-(S)| \geq \rho'(S).$$

(b) At least a $\frac{1}{3}$ fraction of the edges in $A \cap \delta^+(S)$ are shallow. By the definition of A, all these edges have their heads in shallow nodes, $z_1, \ldots, z_r$. Because we are considering a violated cut whose cardinality is maximal, it must be that each z_i is also a branching node — for otherwise, $S \cup \{z_i\}$ would be a violated cut of larger cardinality. Thus $\{z_1, \ldots, z_r\} \subset Z$, and hence each of $P_{z_1}, \ldots, P_{z_r}$ has an edge in $\delta^+(S)$. But from this it follows that each of $P_{z_1}, \ldots, P_{z_r}$ must also have an edge in

$\delta^-(S)$. Since

$$r \geq \frac{1}{3} \cdot \frac{3\rho'(S)}{1+6\varepsilon},$$

we have

$$(1+6\varepsilon)|\delta^-(S)| \geq \rho'(S).$$

Thus in all cases, we have $(1+6\varepsilon)|\delta^-(S)| \geq \rho'(S)$, contradicting our assumption that S was a set violating the cut condition. Hence there are no violated cuts in $(G, U(\vec{\Gamma}), s)$, and so it is fractionally feasible with edge capacities equal to $1+6\varepsilon$. ■

4.2 The Approximation Algorithms

Using Lemma 4.1.7 in place of Lemmas 3.1.3 and 3.1.4, we will be able to obtain approximation algorithms for the three basic objective functions of the previous chapter; the constants will be somewhat larger due to the weaker bound provided by Lemma 4.1.7.

The only essential difference between Lemmas 3.1.4 and 4.1.7, in terms of their application, is the following. The coalition produced by Lemma 4.1.7 depends on the structure of $(G, \mathcal{D}, s)$, and in particular on the demand associated with each terminal in $\mathcal{D}$. This issue did not arise in the undirected case, since Lemma 3.1.4 applied to any coalition.

Congestion

Theorem 4.2.1 *Let G be directed and $(G, \mathcal{D}, s)$ be fractionally feasible. Then it has an unsplittable routing of relative congestion at most* $1 + 9\sqrt{\rho^*} + 9\rho^* = (1 + 3\sqrt{\rho^*})^2$.

Proof. We set $\sigma_0 = \frac{1}{3}\sqrt{\rho^*}$, and use Lemma 4.1.7 to construct a coalition $\vec{\Gamma}$ with respect to the demands $\{\rho_j\}$ on the terminals in $\mathcal{D}$. Now, following the proof of Theorem 3.2.1, we multiply all capacities in G by $1 + 6\sigma_0$ and then round each capacity up to the nearest multiple of σ_0. By Lemma 4.1.7, there is a feasible fractional routing of s to the leaders of $\vec{\Gamma}$ in the resulting graph. Thus, by Corollary 2.9.1, there is a feasible unsplittable routing of s to the leaders of $\vec{\Gamma}$, using paths $\{Q_i\}$.

Again following the proof of Theorem 3.2.1, if $t_j \in \mathcal{D}$ belongs to $\tau^{-1}(A_i)$, then we concatenate the path Q_i with the unique path u_i-t_j path in the arborescence A_i; we use R_j to denote this path. This produces an unsplittable flow path for each terminal $t_j \in \mathcal{D}$. The total relative congestion due to the Q-parts of all these flow paths is at most

$$\frac{\sigma_0 + \rho^*}{\sigma_0} \cdot (1 + 7\sigma_0),$$

and the total relative congestion due to the R-parts of all these flow paths is at most

$$2(\sigma_0 + \rho^*) = \frac{\sigma_0 + \rho^*}{\sigma_0} \cdot (2\sigma_0).$$

Adding these two bounds, we obtain a total relative congestion of at most

$$\frac{\sigma_0 + \rho^*}{\sigma_0} \cdot (1 + 9\sigma_0) = (1 + 3\sqrt{\rho^*})^2.$$

■

When $\rho^* = 1$, this gives a bound of 16. Again, it would be interesting to determine whether the correct bound could be as low as 2.

From this theorem, as in Section 3.2, we obtain the following corollaries.

Corollary 4.2.2 *Let G be directed. Suppose that in $(G, \mathcal{D}, s)$ we have $\rho^* < \frac{1}{9}$, and the cut condition is met by a factor of at least $\beta = (1 - 3\sqrt{\rho^*})^{-2}$. Then $(G, \mathcal{D}, s)$ has a feasible unsplittable solution.*

Corollary 4.2.3 *There is a 16–approximation to $\nu(G, \mathcal{D}, s)$ in an arbitrary directed graph G.*

Maximization

We first observe that, using Corollary 2.9.1 in place of Corollary 2.9.1, Lemmas 3.3.1 and 3.3.2 hold in the directed case. The directed analogue of Theorem 3.3.3 is the following.

Theorem 4.2.4 *Let G be a directed graph and $\rho^* < \frac{11}{49} - \frac{6\sqrt{2}}{49}$. There is a polynomial–time algorithm for the single–source* UFP *producing a flow of value at least $1 - 6\sqrt{\rho^*} + 7\rho^*$ times $\alpha^f(G, \mathcal{D}, s)$.*

Proof. Set $\sigma_0 = \frac{\sqrt{\rho^*}}{3 - 7\sqrt{\rho^*}}$. We require $\rho^* < \frac{11}{49} - \frac{6\sqrt{2}}{49}$ so that the expression $1 - 6\sqrt{\rho^*} + 7\rho^*$ is positive and σ_0 is well–defined.

We compute a fractional maximum flow in $(G, \mathcal{D}, s)$, and let ε_j denote the demand routed to the terminal $t_j \in \mathcal{D}$. Using the set of demand values $\{\varepsilon_j\}$ for the terminals in $\mathcal{D}$, we apply Lemma 4.1.7 to produce a strong coalition $\vec{\Gamma}$ of order σ_0.

Let G' denote the graph obtained by multiplying all capacities of G by $1 + 6\sigma_0$ and then rounding up to the nearest multiple of σ_0. We assign each non–exceptional vertex in $U(\vec{\Gamma})$ a demand of σ_0, and each exceptional vertex in $U(\Gamma)$ a demand equal to the amount assigned to its arborescence. As in the proof of Theorem 3.3.3, there is a maximum flow in $(G', U(\vec{\Gamma}), s)$ that is an unsplittable flow. Moreover, by Lemma 4.1.7, if we were to assign to $u_i \in U(\vec{\Gamma})$ a demand equal to

$$\sum_{j:t_j \in A_i} \varepsilon_j,$$

we would obtain a fractionally feasible flow problem. It follows that the maximum flow value in $(G', U(\vec{\Gamma}), s)$ is at least as large as the fractional maximum flow value in $(G, \mathcal{D}, s)$.

Now, still following the proof of Theorem 3.3.3, we take each $u_i \in U(\vec{\Gamma})$ that is routed by this optimum unsplittable flow and route a subset of the terminals in its arborescence of total demand at least

$$\left(\frac{1 - 2\sigma_0}{1 + 7\sigma_0}\right) \sigma_0 - \rho^*.$$

One argues as before that the set of flow paths for all these routed terminals satisfies the capacity constraints, and we therefore calculate that we have routed a total demand that is within a factor of

$$\frac{\left(\frac{1-2\sigma_0}{1+7\sigma_0}\right) \sigma_0 - \rho^*}{\sigma_0} = 1 - \frac{\frac{9\sqrt{\rho^*}}{3-7\sqrt{\rho^*}}}{\frac{3}{3-7\sqrt{\rho^*}}} - \frac{\rho^*}{\frac{\sqrt{\rho^*}}{3-7\sqrt{\rho^*}}} = 1 - 6\sqrt{\rho^*} + 7\rho^*. \tag{4.3}$$

of the fractional optimum in $(G, \mathcal{D}, s)$. ■

Corollary 4.2.5 *There is a constant–factor approximation algorithm for $\alpha(G, \mathcal{D}, s)$ in an arbitrary directed graph G.*

Routing in Rounds

Theorem 3.4.2, that the realizable subsets in an integral single–source UFP form a matroid, holds in the directed case as well. Consequently, Lemmas 3.4.4, 3.4.5, and 3.4.6 hold in the directed case. Finally, the analogue of Theorem 3.4.7 is the following; the proof is again a direct modification of the undirected version.

Theorem 4.2.6 *Let $\rho^* < \frac{29}{98} - \frac{3\sqrt{22}}{49}$. There is a polynomial–time algorithm that routes $(G, \mathcal{D}, s)$ in $2\lceil \nu_0^f(G, \mathcal{D}, s)\rceil$ rounds.*

Corollary 4.2.7 *Let $(G, \mathcal{D}, s)$ be an arbitrary single–source unsplittable flow problem, with G a directed graph. There is a polynomial–time constant–factor approximation for $\chi(G, \mathcal{D}, s)$.*

Corollary 4.2.8 *Let G be directed. If $(G, \mathcal{D}, s)$ is fractionally feasible, then it is routable without splitting in a constant number of rounds. If additionally $\rho^* < \frac{29}{98} - \frac{3\sqrt{22}}{49}$, then it is routable without splitting in two rounds.*

Chapter 5

The Greedy Algorithm

Two roads diverged in a wood, and I —
I took the one less traveled by,
And that has made all the difference.
— *Robert Frost,* The Road Not Taken.

We now turn our attention to disjoint paths problems in which all terminal pairs do not necessarily share a common vertex; this will essentially be our focus for the remainder of this work. In this chapter, we analyze the performance of a one–pass greedy algorithm in approximating the MDP. Although this algorithm does surprisingly well in certain classes of graphs, its approximation ratio is very large in planar graphs such as the two–dimensional mesh. The development of improved, "non–greedy" algorithms for the mesh and related planar graphs will be taken up in the following chapter.

If G is an arbitrary graph, and $\mathcal{T}$ an arbitrary set of terminal pairs in G, a very natural algorithm to consider for approximating the MDP is the following *greedy algorithm*:

- Proceed through the terminal pairs in one pass.
- When (s_i, t_i) is considered, check whether s_i and t_i belong to the same connected component of G. If so, route (s_i, t_i) on a shortest s_i-t_i path P_i. Delete P_i from G and iterate.

In addition to its simplicity, the greedy algorithm is both on-line and deterministic. As such, lower bounds of [10, 17] imply *a fortiori* that it cannot achieve a good performance ratio on graphs such as trees and two–dimensional meshes.

In this chapter, we consider a slight variant of the greedy algorithm, the *bounded greedy algorithm* (BGA). The BGA is also a determinstic on–line algorithm; it is given an implicit parameter L and runs as follows.

- Proceed through the terminal pairs in one pass.
- When (s_i, t_i) is considered, check whether s_i and t_i can be joined by a path of length of at most L. If so, route (s_i, t_i) on such a path P_i. Delete P_i from G and iterate.

While the BGA suffers from the same bad behavior suggested by the lower bounds of [10, 17], its performance appears to be somewhat easier to analyze in a number of cases than that of the (pure) greedy algorithm. Indeed, developing methods for analyzing the pure greedy algorithm is an interesting open question.

In this chapter, we first prove an upper bound on the performance of the BGA for the maximum disjoint paths problem in an arbitrary graph. We also provide an analogous bound for the UFP. Although there are graphs in which the BGA does quite poorly, the bound we prove here will be useful in analyzing one step of the on–line algorithm we develop in Chapter 6.

We then consider the BGA in bounded-degree expander graphs. Here, we prove that with parameter $L = \Theta(\log n \log\log n)$, the BGA is in fact an $O(\log n (\log\log n)^2)$–approximation for the maximum disjoint paths problem. The nature of this bound is interesting for a number of reasons. First of all, it shows that bounded-degree expanders are a natural class of graphs in which routing algorithms that are both on-line and deterministic can be provably close to optimal. Secondly, the approximation ratio we prove is smaller than the $O(\log^2 n)$ ratio one obtains from the techniques developed by Aumann–Rabani [9] and Raghavan–Upfal [95] for the problem of routing in rounds. (These papers considered the deterministic off-line and randomized on-line settings respectively.) This naturally raises the question of whether the techniques developed here, when incorporated into an algorithm that did not have to contend with the on-line constraint, could lead to even tighter approximation bounds. We leave this as a question for future research.

In the following section, we prove a general bound for the BGA. We then consider the case of bounded–degree expanders in Section 5.2.

5.1 General Bounds

Let $G = (V, E)$ be an arbitrary graph of diameter d, and $\mathcal{T}$ a set of terminal pairs in G.

Theorem 5.1.1 *Define* $d' = \max(d, \sqrt{|E|})$*. The* BGA *with parameter* d' *is a* $(2d'+1)$*–approximation.*

Proof. Let $\mathcal{B}$ denote the set of paths routed by the BGA when run on G with terminal pairs $\mathcal{T}$, and let $\mathcal{O}$ denote a set of disjoint paths connecting a subset of $\mathcal{T}$ of maximum cardinality. Let $\mathcal{O}' \subset \mathcal{O}$ denote the set of paths in $\mathcal{O}$ corresponding to terminal pairs not routed by the BGA. Call a path $Q \in \mathcal{O}'$ *restricted* if it intersects some path in $\mathcal{B}$, and *unrestricted* otherwise. Let $\mathcal{O}'_r$ and $\mathcal{O}'_u$ denote the set of restricted and unrestricted paths in $\mathcal{O}'$, respectively.

Now, every path $Q \in \mathcal{O}'_u$ must have length greater than d'; for otherwise the BGA could have routed the endpoints of Q on the path Q itself. Thus

$$|\mathcal{O}'_u| \leq \frac{|E|}{d'} \leq d' \leq d'|\mathcal{B}|, \tag{5.1}$$

with the last of these inequalities following from the fact that $|\mathcal{B}|$ is always at least 1.

For each path $Q \in \mathcal{O}'_r$, we *charge* it to some path $P \in \mathcal{B}$ that it intersects. Since the paths of $\mathcal{O}$ are edge–disjoint, at most $|P| \leq d'$ paths are charged to P. But since every path in $\mathcal{O}'_r$ is charged to some path in $\mathcal{B}$, we have

$$|\mathcal{O}'_r| \leq d'|\mathcal{B}|. \tag{5.2}$$

Finally, by the definition of $\mathcal{O}'$ we have

$$|\mathcal{O} \setminus \mathcal{O}'| \leq |\mathcal{B}|. \tag{5.3}$$

Adding Equations (5.1), (5.2), and (5.3), we obtain the claimed bound. ■

Now, the approximation ratio implied by this general result is quite large — it has the form $O(n^{\varepsilon})$ for a constant $\varepsilon > 0$. A lower bound due to Blum, Karloff, Fiat, and Rabani [17] implies that no qualitatively better bound is possible for the greedy algorithm on a graph such as the two–dimensional mesh. As the proof of their result is not readily accessible, we provide an alternative proof of it here.

Proposition 5.1.2 (Blum–Karloff–Fiat–Rabani) *For any deterministic on-line algorithm $\mathcal{A}$ for the* MDP *on the $n \times n$ two–dimensional mesh G, there exists a sequence of terminal pairs $\mathcal{T}$, depending on $\mathcal{A}$, such that the ratio of $\alpha(\mathcal{T})$ to the number of pairs routed by $\mathcal{A}$ is $\Omega(\sqrt{n})$.*

Proof. For simplicity, we assume that $\sqrt{n}$ is an integer. Let $A = \{a_1, \ldots, a_n\}$ denote the set of nodes in the first column of G, and let $B = \{b_1, \ldots, b_n\}$ denote the set of nodes in the n^{th} column of G, indexed so that a_i and b_i belong to the i^{th} row of the mesh. Let $\mathcal{T}_0 = \{(a_1, b_n), \ldots, (a_{n-1}, b_2)\}$, which we observe is realizable in G (cf. Theorem 2.2.4). If $\mathcal{A}$, on input $\mathcal{T}_0$, fails to route at least $1 + \sqrt{n}$ terminal pairs, then it is no better than an $\Omega(\sqrt{n})$–approximation. Otherwise, let $\mathcal{T}_0'$ be the minimal prefix of $\mathcal{T}_0$ with the property that $\mathcal{A}$, given $\mathcal{T}_0'$, routes at least $1 + \sqrt{n}$ terminal pairs.

Of the paths produced by $\mathcal{A}$ in routing (a subset of) $\mathcal{T}_0'$, each pair must cross. Let X denote the set of vertices of G at which two of these paths cross; so $|X| \geq \binom{1+\sqrt{n}}{2} > \frac{1}{2}n$. Consider the order in which an arbitrary Hamiltonian path of G meets X, and pair consecutive elements in this order to obtain a partition of X into a set $\mathcal{T}_X$ of terminal pairs. Note that subpaths of this Hamiltonian path constitute a realization of $\mathcal{T}_X$, and hence $\alpha(\mathcal{T}_X) = |\mathcal{T}_X| \geq \frac{1}{4}n$.

Finally, we construct $\mathcal{T}$ by concatenating $\mathcal{T}_0'$ and $\mathcal{T}_X$. Thus we have $\alpha(\mathcal{T}) \geq \frac{1}{4}n$. But $\mathcal{A}$ only routes $1 + \sqrt{n}$ terminal pairs from $\mathcal{T}_0'$, and it then cannot route any pairs from $\mathcal{T}_X$, since all the edges incident to vertices in X have been used. Thus, $\mathcal{A}$ is no better than an $\Omega(\sqrt{n})$-approximation. ∎

Note that the $n \times n$ mesh contains $\Theta(n^2)$ nodes and edges, and has diameter $\Theta(n)$, so the upper bound for the approximation ratio of the BGA provided by Theorem 5.1.1 is $O(n)$. Also, the above lower bound applies only to on–line algorithms that are deterministic; in Chapter 6, we will see that it is possible to obtain an on–line $O(\log n)$–approximation through the use of a randomized algorithm.

We now formulate and prove an analogue of Theorem 5.1.1 for the unsplittable flow problem on uniform capacity graphs. By re-scaling, we therefore assume that every edge in the underlying graph has capacity 1. We must first take care of the following issue — it turns out that for the general UFP, the BGA does not achieve an approximation ratio that can be bounded in terms of the size of the underlying graph G. Specifically, let G consist of a single edge of unit capacity; let (s_1, t_1) request a demand of ε across this edge, and let (s_2, t_2) request a demand of 1 across this edge. Then the BGA will accept the first request and therefore not be able to accept the second; hence its approximation ratio is ε^{-1}, which is independent of the size of G.

Thus, the natural generalization of Theorem 5.1.1 in fact shows that the BGA with parameter $d' = \max(d, \sqrt{|E|})$ is an $O(\rho_*^{-1} d')$–approximation, where $\rho_* = \min_i \rho_i < 1$. However, it turns out that the BGA also obtains an $O(d')$ approximation ratio if we require that the *maximum* demand be bounded away from 1. In particular, let us define an instance of the UFP to be *ε-bounded* if the maximum demand is at most $(1 - \varepsilon)$. Then we show

Theorem 5.1.3 *Let G be a unit–capacity graph, and $\mathcal{T}$ a set of terminal pairs with demands, such that the resulting* UFP *is ε-bounded. As before, let d be the diameter of G and $d' = \max(d, \sqrt{|E|})$. Then the* BGA *with parameter d' is an $O(\varepsilon^{-1} d')$–approximation.*

Proof. The proof will be quite similar to the previous one. For any set $\mathcal{P}$ of paths, we use $\rho(\mathcal{P})$ to denote the total demand of terminal pairs routed by $\mathcal{P}$. Define the sets of paths $\mathcal{B}$, $\mathcal{O}$, and $\mathcal{O}'$ as before, except that the optimum set of paths $\mathcal{O}$ is now the one with maximum $\rho(\mathcal{O})$. We now define a path $Q \in \mathcal{O}'$ to be *restricted* if it uses an edge on which the BGA has routed total demand more than ε. Let $\mathcal{O}'_r$ and $\mathcal{O}'_u$ denote the set of restricted and unrestricted paths in $\mathcal{O}'$, respectively.

First, note that $\rho(\mathcal{B}) \geq \varepsilon$, since the BGA will clearly not reject a terminal pair until it has routed at least ε demand. By direct analogy with the previous proof, one can then show that

$$\rho(\mathcal{O}'_u) \leq d' \leq \varepsilon^{-1} d' \rho(\mathcal{B}).$$

By definition we have

$$\rho(\mathcal{O} \setminus \mathcal{O}') \leq \rho(\mathcal{B}).$$

It remains only to bound $\rho(\mathcal{O}'_r)$, which we do as follows.

Let $Q \in \mathcal{O}'_r$, and e be any edge in Q on which the BGA has routed total demand more than ε. We *charge* Q to e. Let E' denote the set of edges to which some path in $\mathcal{O}'_r$ has been charged; and for $e \in E'$, let $\mathcal{O}'_r(e)$ denote the set of paths charged to e. Thus we have

$$\rho(\mathcal{O}'_r) = \sum_{e \in E'} \rho(\mathcal{O}'_r(e)) \leq |E'|.$$

But every edge in E' carries more than ε units of flow that has been routed by the BGA; thus we have

$$\rho(\mathcal{B}) \geq \frac{\varepsilon|E'|}{\max_{P \in \mathcal{B}} |P|} \geq \frac{\varepsilon|E'|}{d'}.$$

From this it follows that

$$\rho(\mathcal{O}'_r) \leq \varepsilon^{-1} d' \rho(\mathcal{B}).$$

■

5.2 The Algorithm on Bounded–Degree Expanders

We now analyze the BGA on bounded-degree expander graphs. Expanders were introduced in Section 2.8, and we refer the reader to this section for the relevant definitions. Throughout the discussion here, let G be an n-node α-expander graph, with maximum degree Δ. In this section, we deal only with the maximum disjoint paths problem, when all demands and capacities are equal to 1.

The analysis of the BGA on G will in fact be almost the same as it was in the previous section — we consider any set of edge–disjoint paths joining terminal pairs in G, charge these appropriately to paths routed by the BGA, and conclude that the number of paths routed by the BGA is large relative to the optimum. What gives our analysis considerable power in this case is the set of results discussed in Section 2.8, which essentially say that very large partial permutations in G can be routed on short paths.

In particular, Theorem 2.8.2 says the following here: there are constants c_1, c_2, and c_3 such that any partial permutation in G of size at most $c_1 n/(\log n \log\log n)$ can be routed with congestion at most $c_2 \log\log n$ using paths of length at most $c_3 \log n$. Using this, we prove the following. For simplicity, we will suppose that $n \geq 2^{(c_3/c_1\Delta)}$.

Theorem 5.2.1 *The* BGA *with parameter* $L = c_1 \Delta \log n \log\log n$ *is an* $O(\log n(\log\log n)^2)$*–approximation for the* MDP *in* G.

Proof. We use analogous notation to the proof of Theorem 5.1.1. Let $\mathcal{B}$ denote the set of paths routed by the BGA when run on G with terminal pairs $\mathcal{T}$, and let $\mathcal{O}$ denote a set of paths of maximum cardinality. Let $\mathcal{O}' \subset \mathcal{O}$ denote the set of paths in $\mathcal{O}$ corresponding to terminal pairs not routed by the BGA. Since the paths in $\mathcal{O}'$ are edge-disjoint, at most Δ of the pairs routed by $\mathcal{O}'$ contain any fixed vertex $v \in G$. Thus, by Vizing's Theorem (cf. [18]), the pairs routed by $\mathcal{O}'$ can be partitioned into at most $\Delta + 1$ partial permutations; let $\mathcal{O}'_1 \subset \mathcal{O}'$ denote the set of paths corresponding to the largest of these partial permutations. We now define a set of paths $\mathcal{O}''_1$ as follows.

(i) If $|\mathcal{O}'_1| > c_1 n/(\log n \log\log n)$, then by the pigeonhole principle, at least half of the paths in $\mathcal{O}'_1$ have length at most $|E(G)|/(\frac{1}{2}|\mathcal{O}'_1|) \leq c_1 \Delta \log n \log\log n$. We define $\mathcal{O}''_1$ to be this set of paths.

(ii) If $|\mathcal{O}'_1| \leq c_1 n/(\log n \log\log n)$, then by Theorem 2.8.2, the pairs routed by $\mathcal{O}'_1$ can instead be routed on paths of length at most $c_3 \log n$ and congestion at most $c_2 \log\log n$. We define $\mathcal{O}''_1$ to be this set of paths.

First, we claim that every path in $\mathcal{O}''_1$ intersects some path in $\mathcal{B}$. For suppose that $Q \in \mathcal{O}''_1$ is disjoint from all the paths in $\mathcal{B}$. Then the BGA with parameter L would have routed the endpoints of $\mathcal{O}''_1$ on the path Q — a contradiction.

Given this, we charge each path in $\mathcal{O}''_1$ to some path in $\mathcal{B}$ that it intersects. Since the paths in $\mathcal{O}''_1$ have congestion at most $c_2 \log\log n$, and the paths in $\mathcal{B}$ have length at most $L = c_3 \Delta \log n \log\log n$, each path in $\mathcal{B}$ is charged at most $c_2 c_3 \Delta \log n (\log\log n)^2$ times. Thus we have

$$\begin{aligned} |\mathcal{O}'| &\leq (\Delta + 1) \cdot |\mathcal{O}'_1| \\ &\leq 2(\Delta + 1) \cdot |\mathcal{O}''_1| \\ &\leq 2 c_2 c_3 \Delta (\Delta + 1) \log n (\log\log n)^2 \cdot |\mathcal{B}|. \end{aligned}$$

Since by the definition of $\mathcal{O}'$ we also have

$$|\mathcal{O} \setminus \mathcal{O}'| \leq |\mathcal{B}|,$$

the result follows. ■

We can obtain a slight strengthening of this result in the case in which $\mathcal{T}$ is a permutation. Here we make use of Theorem 2.8.3, which in our case can be phrased as follows: There exist constants c_4 and c_5 such that every permutation in G can be routed with congestion at most $c_4 \log n$, using paths of length at most $c_5 \log n$.

Theorem 5.2.2 *The* BGA *with parameter* $L = c_5 \log n$ *is an* $O(\log n)$*-approximation for the* MDP *in* G*, when the set of terminal pairs is restricted to be a permutation.*

Proof. We define $\mathcal{B}$, $\mathcal{O}$, and $\mathcal{O}'$ as before. Let $\mathcal{T}'$ denote the partial permutation consisting of pairs not routed by the BGA. We define $\mathcal{Q}$ to be a set of paths, one for each pair in the partial permutation $\mathcal{T}'$, such that each path in $\mathcal{Q}$ has length at most $c_5 \log n$ and the set $\mathcal{Q}$ has congestion at most $c_4 \log n$. As in the previous proof, we observe that each path in $\mathcal{Q}$ must intersect some path in $\mathcal{B}$; thus, charging paths in $\mathcal{Q}$ to paths in $\mathcal{B}$, we have

$$|\mathcal{T}'| = |\mathcal{Q}| \leq c_4 c_5 \log^2 n |\mathcal{B}|.$$

Since we also have $|\mathcal{T} \setminus \mathcal{T}'| \leq |\mathcal{B}|$, it follows that

$$|\mathcal{B}| \geq \frac{|T|}{1 + c_4 c_5 \log^2 n} = \frac{\frac{1}{2}n}{1 + c_4 c_5 \log^2 n}. \tag{5.4}$$

Now, if

$$|\mathcal{O}'| \leq \frac{|E|}{\frac{1}{2}c_5 \log n} = \frac{\Delta n}{c_5 \log n},$$

then by Equation (5.4), we have $|\mathcal{B}| \leq O(\log n) \cdot |\mathcal{O}|$, as desired. Otherwise, by the pigeonhole principle, there is a set $\mathcal{O}'' \subset \mathcal{O}'$ of size at least $\frac{1}{2}|\mathcal{O}'|$, consisting of paths of length at most $c_5 \log n$. Each such path must meet a path in $\mathcal{B}$; charging paths in $\mathcal{O}''$ to paths in $\mathcal{B}$ as usual, we have $|\mathcal{O}''| \leq c_5 \log n |\mathcal{B}|$. The bound now follows in this case as well, since

$$\begin{aligned}
|\mathcal{O}| &\leq |\mathcal{O}'| + |\mathcal{O} \setminus \mathcal{O}'| \\
&\leq 2|\mathcal{O}''| + |\mathcal{B}| \\
&\leq (2c_5 \log n + 1)|\mathcal{B}|
\end{aligned}$$

∎

Chapter 6

The Two–Dimensional Mesh

"I declare, it's marked out just like a large chess-board!" Alice exclaimed at last.

— *Lewis Carroll,* Through the Looking–Glass.

In the previous chapter, we saw that the greedy algorithm for approximating the maximum disjoint paths problem could only provide a very weak approximation ratio on planar graphs such as the two–dimensional mesh. This tells us that alternative techniques must be developed for dealing with such graphs. Previous work in this direction was undertaken by Awerbuch, Gawlick, Leighton, and Rabani [12], and by Aumann and Rabani [9]. The first of these papers gives an on–line $O(\log n \log\log n)$–approximation for the MDP on the two–dimensional mesh; the second of these gives an off-line $O(\log n)$–approximation.

In this chapter, we obtain a constant–factor approximation for the unsplittable flow problem on the uniform–capacity two–dimensional mesh. This is the first constant–factor approximation for any class of graphs other than trees, and also the first algorithm in the context of the mesh to deal with arbitrary unsplittable demands, rather than just the problem of edge–disjoint paths. We also use the same techniques to provide an on-line $O(\log n)$–approximation algorithm. This matches a lower bound of Awerbuch et al. [12], showing that no on–line algorithm for the MDP on the two–dimensional mesh can be better than an $\Omega(\log n)$–approximation.

The algorithms developed have the property, in common with the previous algorithms for the mesh [12, 9], that they are quite specific to the fixed row/column structure of the mesh. Our basic method, however, can be generalized to a broader class of graphs; in Chapter 7, we obtain a constant–factor approximation for the UFP in a class of locally planar, Eulerian graphs.

The (constant) factor of approximation that we obtain, while not astronomical, is large enough to push the algorithm outside the range of immediate practical utility. While we have not attempted to optimize the constants involved in the development of the algorithm, it seems very likely that obtaining an approximation ratio of 2 or 3 would involve non–trivial work beyond what is presented here. Having said this, it is worth noting two additional points. First, the previous best approximation bounds — both off-line and on-line — for the two-dimensional mesh [12, 9] involve similarly large constants inside the $O(\cdot)$ notation. Moreover, we feel that many of the ideas and components used in the algorithms here may well be of use in suggesting more practical heuristics.

Throughout this chapter, let $G = (V, E)$ denote the $n \times n$ mesh, in which all edges have been given the same capacity. (We can assume by re-scaling that this common capacity is 1.) A very rough sketch of the algorithms is as follows. Since much stronger results are known for cases in which edges have capacity $\Omega(\log n)$, we want to model G by a high-capacity "simulated network" $\mathcal{N}$. To do this we choose, for a constant γ, a maximal set of $\gamma \log n \times \gamma \log n$ subsqures of G subject to the condition that the distance between subsquares is at least $2\gamma \log n$. These will serve as the nodes of $\mathcal{N}$. We now label some pairs of subsquares as "neighbors" and connect them with $\Omega(\log n)$ parallel edges; these are the high-capacity edges of $\mathcal{N}$.

On this network $\mathcal{N}$ we run the algorithm of Awerbuch, Azar, and Plotkin [10] in the on-line case, and the algorithm of Raghavan and Thompson [94] in the off-line case. The algorithms running in $\mathcal{N}$ will return routes consisting of a sequence of neighboring subsquares. To convert such a route into a path in G, we construct disjoint paths between neighboring subsquares. We link a sequence of neighboring pairs together by the natural "crossbar" structures surrounding each subsquare. This leaves us with the problem of routing from each original terminal to the boundary of its subsquare. In the on-line case we will route a total demand of at most 1 from each subsquare, and hence routing out of a subsquare will be easy. In the off-line case, we use the single–source unsplittable flow algorithm from Chapter 3 to route the appropriate subset of terminals to the boundary of the subsquare. Finally, to prove the approximation ratios, we argue that the total demand routed by the optimum in G is upper-bounded by the maximum demand that can be routed in a copy of $\mathcal{N}$ in which all edges have capacity $O(\log n)$.

6.1 The Simulated Network

When G denotes the two-dimensional mesh, let $G[i, j]$ denote the vertex with row number i and column number j, and $G[i : i', j : j']$ the subsquare

$$\{G[p, q] : i \leq p \leq i',\ j \leq q \leq j'\}.$$

As before, let $d(u, v)$ denote the fewest number of edges in a u-v path. By the L_∞ distance between vertices $G[i, j]$ and $G[i', j']$, we mean $L_\infty(G[i, j], G[i', j']) = \max(|i - i'|, |j - j'|)$.

We choose a constant $\gamma > 1$ (any constant will do; it will have an influence on the approximation ratio we obtain). Our first goal is to choose a maximal set of "mutually distant" vertices around which to grow nodes of the simulated network. We divide the the mesh into $\gamma \log n$ by $\gamma \log n$ subsquares as follows.

Definition 6.1.1 *A subsquare of V is called a γ-*block *if for some natural numbers i and j, it is equal to the set*

$$G[(i - 1)\gamma \log n : i\gamma \log n, (j - 1)\gamma \log n : j\gamma \log n].$$

If X is a γ-block with associated natural numbers i and j, then the vertex

$$v = G[(i - \frac{1}{2})\gamma \log n, (j - \frac{1}{2})\gamma \log n]$$

will be called the center *of the γ-block, and we will denote X by C_v. A boundary vertex of X is one with maximal or minimal row or column number. We use X^* to denote the union of X with the (at most) eight other γ-blocks that share boundary vertices with X.*

By a *wall* of X, we mean a maximal set of boundary vertices having the same row or column number. A vertex of X is *internal* if it is not a boundary vertex.

Let V' denote the set of all centers of γ-blocks. We build a graph G' on V' by joining $u, v \in V'$ if the corresponding sets C_u^* and C_v^* intersect at an internal vertex. We now run a randomized version of Luby's maximal independent set algorithm [79] on this graph. That is, each vertex picks a random number between 1 and j, where j is large enough that the probability of ties is small. If v has a number higher than any of its neighbors', it enters the

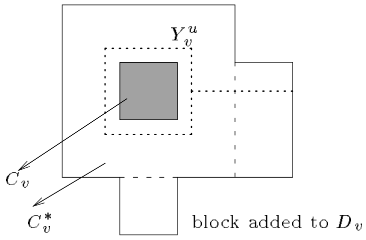

Figure 6-1: A cluster and its surroundings

MIS and its neighbors drop out. We then iterate.

Let $M \subset V'$ denote the resulting MIS. For any $v \in V'$, C_v^* intersects C_u^* internally for at most 24 other vertices $u \in V'$; thus with probability $\frac{1}{25} - o(1)$, v will enter M on the first iteration. Moreover, if $u, v \in V'$ are at a distance of at least $11\gamma \log n$ from each other, then they have no common neighbors in G', and so these events are independent. Thus,

Lemma 6.1.2 *Let $u, v \in V'$ be such that $d(u, v) \geq 11\gamma \log n$. Then with constant probability, both u and v belong to the set M constructed above.*

If $v \in M$, we will call C_v a *cluster*. We now want to construct internally disjoint *enclosures* D_v around each C_v^*, for $v \in M$, such that every vertex of G belongs to some enclosure and such that each D_v is a union of γ-blocks. The sets C_v^* are disjoint and are unions of γ-blocks, but they do not cover all of G. However, by the maximality of M, any γ-block X that does not belong to C_v^* for some $v \in M$ must share a boundary vertex with such a C_v^*. For each such X, we pick such a C_v^* arbitrarily and add X to D_v. Thus the D_v now form a partition of G, and each D_v is a union of γ-blocks.

We now define $\mathcal{N}$ to be the graph on vertex set M, with u and v joined by an edge if some γ-block of D_u shares a wall with a γ-block of D_v. As argued above, any γ-block in D_v must belong to the 5×5 set of γ-blocks centered at C_v, and from this it is easy to argue that at most 20 γ-blocks not contained in D_v can share a wall with a γ-block of D_v. Thus we have

Lemma 6.1.3 *The degree of a vertex in $\mathcal{N}$ is at most* 20.

If we were to contract each enclosure D_v, the resulting graph would contain all the edges of $\mathcal{N}$ with multiplicity $\Theta(\log n)$ — enough to run a "high-capacity" algorithm. But given a routing in $\mathcal{N}$, we then run into the following problem: we need to convert the sequence of neighboring clusters connecting the clusters of s_i and t_i in the contracted graph back to a routing in G. For this we use the natural "crossbar" structures in the mesh in the on-line case, with additional flow techniques in the off-line case.

We start by developing the "crossbar" structures we use.

Definition 6.1.4 *A* v-ring *is a subgraph* $G[X]$*, where* X *is the set of vertices at* L_∞ *distance exactly* r *from* v*, for some* r *between* $\frac{1}{2}\gamma \log n$ *and* $\gamma \log n$*. Thus a* v*-ring is either a cycle or a path, depending on whether* C_v *has a wall on the boundary of* G*. If* R *and* R' *are* v*-rings, we say that* R *is* inside *(resp.* outside*)* R' *if the distance from* R *to* v *is less than (resp. greater than) the distance from* R' *to* v*.*

Note that the set of v-rings are the $\frac{1}{2}\gamma \log n$ disjoint cycles around v right outside the boundary of C_v, in the "inner half" of $C_v^* \setminus C_v$.

For each pair (v, w) that is an edge of $\mathcal{N}$, we choose, for a sufficiently small constant β, a set $\tau_{v,w}$ of $\beta \log n$ edges in $\delta(D_v, D_w)$. Let τ_v' denote the set of all vertices in D_v that are incident to an edge in some $\tau_{v,w}$. We also choose a set σ_v' of $\beta \log n$ vertices evenly spaced on the outer boundary of C_v. Now by Lemma 6.1.3, we can choose β small enough that $|\tau_v' \cup \sigma_v'| \leq \frac{1}{2}\gamma \log n$, and hence we can associate a different v-ring to each vertex in $\tau_v' \cup \sigma_v'$. Moreover, in a straightforward fashion we can construct edge-disjoint paths from each such vertex to its associated ring. We assign the outermost $\beta \log n$ rings to the vertices of σ_v'. For $u \in \tau_v' \cup \sigma_v'$, let Y_v^u denote the union of the ring associated with u with the path from u to this ring. Then we have

Lemma 6.1.5 *The (non-simple) paths* Y_v^u *are mutually edge-disjoint, and each pair meets at some vertex of* $C_v^* \setminus C_v$*.*

Proof. The paths are edge-disjoint by construction. Now suppose $u, w \in \tau_v'$, and that the ring associated with u is inside the ring associated with w. Then the path from u to its ring must intersect the ring associated with w, and so Y_v^u and Y_v^w meet at a vertex. The same argument applies if $u, w \in \sigma_v'$, and also if $u \in \sigma_v'$ and $w \in \tau_v'$ because in this case the ring associated with u lies outside the ring associated with w. ■

Finally, let us show that the optimal unsplittable flow in this simulated network $\mathcal{N}$ provides an approximate bound on the optimal unsplittable flow in G. Let $\mathcal{N}(c)$ denote the graph $\mathcal{N}$ in which each edge has been given a capacity of c. If u is a vertex in a cluster C_v, we define $\psi(u) = v$; we say that $\psi(u)$ is undefined if $u \notin \cup_{v \in M} C_v$. If (s_i, t_i) is a terminal pair with both ends in clusters of the form C_v, we define $\psi(s_i, t_i) = (\psi(s_i), \psi(t_i))$. We now have the following fact.

Lemma 6.1.6 *Let $\mathcal{T}'$ be a realizable subset of $\mathcal{T}$, with the property that all terminals in $\mathcal{T}'$ belong to $\cup_{v \in M} C_v$. Then $\psi(\mathcal{T}')$ is realizable in $\mathcal{N}(5\gamma \log n)$.*

Proof. Consider a realization of $\mathcal{T}'$ in G, and let P_i denote the path associated with $(s_i, t_i) \in \mathcal{T}'$. We construct the following path for $(\psi(s_i), \psi(t_i))$ in $\mathcal{N}$ — when P crosses from D_w into $D_{w'}$, we add an edge from w to w'. Since $|\delta(D_w, D_{w'})| \leq 5\gamma \log n$, this is an upper bound on the total demand of paths in the constructed routing that use the edge (w, w') in $\mathcal{N}$. ■

We are now ready to describe and analyze the routing algorithms themselves.

6.2 The On–Line Algorithm

For the on–line algorithm, as suggested at the beginning of this chapter, we will use a variant of the high–capacity algorithm of Awerbuch, Azar, and Plotkin [10] to determine "global routes" in the simulated network $\mathcal{N}$. However, deciding whether the terminals selected by this high–capacity algorithm can be routed out of the clusters in which they originate must be made by separate on–line algorithms, one for each cluster. Our goal is to show that the final on–line algorithm obtained in this way is an $O(\log n)$–approximation.

Thus, in the following subsection, we discuss a general method for analyzing such "combinations" of on–line algorithms. We then develop the variant of the high–capacity algorithm that we need in Section 6.2.2. Finally, in Sections 6.2.3 and 6.2.4, we describe the on–line algorithm itself.

6.2.1 Combining Maximization Algorithms

In routing connections on-line, we adopt an approach in which the decision whether to accept a given connection is made by a combination of several algorithms — the connection is accepted if each of the individual algorithms accepts it. From the approximation ratios of these individual algorithms one can infer an approximation ratio for this combined algorithm; in this section we show how this can be done, in a fairly general framework.

Let U denote a finite set, with $S_1, \ldots, S_n$ subsets of U such that $U = \cup_i S_i$. Let $\mathcal{F}_i$ denote a collection of subsets of S_i closed with respect to inclusion, and let

$$\mathcal{F} = \{C : \forall i \ (C \cap S_i) \in \mathcal{F}_i\}.$$

Each element $u \in U$ has associated with it a positive *weight* w_u. Given a set $U' \subset U$, define

$$w(U') = \sum_{u \in U'} w_u,$$

and define $\mu(U')$ to be the maximum weight of a member of $\mathcal{F}$ contained in U'.

We wish to design an algorithm for the following on-line maximization problem with respect to U and $\mathcal{F}$. Elements of some $U' \subseteq U$ arrive in an arbitrary order, and on each element our algorithm either accepts or rejects it; the goal is to accept a subset of these elements that is in $\mathcal{F}$ and whose weight is as large as possible relative to $\mu(U')$. Our algorithm will be called a c-approximation if it always accepts a set of size at least $\frac{1}{c}\mu(U')$.

We can define the corresponding on-line maximization problems with respect to S_i and $\mathcal{F}_i$, for each $i = 1, \ldots, n$, in exactly the same way. Say that for each i, we are given an algorithm A_i which is a c_i-approximation for the problem associated with S_i and $\mathcal{F}_i$. Moreover, we assume that the state of A_i is completely determined by the set of elements it has accepted so far. We then define our "combined algorithm" $A = \wedge_{i=1}^{n} A_i$ for the on-line maximization problem with respect to U and $\mathcal{F}$ as follows. As each $u \in U'$ is presented to A, it accepts u iff for each i such that $u \in S_i$, A_i accepts u. The total set accepted so far, intersected with S_i, serves as the state for A_i. Let c^* denote the maximum approximation ratio of any of the algorithms A_i, and suppose each element of U appears in at most d of the S_i.

Theorem 6.2.1 *A is a c^*d-approximation.*

Proof. Assume the algorithm A was presented with a set U' and it returned X. Let Y denote a member of $\mathcal{F}$ contained in U' of maximum weight; we show that $w(Y) \leq c^*d \cdot w(X)$. Let R_i'

denote the elements of $Y \setminus X$ that were rejected by algorithm A_i, $J_i = X \cap Y \cap S_i$ the elements of S_i accepted by *both* A and the optimal solution Y, and $R_i = J_i \cup R_i'$. Note that $Y = \cup_i R_i$, and $R_i \subset Y \cap S_i \in \mathcal{F}_i$.

We want to prove that $w(R_i) \leq c^* w(X \cap S_i)$ $(i = 1, \ldots, n)$. Set $U_i' = (X \cap S_i) \cup R_i$; these are the elements of $U' \cap S_i$ either accepted by A or rejected by A_i. Order U_i' as it appears in U', and present it as input to A_i. Then as in the running of the combined algorithm A, A_i will accept precisely the set $X \cap S_i$. Since A_i is a c^*-approximation, and $R_i \in \mathcal{F}_i$, we have

$$w(R_i) \leq c^* w(X \cap S_i).$$

We also have $w(Y) \leq \sum_i w(R_i)$, and by the definition of d we have $\sum_i w(X \cap S_i) \leq d \cdot w(X)$. Thus

$$w(Y) \leq \sum_i w(R_i) \leq c^* \sum_i w(X \cap S_i) \leq c^* d \cdot w(X). \tag{6.1}$$

■

In the natural way, one can define a fractional version of this model, as follows. A "fractional" choice of a subset of U is specified by a vector $Y \in \mathbf{R}^{|U|}$, indicating the fraction of each element that is taken. The weight of such a fractional subset Y is now defined to be

$$w(Y) = \sum_{u \in U} Y(u) \cdot w_u.$$

A collection $\mathcal{F}_i$ of "feasible" subsets of U is now specified by some subset of $[0,1]^{|U|}$ that is closed downward in the sense that $Y \in \mathcal{F}_i$ and $Y' \leq Y$ coordinatewise implies that $Y' \in \mathcal{F}_i$.

The algorithms A_i work as before, and in particular can only choose integer–valued subsets of U. However, their respective approximation ratios c_i are now with respect to the *fractional* optimum. In this setting, one can show the following by a minor modification of the proof above.

Theorem 6.2.2 *If each A_i is a c^*-approximation relative to the fractional optimum, then A is a c^*d-approximation relative to the fractional optimum.*

Proof. As in the previous proof, let U' denote the sequence of elements presented to A, X the integer–valued set returned by A, and Y a fractional subset of U' of maximum weight. (A fractional set attaining the maximum exists since the $\mathcal{F}_i$ are closed subsets of $[0,1]^{|U|}$, and hence compact.) Define R_i' to be the fractional set obtained by assigning zero to all coordinates of Y except those corresponding to elements rejected by algorithm A_i, J_i the fractional set obtained by assigning zero to all coordinates of Y except those corresponding to elements in $X \cap S_i$, and $R_i = R_i' + J_i$. Note in this last definition that R_i' and J_i are non-zero on disjoint sets of coordinates; thus, $R_i \leq Y$, and hence $R_i \in \mathcal{F}_i$.

With these definitions, one can simply follow the previous proof. One verifies that $w(R_i) \leq c^* w(X \cap S_i)$, by the definition of c^*, and hence that

$$w(Y) \leq \sum_i w(R_i) \leq c^* \sum_i w(X \cap S_i) \leq c^* d \cdot w(X). \tag{6.2}$$

■

6.2.2 The AAP Algorithm

If H is a graph with n nodes in which each edge has capacity at least $\log 2n$, the on–line UFP algorithm of Awerbuch, Azar, and Plotkin [10] achieves an approximation ratio of $2\log 4n$ for arbitrary sets of terminal pairs. For our purposes, we need to develop a strengthening of this "AAP algorithm": we want an approximation guarantee that holds against the fractional optimum; and when we deal with the more general class of graphs developed in the next chapter, we want only to require capacities to be $\varepsilon \log n$, for an arbitrary $\varepsilon > 0$. Here, we show how to obtain such an algorithm.

Theorem 6.2.3 *If all edge capacities are at least $(\varepsilon \log n + 1 + \varepsilon)$, there is a deterministic on-line* UFP *algorithm that provides an $O(2^{1/\varepsilon} \log n)$–approximation relative to the fractional optimum.*

Proof. We follow the AAP algorithm and its analysis very closely. We vary a little from their notation, since we assume all requests have infinite duration, and the profit of a request is proportional to its demand. Thus, the i^{th} request is specified by a pair (s_i, t_i) of terminals, and a demand $\rho_i \leq 1$. We define the *profit* of the connection to be $\omega_j = n\rho_j$.

Define $\mu = 2^{1+1/\varepsilon} n$, so we have

$$\varepsilon \log \mu = \varepsilon \log n + 1 + \varepsilon.$$

Let u_e denote the capacity of edge e; thus we can assume that for all j and e,

$$u_e \geq \varepsilon \log \mu.$$

With this value of μ, we now run the AAP algorithm – for the sake of completeness, we state this algorithm here.

For $j = 1, 2, \ldots, k$:

Define λ_e^j to be the fraction of u_e consumed by paths already routed.

Define $c_e^j = u_e(\mu^{\lambda_e^j} - 1)$.

For a request (s_i, t_i), route it on any path P satisfying $\sum_{e \in P} \frac{\rho_j}{u_e} c_e^j \leq \omega_j$.

If no such path is available, then reject the request.

First we argue why the relative load on an edge will never exceed 1. At the moment before this happened, on edge e say, we had

$$\lambda_e^j > 1 - \frac{\rho_j}{u_e} \geq 1 - \frac{1}{\varepsilon \log \mu}.$$

Thus

$$\begin{aligned}
\frac{c_e^j}{u_e} &= \mu^{\lambda_e^j} - 1 \\
&> \mu^{1 - \frac{1}{\varepsilon \log \mu}} - 1 \\
&= \frac{\mu}{2^{1/\varepsilon}} - 1 = 2n - 1 \\
&\geq n.
\end{aligned}$$

So we have

$$\frac{\rho_j c_e^j}{u_e} > n\rho_j = \omega_j$$

and thus the connection could not have used this edge.

Suppose there are a total of k requests. Let A denote the set of requests routed by the AAP algorithm. Then we show

$$2^{1+1/\varepsilon} \log \mu \sum_{j \in A} \omega_j \geq \sum_e c_e^{k+1}. \tag{6.3}$$

As in the proof in [10] we show this by induction on the number of admitted requests, via proving that if the algorithm admits the j^{th} request, we have

$$\sum_e c_e^{j+1} - c_e^j \leq 2^{1+1/\varepsilon} \omega_j \log \mu.$$

So consider edge e, on the path used by the AAP algorithm for connection j. We have

$$c_e^{j+1} - c_e^j = u_e \left(\mu^{\lambda_e^j} (2^{(\log \mu) \cdot \rho_j / u_e} - 1) \right).$$

Now the exponent on the 2 is at most $1/\varepsilon$, and for $x \in [0, 1/\varepsilon]$ we clearly have $2^x - 1 \leq 2^{1/\varepsilon} \cdot x$. Thus

$$\begin{aligned} c_e^{j+1} - c_e^j &\leq u_e \cdot \mu^{\lambda_e^j} \cdot 2^{1/\varepsilon} \cdot \log \mu \cdot \rho_j / u_e \\ &= \mu^{\lambda_e^j} \cdot 2^{1/\varepsilon} \cdot \log \mu \cdot \rho_j \\ &= 2^{1/\varepsilon} \cdot \log \mu \cdot \left[\frac{c_e^j}{u_e} \cdot \rho_j + \rho_j \right]. \end{aligned}$$

Summing over all edges gives the desired bound.

Finally, we show that the expression

$$\sum_e c_e^{k+1} \tag{6.4}$$

is an upper bound on the profit of the *fractional* optimum minus the profit of the on-line algorithm. ([10] shows this for the integer optimum, but the proof is essentially the same.)

Let $\mathcal{Q}$ denote the set of indices which were rejected by the on-line algorithm but for which a positive fraction of the demand was routed by the optimum. For $j \in \mathcal{Q}$, suppose that the fractional optimum uses paths $P_j^1, \ldots, P_j^z$, with weights $\gamma_j^1, \ldots, \gamma_j^z$. Then since j was rejected by the on-line algorithm, and the costs are monotonic in the indices, we must have

$$n\rho_j = \omega_j \leq \sum_{e \in P_j^i} \frac{\rho_j c_e^{k+1}}{u_e}$$

for any i, j. Also, for any edge e we have

$$\sum_{i,j : e \in P_j^i} \frac{\gamma_j^i \rho_j}{u_e} \leq 1,$$

and hence we have

$$\begin{aligned} \sum_j \sum_i \gamma_j^i \omega_j &\leq \sum_j \sum_i \sum_{e \in P_j^i} \frac{\gamma_j^i \rho_j c_e^{k+1}}{u_e} \\ &\leq \sum_e c_e^{k+1} \cdot \sum_{i,j : e \in P_j^i} \frac{\gamma_j^i \rho_j}{u_e} \\ &\leq \sum_e c_e^{k+1}. \end{aligned}$$

Combining the bounds in Equations (6.3) and (6.4), we obtain the claimed approximation ratio. ■

A lower bound of [10] implies that the factor of $2^{1/\varepsilon}$ is unavoidable for deterministic on-line algorithms.

6.2.3 Routing Long Connections

Say that a terminal pair $(s_i, t_i) \in \mathcal{T}$ is *short* if $d(s_i, t_i) < 16\gamma \log n$, and *long* otherwise. Say that (s_i, t_i) is *light* if $\rho_i \leq \frac{1}{2}$, and *heavy* otherwise. The on-line algorithm makes an initial random decision whether to accept only short connections or only long connections, and an independent random decision whether to accept light or heavy connections; this costs at most a factor of two in the approximation ratio. In this section, we give an $O(\log n)$ approximation for long connections (either light or heavy); in the following section, we show the bounded greedy algorithm of Chapter 5 gives an $O(\log n)$–approximation for short connections.

For concreteness, we will present the following in terms of light connections. The algorithm is essentially identical for the case of heavy connections; this can be easily verified.

First, we only consider terminal pairs with both ends in sets of the form C_v — denote this set of terminal pairs by $\mathcal{T}_M$. If $\mathcal{T}^*$ denotes a realizable subset of maximum size, then by Lemma 6.1.2, the expected demand associated with terminal pairs in $\mathcal{T}^*$ that belong to $\mathcal{T}_M$ is a constant fraction of $\rho(\mathcal{T}^*)$. Thus we only lose a constant factor in the approximation ratio by restricting attention to $\mathcal{T}_M$.

We first define an on–line routing problem in the simulated network $\mathcal{N}(\frac{1}{2}\beta \log n)$; we will use the solution generated by this problem to guide the construction of paths in G. Recall that for $u \in C_v$, we define $\psi(u) = v$. The input for our problen in the simulated network will simply be the sequence of terminal pairs $(\psi(s_i), \psi(t_i))$, where (s_i, t_i) is the sequence of pairs presented to the algorithm running in G. (We omit those pairs for which both ends do not lie in clusters of the form C_v.) The demand for $(\psi(s_i), \psi(t_i))$ will be ρ_i, the same as it is in G. Our algorithm for this problem in the simulated network is as follows: we route (v, w) if

(i) the AAP algorithm on $\mathcal{N}(\frac{1}{2}\beta \log n)$ accepts (v, w), and

(ii) routing (v, w) will not cause the total routed demand with an end equal to v (resp. w) to exceed 1.

Lemma 6.2.4 *The above algorithm is an $O(\log n)$-approximation relative to the fractional optimum in $\mathcal{N}(\frac{1}{2}\beta \log n)$.*

Proof. Let (v, w) be a requested connection in $\mathcal{N}$. Following the approach developed in Section 6.2.1, we view (v, w) as being "judged" by three on-line algorithms: the AAP algorithm, an algorithm A_v that will only accept a total demand of 1 originating at v, and an algorithm A_w that will only accept a total demand of 1 originating at v.

By Theorem 6.2.3, the AAP algorithm is an $O(\log n)$–approximation relative to the fractional optimum. To see that the algorithm A_v is an $O(\log n)$–approximation, let y_v denote the total amount of demand associated with terminal pairs with one end equal to v. If $y_v < 1$, then A_v by itself can accept all the demand; otherwise, A_v by itself will accept at least $\frac{1}{2}$ units of demand, while the maximum fractional weight of connections that the optimum can accept originating at any one node of $\mathcal{N}$ is $O(\log n)$. The same argument applies to A_w.

Thus, applying Theorem 6.2.2 with $c^* = O(\log n)$ and $d = 3$, we see that the combined routing algorithm is an $O(\log n)$-approximation relative to the fractional optimum. ■

Our on-line algorithm in G simply runs the above simulation; whenever $(\psi(s_i), \psi(t_i))$ is accepted, it routes the pair (s_i, t_i) in G using the paths constructed in Lemma 6.1.5. The following lemma says that it will not run out of "bandwidth" while doing this.

Lemma 6.2.5 *The algorithm in G can route all the connections accepted by the simulation.*

Proof. When the simulation accepts $(\psi(s_i), \psi(t_i))$, it specifies a sequence of neighboring clusters $C_{v_1}, C_{v_2}, \ldots, C_{v_r}$, where $v_1 = \psi(s_i)$ and $v_r = \psi(t_i)$.

The algorithm in G routes (s_i, t_i) by simply moving from one cluster in this sequence to the next using paths of the form $Y^u_{v_i}$. More concretely, it first chooses any $w \in \sigma'_{v_1}$ and $w' \in \sigma'_{v_r}$ and sets $Z_0 = Y^w_{v_1}$ and $Z_r = Y^{w'}_{v_r}$. Then for each $j = 1, \ldots, r-1$, it chooses any edge $(u, u') \in \tau_{v_j, v_{j+1}}$ on which the available capacity is at least $\rho_i \leq \frac{1}{2}$, and it sets $Z_j = Y^u_{v_j} \cup Y^{u'}_{v_{j+1}}$. We observe that such an edge can always be found; for if not, then more than $\frac{1}{2}$ units of demand cross each of the $\beta \log n$ edges in $\tau_{v_j, v_{j+1}}$, contradicting the fact that our set of paths in $\mathcal{N}(\frac{1}{2}\beta \log n)$ respected the capacity constraint of edge (v_j, v_{j+1}).

Finally, by Lemma 6.1.5, the union $Z_0 \cup \cdots \cup Z_r$ contains a path from w on the boundary of C_{v_1} to w' on the boundary of C_{v_r}. Since the total amount of demand routed out of C_{v_1} and C_{v_r} is at most 1, s_i-w and w'-t_i paths respecting the capacity constraints can be easily found. ■

Now, since the on-line algorithm is an $O(\log n)$-approximation relative to the fractional optimum in $\mathcal{N}(\frac{1}{2}\beta \log n)$, it is also an $O(\log n)$-approximation relative to the fractional optimum in $\mathcal{N}(5\gamma \log n)$. By Lemma 6.1.6, this is at least the weight of the maximum realizable subset of $\mathcal{T}_M$, which by Lemma 6.1.2 is, in expectation, within a constant factor of the weight of the maximum realizable subset of $\mathcal{T}$. Thus the algorithm here is an $O(\log n)$-approximation for the case of long connections.

6.2.4 Routing Short Connections

Again, we will discuss the case of light connections; the algorithm for heavy connections is essentially the same, and easier. We first require the following fact.

Lemma 6.2.6 *Let $X = G[(i-r):(i+r), (j-r):(j+r)]$ for some natural numbers i, j, and r; let $Y = G[(i-2r):(i+2r), (j-2r):(j+2r)]$; and let $\mathcal{T}'$ be a set of terminal pairs with both* *ends in X. Then the maximum weight of a subset of $\mathcal{T}'$ that is realizable in $G[Y]$ is at least $\frac{1}{8}$ the maximum weight of a subset of $\mathcal{T}'$ that is realizable in G.*

Proof. Let $\mathcal{T}'' \subset \mathcal{T}'$ denote a realizable set of maximum weight, and consider the set of paths in a routing of $\mathcal{T}''$. Since $|\delta(X)| \leq 8r$, a total demand of at most $4r$ can make use of paths that leave X. Let $\mathcal{T}_0''$ denote the terminal pairs corresponding to these paths. We show how to route at least $\frac{1}{8}$ of the demand of $\mathcal{T}_0''$ within $G[Y]$, which will imply the lemma.

Let $W_1, \ldots, W_r$ denote the r disjoint rings in $G[Y \setminus X]$ (as defined in the previous section). For $(s_i, t_i) \in \mathcal{T}_0''$, let (a_i, b_i) denote the first and last intersection points of the s_i-t_i path with $\delta(X)$. We arbitrarily choose a set of terminal pairs in $\mathcal{T}_0''$ whose total demand is between $\frac{1}{2}$ and 1; we route all these pairs by joining their "breakpoints" using a subpath of W_1. We then do the same for $W_2, \ldots, W_r$; in this way, we route at least $\frac{1}{8}$ of the demand in $\mathcal{T}_0''$. ∎

The algorithm for short (and light) connections is now as follows. Set $r = 32\gamma \log n$, and run Luby's algorithm to find a subset M' of V, maximal subject to the property that $L_\infty(u, v) \geq 4r$ for $u, v \in M'$. For $u \in M'$, define X_u to be the set of all vertices whose L_∞ distance from u is at most r, and Y_u to be the set of all vertices whose L_∞ distance from u is at most $2r$.

Lemma 6.2.7 *If (s_i, t_i) is a short connection, then with constant probability there is a u such that $s_i, t_i \in X_u$.*

Proof. A sufficient condition for some X_u containing both s_i and t_i to enter the MIS in the first iteration is that among all vertices within L_∞ distance $4r$ of s_i, the one that picks the highest number has L_∞ distance at most $\frac{1}{2}r$ from s_i. This happens with constant probability. ∎

Let $\mathcal{T}_u$ denote the set of short connections both of whose ends lie in X_u. We now run the greedy algorithm of Theorem 5.1.3 on the (disjoint) subgraphs $G[Y_u]$ simultaneously, using the $\mathcal{T}_u$ as the sets of terminal pairs. By Theorem 5.1.3 and Lemma 6.2.6, this is an $O(\log n)$ approximation in each subgraph. Thus, by Lemma 6.2.7, we are, in expectation, within a constant factor of optimal in routing short connections. Thus, we have the main result of this section.

Theorem 6.2.8 *There is an on-line $O(\log n)$-approximation for the UFP in the two-dimensional mesh.*

6.3 The Off-Line Algorithm

In the off–line case, when the entire set of terminal pairs is given as input, we are able to obtain a constant–factor approximation. As in the on–line case, we will partition the terminal pairs based on whether they are *long* and *short* connections; and whether they are *heavy* or *light*. Changing terminology slightly from the previous section, we will say that a terminal pair (s_i, t_i) is *light* if $\rho_i \leq \frac{1}{16}$. Thus we obtain a partition into four sets; by approximating the maximum realizable subset in each set and taking the one in which we route the most demand, we obtain a constant–factor approximation for the overall problem. Throughout this section, we will consider the problem of routing light connections. Again, the algorithm for heavy connections is essentially the same, and easier; we will discuss this in a little more detail at the end.

First we deal with the long connections. As in the on–line case, our algorithm only considers the set $\mathcal{T}_M$ of terminal pairs with both ends in clusters of the form C_v; this results in a loss of only a constant factor in the approximation ratio. But whereas we previously used the high–capacity on–line algorithm of Awerbuch, Azar, and Plotkin, we now apply the randomized rounding algorithm of Raghavan and Thompson [94, 93] in the simulated network. (See Section 2.5 for the details of this algorithm.) For the problem in which we have terminal pairs $(\psi(s_i), \psi(t_i))$ in the simulated network $\mathcal{N}$, this gives a constant–factor approximation. But to get a constant-factor approximation for the problem in G, we also have to be within a constant factor of the optimum in routing terminals out of the clusters. (In the on-line algorithm it was enough to route only a single unit of demand). To this end, we build the following more complicated network $\mathcal{N}'$. Let z_v denote the node representing $v \in M$ in the network $\mathcal{N}$; we construct $\mathcal{N}'$ by attaching C_v to z_v via an edge from z_v to *each* node in $\pi(C_v)$. Let $\mathcal{N}'(\gamma)$ denote the network $\mathcal{N}'$ in which each edge *between nodes of the subgraph* $\mathcal{N}$ has capacity γ, and all other edges have unit capacity.

Let f^* denote the value of the fractional optimum solution to the UFP in $\mathcal{N}'(\frac{1}{2}\beta \log n)$. We now run the randomized rounding algorithm on $\mathcal{N}'(\frac{1}{2}\beta \log n)$, with a sufficiently small scaling factor μ. With high probability the parts of all selected paths lying in the subgraph $\mathcal{N}(\frac{1}{2}\beta \log n)$, taken together, do not violate any capacity constraint; and the number of pairs that are rounded up is within a constant factor of the fractional optimum f^*. We now must convert the selected paths in $\mathcal{N}'(\frac{1}{2}\beta \log n)$ into s_i-t_i paths in G. We can use the technique of the previous section to produce, for each selected pair (s_i, t_i), a "global" path P_i that begins at $\tau'_{\psi(s_i)} \subset \pi(D_{\psi(s_i)})$ and ends at $\tau'_{\psi(t_i)} \subset \pi(D_{\psi(t_i)})$.

The real problem is how to find paths *within* the clusters such that each s_i (resp. t_i) that has been rounded up can reach the endpoint of this associated global path on $\tau'_{\psi(s_i)}$ (resp. $\tau'_{\psi(t_i)}$). For this, the paths returned by the randomized rounding are of no value, since the edges of $\mathcal{N}'$ within the clusters C_v have only unit capacity. Instead we argue as follows.

Let S_v denote the set of terminals in C_v that are rounded up. Each is trying to "get out" to its associated path that begins at τ'_v. Given the crossbar in $C^*_v \setminus C_v$, it is sufficient to route each terminal in S_v to any vertex in σ'_v. So this leaves us with the following *escape problem.* We are given the set S_v of terminals that have been rounded up, and we want to route a large fraction of their total demand to σ'_v. The following lemma, whose proof contains the central step of the algorithm, says that this can be done. By abuse of notation, if X is a set of vertices, we will we use $\rho(X)$ to denote the total demand of terminal pairs with both ends in X.

Lemma 6.3.1 *For a sufficiently small (constant) value of μ, there is a constant $c < 1$ so that the following holds. We can find sets $S'_v \subset S_v$, such that*

(i) if one end of a pair (s_i, t_i) belongs to $\cup_v S'_v$ then so does the other,

(ii) each set S'_v can be linked to $\sigma'_v \subset \pi(C_v)$ via edge-disjoint paths, and

(iii) the probability that $\rho(\cup_v S'_v) > cf^*$ is at least a constant, where the probability is taken over the randomized rounding that produced the sets S_v.

Proof. We will first construct such a set with condition (ii) weakened to the requirement that each S'_v can be linked to $\pi(C_v)$ via edge-disjoint paths (rather than to the smaller set σ'_v). This is sufficient to imply the lemma, as follows. Suppose we obtain such sets S''_v. For each $s \in S''_v$, identify the vertex in σ'_v closest to s. From among the terminal pairs in $\cup_v S''_v$, we now choose a set $\mathcal{T}'$ of terminal pairs that is *maximal* subject to the constraint that the total demand of all terminals in $\mathcal{T}'$ with the same closest vertex in σ'_v is at most 1. It is easy to verify that by the maximality of $\mathcal{T}'$, we have

$$\rho(\mathcal{T}') \geq \frac{\rho(\cup_v S''_v)}{8\gamma\beta^{-1} + 1}.$$

Thus, letting the terminals in $\mathcal{T}'$ constitute the sets S'_v, we satisfy the requirements of the lemma.

This allows us to deal with a standard escape problem on a rectangular mesh: each terminal is allowed to "escape" to any vertex on the boundary. By attaching a source v^* by an edge to every node in $\pi(C_v)$, we obtain a single–source unsplittable flow problem, with terminal set S_v. Now we observe the following fact: the cut condition for this problem holds by a factor of α if and only if, for all p, q, it holds by a factor of α for every $p \times q$ sub-rectangle. To see this, note that on a rectangular mesh, the smallest rectangle enclosing any connected cut has no greater capacity, and contains at least as many terminals, as the original cut; thus the cut condition holds by a factor of α if and only if it holds by a factor of α for all subrectangles.

Recall that the maximum demand is $\frac{1}{16}$, so by Corollary 3.2.2, our unsplittable escape problem in C_v is feasible if the cut condition holds by a factor of 4. This suggests the following algorithm to construct the set S''_v. Call a rectangle *overfull* if it does not meet the cut condition by a factor of 4. We go through each $s \in S_v$, deleting it if it is contained in any overfull rectangle — we also then delete its matching terminal in some other cluster. This results in a set S''_v, which by the above argument can be completely routed, unsplittably, to $\pi(C_v)$.

It remains to lower-bound the expected size of $\cup_v S''_v$. Say that a terminal s *survives* if

(i) it is initially rounded up, and hence included in a set S_v, and

(ii) it is not deleted because it or its matching terminal is contained in an overfull rectangle.

So what is the probability that a terminal s survives? We will let ρ_s denote the demand of the terminal pair associated with s, and $f_s \in [0,1]$ the weight assigned to it by the fractional optimum. First we consider the probability that s is contained in a *fixed* overfull $p \times q$ rectangle R, *given* that s has been rounded up. In order for this rectangle R to become overfull, it must be that a total demand of at least

$$\frac{1}{2}(p+q) - d_s \geq \frac{15}{32}(p+q)$$

was rounded up by the Raghavan–Thompson algorithm, associated with terminals in R other than s. But since the un-scaled fractional flow was feasible, the total fractional weight contained in R, after scaling down, is at most $2\mu(p+q)$. Thus, setting

$$y = \left(\frac{64e\mu}{15}\right)^{\frac{15}{32}}$$

and applying Corollary 2.5.2, the probability that the rectangle R becomes overfull after rounding, given that s is rounded up, is at most y^{p+q}.

Now, since s is contained in at most pq rectangles of dimensions $p \times q$, the probability that s is contained in any overfull rectangle after rounding, given that it is rounded up, is clearly bounded by the infinite sum

$$\sum_p \sum_q pqy^{p+q} = \frac{y^2}{(1-y)^4}.$$

s can also be deleted if its matching terminal is contained in an overfull rectangle, so the probability of s being deleted, after having been rounded up, is at most $\frac{2y^2}{(1-y)^4}$. By taking μ small enough, we can make this last expression a constant less than $\frac{1}{2}$.

Finally, the probability that s survives is equal to the probability that it is rounded up, which is μf_s, times the probability that it is not deleted after being rounded up, which is at least $\frac{1}{2}$. Thus, the expected size of $\cup_v S''_v$ is at least $\frac{1}{2}\mu f^*$, for a small enough constant value of μ. So by Markov's inequality, the probability that $\rho(\cup_v S''_v) \geq \frac{1}{4}\mu f^*$ is at least an absolute constant; and the lemma follows. ■

To turn the above lemma from a statement holding with constant probability to one with high probability, we repeat the randomized rounding $O(\log n)$ times and take the best routing obtained. A single run of randomized rounding fails if one of the following two bad events happens:

(1) one of the high capacity edges of the simulated network is used by too many selected paths, or

(2) the set $\cup_v S'_v$ selected by Lemma 6.3.1 contains too little demand.

The first event has inverse polynomially small probability, and the probability of the second is bounded by a constant, so the total probability of any bad event is bounded by a constant less than 1. This implies that out of the $O(\log n)$ tries, one must succeed with high probability.

This gives a constant-factor approximation for long (light) connections: if (s_i, t_i) is rounded up, and $s_i, t_i \in \cup_v S'_v$, then we concatenate the paths from s_i to $\pi(C_{\psi(s_i)})$ (given by Lemma 6.3.1) to $\pi(D_{\psi(s_i)})$ (given by the crossbar in $C^*_v \setminus C_v$) to $\pi(D_{\psi(t_i)})$ (given by the edges of the path in $\mathcal{N}(\frac{1}{2}\beta \log n)$, joined together with crossbars as in Lemma 6.2.5), and now symmetrically to $\pi(C_{\psi(t_i)})$ and to t_i.

We handle short connections as in Section 6.2.4. That is, we use a randomized algorithm to construct subsquares $X_u \subset Y_u$, with all Y_u disjoint, and only handle short connections both of whose ends lie in a single X_u. In this case, however, we now run the above algorithm recursively on each $G[Y_u]$. Call a connection "medium" if it is now a long connection in this recursive call, and "small" otherwise. Medium connections are treated as long connections in the recursion, and handled as described above. Small connections take place within clusters of size $O(\log\log n)$ and therefore can be simply solved to optimality by brute force. We can then take the largest realizable set we find among the long, medium, and small connections, obtaining a constant–factor approximation for light connections.

Finally, a word about heavy connections; i.e. those with $\rho_i > \frac{1}{16}$. For those that are long, we assign a demand of 1 to each, and run the above algorithm. In Lemma 6.3.1, all demands are the same, and we can therefore use the max-flow min-cut theorem directly, rather than the unsplittable flow theorem, 3.2.2. This would give us a constant factor approximation, if all demands were equal to 1. We are in fact overestimating each demand by up to a factor of 16; but since our constant–factor approximation is with respect to the fractional optimum, this results in a loss of only a factor of 16 in the approximation ratio. Short connections are still handled recursively.

Thus we have

Theorem 6.3.2 *There is a randomized (off-line)* UFP *algorithm in the two-dimensional mesh that produces a constant-factor approximation with high probability.*

6.4 Routing in Rounds

We now use the constant–factor UFP approximation just developed to provide an approximation algorithm for routing a set $\mathcal{T}$ of terminal pairs in the minimum number of *rounds of communication*, on the two–dimensional mesh. Thus, we are addressing the following question: what is the minimum size of a partition of $\mathcal{T}$ into realizable sets? We use $\chi(\mathcal{T})$ to denote this minimum.

The quantity $\chi(\mathcal{T})$ also arises in the analysis of routing algorithms for *all–optical networks* [1, 95, 9]. Here, one is given a graph G and set of terminal pairs $\mathcal{T}$ (with all demands equal to 1); for each pair $(s_i, t_i) \in \mathcal{T}$ one must choose a path in G and a *wavelength* to use for routing the path. Two paths with the same wavelength must be edge–disjoint; the goal is to minimize the number of wavelengths used. Clearly the minimum number of wavelengths possible is precisely $\chi(\mathcal{T})$.

Within the context of optical routing — so that all demands are 1 — approximations for $\chi(\mathcal{T})$ have been given by Raghavan and Upfal [95] and Aumann and Rabani [9] for restricted types of graphs. Raghavan and Upfal provide a $\frac{3}{2}$–approximation for $\chi(\mathcal{T})$ in trees, and a 2-approximation in cycles. They also show, using the methods discussed in Section 2.8, that $\chi(\mathcal{T}) = O(\log^2 n)$ for every permutation on a bounded–degree expander. Aumann and Rabani obtain the same bound for expanders, via different methods, and also provide an $O(\log^2 n)$–approximation for $\chi(\mathcal{T})$ in the two–dimensional mesh.

The notion of partitioning $\mathcal{T}$ into realizable sets also arises in certain VLSI problems; we will encounter this in Chapter 9.

Our approach is based on a direct application of the greedy set cover technique of (independently) Chvátal, Johnson, and Lovász [24, 53, 77]; this allows one to take an approximation algorithm for the MDP and obtain an approximation for routing in rounds with a performance guarantee that is only larger by a logarithmic factor. This reduction method was applied by Aumann and Rabani [9] to derive their $O(\log^2 n)$–approximation for $\chi(\mathcal{T})$ on the mesh from an $O(\log n)$–approximation for $\alpha(\mathcal{T})$ on the mesh. In the present case, our constant–factor approximation for the MDP in the two–dimensional mesh allows us to obtain an $O(\log n)$–approximation for $\chi(\mathcal{T})$ in the mesh, when all demands and capacities are 1. Finally, it is easy to extend the approximation for χ to handle a set of terminal pairs with arbitrary demands in $[0, 1]$, and edges of unit capacity. When the minimum demand of any pair of terminals is ρ_*, the approximation ratio we obtain is $O(\log n + \log \rho_*^{-1})$.

We present the framework in a fairly general setting, and then show how to apply it to the problem of approximating $\chi(\mathcal{T})$. Let U denote a finite set, and $\mathcal{F}$ a collection of "feasible" subsets of U that is closed with respect to inclusion, and which satisfies

$$\bigcup_{S \in \mathcal{F}} S = U.$$

For a subset $U' \subset U$, let $\mu(U')$ denote the maximum cardinality of a member of $\mathcal{F} \cap 2^{U'}$.

Suppose we are given a maximization algorithm A which works as follows: there is a number a so that on any set $U' \subset U$, it returns a set in $\mathcal{F} \cap 2^{U'}$ of cardinality at least $\frac{1}{a}\mu(U')$.

Our goal is to cover U by as few feasible sets as possible. For this purpose, we define the following algorithm A', which is simply the greedy set-cover algorithm, making use of the maximization algorithm A.

- Define $U_0 = U$.
- For $i = 0, 1, \ldots$
  - If $U_i = \phi$ then terminate.
  - Else run A on U_i, obtaining a set U_i'. Define $U_{i+1} = U_i \setminus U_i'$.

The size of the cover of U produced is simply the number of non–empty sets U_i'. Let H_m denote the m^{th} *harmonic number*, defined by $H_m = \sum_{i=1}^{m} \frac{1}{i}$. Given this definition we have the following result — it is simply the main result of [24, 53, 77], for the case in which the maximization algorithm A only produces a set whose cardinality is *approximately* maximum.

Theorem 6.4.1 *The size of the cover produced by A' is at most $aH_{\mu(U)}$ times more than the size of the smallest cover of U by sets of $\mathcal{F}$.*

Proof. Suppose that A' produces a cover with r sets. For each $u \in U$, we identify the unique i such that $u \in U_i'$, and we define the *cost* c_u of u to be $|U_i'|^{-1}$. Thus we have $\sum_{u \in U} c_u = r$. Now we claim that for any set $X \in \mathcal{F}$, we have $\sum_{u \in X} c_u \leq aH_{|X|}$; the theorem follows directly from this.

To verify the claim, consider any set $X \in \mathcal{F}$, and suppose that it contains a subset $X' = \{x_1, \ldots, x_{at+1}\}$, such that each $x_i \in X'$ has cost at least $\frac{1}{t}$. Let j be the maximum index such that $X' \subseteq U_j$; hence there is some $x_i \in X' \cap U_j'$. Thus, $\left|U_j'\right| = c_{x_i}^{-1} \leq t$. But X' is a set of cardinality at least $at + 1$ that is contained in $\mathcal{F}$; and it is a subset of U_j — this contradicts the performance guarantee of A. From this contradiction, it follows that for each t, X' contains at most at elements of cost at least $\frac{1}{t}$; and hence $\sum_{u \in X} c_u \leq aH_{|X|}$. ∎

We can easily specialize this to the present setting as follows. Let G denote the $n \times n$ two–dimensional mesh, with every edge having unit capacity, and let $\mathcal{T}$ denote a set of terminal pairs each of demand 1. Let $\mathcal{F}$ denote the collection of realizable subsets of $\mathcal{T}$, and A the constant–factor approximation for the MDP develop in Chapter 6. Then defining the algorithm A' as above, we have

Corollary 6.4.2 *There is an $O(\log n)$–approximation for $\chi(\mathcal{T})$ in G, when all terminals have demand* 1.

The above set–cover algorithm A' can be implemented without any modification in the on–line setting, provided that the maximization algorithm A is on–line. That is, we define an infinite number of copies of the on–line maximization algorithm A; we label these $A_1, A_2, \ldots$. As each terminal pair arrives, it is evaluated by the algorithms $A_1, A_2, \ldots$ in succession, until one of them accepts it. The set of terminal pairs accepted by algorithm A_i constitutes the i^{th} round. Using the on–line algorithm of Theorem 6.2.8, together with Theorem 6.4.1, we have

Corollary 6.4.3 *There is an on–line $O(\log^2 n)$–approximation for $\chi(\mathcal{T})$ in G, when all terminals have demand* 1.

We now turn to the case of a set $\mathcal{T}$ of terminal pairs, with arbitrary demands in the interval $[0, 1]$, again on a unit–capacity two–dimensional mesh G. Let ρ_* denote the minimum demand associated with any pair of terminals in $\mathcal{T}$.

Theorem 6.4.4 *There is an $O(\log n + \log \rho_*^{-1})$–approximation for $\chi(\mathcal{T})$ in G.*

Proof. Let A denote the UFP maximization algorithm provided by Theorem 6.3.2. We use A to define a partitioning algorithm A' exactly as above. However, in analyzing A', we run into the following technical complication: Theorem 6.4.1 deals only with unweighted cardinalities, whereas the algorithm A approximately maximizes the total *demand* routed. There are a number of straightforward ways to get around this difficulty; we handle it by creating an artificial set cover problem in which each demand is "discretized" in units of ρ_*.

For a terminal pair (s_i, t_i), with demand ρ_i, let W_i denote a set of $q_i = \lfloor \rho_i \rho_*^{-1} \rfloor$ dummy elements $w_{i1}, \ldots, w_{iq_i}$. For any set $\mathcal{T}' \subset \mathcal{T}$, define

$$W(\mathcal{T}') = \bigcup_{i:(s_i,t_i)\in\mathcal{T}'} W_i,$$

and $U = W(\mathcal{T})$. The set $\mathcal{F}$ of feasible sets will consist of all subsets of sets of the form $W(\mathcal{T}')$, where $\mathcal{T}'$ is realizable in G. Now we observe that the algorithm A has the following property: on any set $\mathcal{T}' \subset \mathcal{T}$ of terminal pairs, it produces a realizable set $\mathcal{T}'' \subset \mathcal{T}'$ such that $|W(\mathcal{T}'')|$ is within a constant factor of $\mu(W(\mathcal{T}'))$. Since the maximum demand that can be routed in a single round in G is at most $4n^2$, the maximum cardinality of any set in $\mathcal{F}$ is at most $4n^2\rho_*^{-1}$, and hence the result follows from Theorem 6.4.1. ■

Chapter 7

Generalizing the Mesh

What have you urged that I cannot reprove?
The path is smooth that leadeth on to danger . . .
— *William Shakespeare,* Venus and Adonis.

We remarked at the beginning of the previous chapter that the approximation algorithms given there rely crucially, in places, on the precise row/column structure of the two–dimensional mesh. This is true for previous approximation algorithms as well [12, 9]. Our goal in this chapter is to extend our algorithms to a more general class of graphs, defined below.

There are a number of motivations for generalizing the algorithms in this way. Dealing with a broader class of graphs results in algorithms that are more robust, in that they are not sensitive to small variations in the structure of the underlying graph. This could be of value in the context of network routing, where the underlying network may often have a "mesh-like" topology, without having the completely regular structure of the mesh itself. Thus, designing algorithms in this setting helps to isolate the basic techniques involved, and suggests general approaches for constructing disjoint paths in planar graphs.

In our case, we see that the main respect in which we used the "fine structure" of the mesh was in the construction of the simulated network $\mathcal{N}$, using a decomposition into blocks and crossbar structures within these blocks. Once the simulated network was in place, the routing algorithm was much less dependent on the nature of the underlying graph. This suggests that the approach of the previous chapter should be applicable to any type of graph in which one can construct an analogous simulated network; and this is the direction we investigate here.

Our main result in this chapter is a constant–factor approximation for the UFP in the class of *densely embedded graphs*, which we define in the following section. We also obtain an on–line $O(\log n)$–approximation. Finally, we note that the general analysis of Section 6.4 is applicable here as well, and provides an approximation algorithm for the problem of routing in rounds.

7.1 Definitions and Properties

Following the definition of expansion in Section 2.8, let us make precise the type of "expansion" exhibited by the two–dimensional mesh.

Definition 7.1.1 *A graph H is an α-semi-expander if for every $X \subset V(H)$ for which $|X| \leq \frac{1}{2}|V(H)|$, we have $|\delta(X)| \geq \alpha\sqrt{|X|}$.*

Since our goal is to generalize the two-dimensional mesh, let us note the following properties of the mesh.

(i) It is a planar graph with bounded degree, and (aside from one "exceptional face") it is Eulerian and has bounded face size.

(ii) It is an α-semi-expander, for a constant $\alpha > 0$ based on the ratio of the two side lengths of the mesh.

(iii) Square sub-meshes of the mesh satisfy (i) and (ii).

In the arguments to follow, it is quite cumbersome — though not technically difficult — to deal with "exceptional faces" of the type in (i). Thus, for the sake of presentation, we will work in this chapter with a class of Eulerian graphs in which *all* faces have bounded size. While this class therefore will include the $n \times n$ torus but not the mesh, it is straightforward to adapt our algorithms and their analysis to handle graphs with an exceptional face, which may be of arbitrary size and contain odd-degree vertices. In this way, we can obtain a class of graphs that includes the mesh as well.

Our class of graphs is defined to satisfy analogues of properties (i), (ii), and (iii) locally.

Definition 7.1.2 *A graph $G = (V, E)$ is* densely embedded *with parameters α, λ, Δ, and ℓ if:*

(i) G is strongly embedded on a compact orientable surface Σ, it has maximum degree Δ, and each face is bounded by at most ℓ edges.

(ii) For each $r \leq \lambda \log n$ and each $v \in V$, the drawing of $G[B_r(v)]$ is contained in a Σ-disc.

(iii) For each $r \leq \lambda \log n$ and each $v \in V$, the graph $G[B_r(v)]$ is an α-semi-expander.

Thus, for the remainder of this chapter, we will assume that G is a simple Eulerian graph that is *densely embedded* on a surface Σ with parameters α, λ, Δ, and ℓ. It is worth noting that Definition 7.1.2 is strictly local in nature, and places no restrictions on the global structure of the graph G.

Some Basic Properties

We now show that our definition implies G has some additional mesh-like properties. First of all, for any $v \in V$ and $r \leq \lambda \log n$, the fact that $G[B_r(v)]$ is a bounded-degree semi-expander implies that the set $B_r(v)$ has size at least quadratic in r; by also using the planarity of $G[B_r(v)]$, one can show an analogous upper bound. We summarize this as follows.

Lemma 7.1.3 *There are constants $\bar{\alpha}$ and β depending only on α and Δ such that the following holds. For each $r \leq \lambda \log n$ and each $v \in V$, we have $\bar{\alpha} r^2 \leq |B_r(v)| \leq \beta r^2$.*

Proof. Fix $r \leq \lambda \log n$ and $v \in V$, and let $S = B_r(v)$. To see the lower bound, note that for any $i \leq r$, if $x_i = |B_i(v)|$, then by the semi-expansion of H we have

$$x_i \geq x_{i-1} + \frac{\alpha}{\Delta - 1}\sqrt{x_{i-1}}.$$

For at least $\alpha\sqrt{x_{i-1}}$ edges leave $B_{i-1}(v)$, and at most $\Delta - 1$ are incident to any one vertex. Let $\nu = \frac{\alpha}{\Delta-1}$; now one verifies by induction that $x_i \geq \frac{1}{16}\nu^2 i^2$:

$$\begin{aligned} x_i &\geq \frac{1}{16}\nu^2(i-1)^2 + \frac{1}{4}\nu^2(i-1) \\ &= \frac{1}{16}\nu^2(i-1)(i+3) \\ &\geq \frac{1}{16}\nu^2 i^2. \end{aligned}$$

To see the upper bound, we observe that $G[S]$ is planar and has diameter at most $2r$. Let $n = |S|$. By the Lipton-Tarjan planar separator theorem [75], there is a set of at most $4r + 1$

vertices whose removal breaks H into components each of size at most $\frac{2}{3}n$. Let X be a union of these components of size between $\frac{1}{3}n$ and $\frac{1}{2}n$. Then

$$\begin{aligned}\frac{\alpha\sqrt{n}}{\sqrt{3}} &\leq \alpha\sqrt{|X|} \\ &\leq |\delta(X)| \\ &\leq (\Delta-1)(4r+1)\end{aligned}$$

from which the result follows. ■

We introduce some additional notation. If C is a connected subset of $G \setminus X$, we use $\Gamma(X, C)$ to denote the (unique) connected component of $G \setminus X$ containing C. The set of vertices in $\pi(X)$ which have a neighbor in $\Gamma(X, C)$ will be called the *segment of* $\pi(X)$ *bordering* C and denoted $\sigma(X, C)$. We say that a set $X \subset V$ is *simple* if $G \setminus X$ is connected.

The following two facts are quite useful; the first essentially relates the size of the "perimeter" of a set $B_r(v)$ ($r \leq \lambda \log n$) to its radius.

Lemma 7.1.4 *Let $c > b \geq 1$ and r a positive integer be such that $cr < \lambda \log n$. Then for some r' between br and cr, we have $|\pi(B_{r'}(v))| \leq \beta \cdot \frac{c^2}{c-b} \cdot r$.*

Proof. Since $\pi(B_{r'}(v)) \subset \{u : d(v, u) = r'\}$, the sets $\pi(B_{br}(v)), \pi(B_{br+1}(v)), \ldots, \pi(B_{cr}(v))$ are all disjoint and contained in $B_{cr}(v)$. Since $|B_{cr}(v)| \leq \beta c^2 r^2$, one of these sets has size at most $\beta \cdot \frac{c^2}{c-b} \cdot r$. ■

Using this, we show that we can extend any small enough set U to a simple set with at most a constant-factor increase in its radius.

Lemma 7.1.5 *There is a constant ξ such that the following holds. Let $U \subset B_r(v)$, where $r \leq \frac{1}{\xi}\lambda \log n$. Then there is a component Γ of $G \setminus U$ and a planar simple set U' such that $U \subset U' \subset B_{\xi r}(v)$, $G \setminus U' = \Gamma$, and $\sigma(U, \Gamma) = \sigma(U', \Gamma)$.*

Proof. Choose r' between r and $2r$ so that $|\pi(B_{r'}(v))| \leq 4\beta r$. Let $U_0 = B_{r'}(v)$, and $G \setminus U_0$ have components $\Gamma_1, \ldots, \Gamma_p$.

Now set $s = 8\bar{\alpha}^{-1/2}\alpha^{-1}\beta\Delta$ and $\xi = 2s + 2$. We claim that all but one of the components Γ_i are contained in $B_{\xi r}(v)$. For suppose not; then for $i \neq j$ there are $w \in \Gamma_i$ and $w' \in \Gamma_j$ such that

w and w' are each at distance s from U_0, $B_s(w) \subset \Gamma_i$, and $B_s(w') \subset \Gamma_j$. Now consider the edge cut of size at most $4\beta\Delta r$ formed by $\delta(U_0)$; one of the two spheres $B_s(w)$ and $B_s(w')$, say the latter, is contained in a small component of this cut in $G[B_{\xi r}(v)]$. But then the semi-expansion of $G[B_{\xi r}(v)]$ requires that $4\beta\Delta r \geq \alpha\sqrt{|B_s(w')|}$, which is a contradiction since by Lemma 7.1.3 we have $|B_s(w')| \geq \bar{\alpha}s^2$.

So for some i, only Γ_i is not contained in $B_{\xi r}(v)$. Now let $\Gamma'_1, \ldots, \Gamma'_q$ denote the components of $G \setminus U$; so Γ_i is contained in one of these, say Γ'_1, and $\Gamma'_2, \ldots, \Gamma'_q$ are all contained in $B_{\xi r}(v)$. Now define

$$U' = U \cup \bigcup_{j>1} \Gamma'_j \subseteq B_{\xi r}(v).$$

In particular, U' is planar since $\xi r \leq \lambda \log n$, and it is simple since $G \setminus U'$ has only the component Γ'_1. Thus U' satisfies the conditions of the lemma. ■

Finally, we show a general property of planar graphs H with small face size: if the distance between two nodes in H is large, than the value of any edge cut which contains both in the same segment of its boundary must also be relatively large.

Lemma 7.1.6 *Let H be a planar graph, with distinguished faces $\Phi_1, \ldots, \Phi_r$ bounded by cycles $Q_1, \ldots, Q_r$ respectively. Suppose that all faces other than $\Phi_1, \ldots, \Phi_r$ are bounded by at most ℓ edges, and for a constant d' and all $i \neq j$ we have $d(Q_i, Q_j) \geq d'$.*

Let $U \subset V(H)$ and $v, w \in \sigma(U, C)$ for some component C of $G \setminus U$. Then

$$|\delta(U)| \geq \min(\ell^{-1}d', \ell^{-1}d(v, w)).$$

Proof. Let $S = \sigma(U, C) \subset U$. In the graph $H[U]$, S lies on a single facial cycle Q. Traversing Q in a clockwise direction starting at v, we encounter faces $R_1, \ldots, R_p$ whose boundaries contain vertices both of U and of $H \setminus U$.

Suppose that among the $\{R_i\}$ there are two distinct large faces Φ_m and $\Phi_{m'}$. Choose such a pair for which $R_a = \Phi_m$, $R_b = \Phi_{m'}$, and $R_c \notin \{\Phi_i\}$ for $a < c < b$. Let P denote the corresponding maximal subpath of Q whose internal vertices are incident only to faces R_c, for $a < c < b$. Then among every ℓ consecutive vertices of P, there must be one incident to an edge in $\delta(U)$; since $|P| \geq d'$ by the hypotheses of the lemma, this implies the claimed bound.

Otherwise, there is a single large face Φ_m among the $\{R_i\}$; note that Φ_m may appear several times on the traversal of Q. Now there are two sub-paths of Q from v to w, which we denote P_0 and P_1. Since v, w border the same component of $H \setminus U$, the face Φ_m does not appear in a traversal of one of P_0 or P_1 — suppose it is P_0. So as above, among every ℓ consecutive vertices of P_0, there must be one incident to an edge in $\delta(U)$; since we have $|P_0| \geq d(v, w)$, the result follows. ■

Related Classes of Graphs

In this section, we show a natural construction which produces densely embedded graphs. The material here is independent of the rest of the chapter.

We say that a graph H is *geometrically well–formed* if it can be embedded on the unit sphere in such a way that every face has, geometrically, about the same size. That is, there is a constant c so that for every pair of faces Φ_1 and Φ_2 in the embedding of H, one can dilate Φ_1 by a factor of c from some point in its interior so that it contains a congruent copy of Φ_2.

One can show that every geometrically well-formed graph is densely embedded; we will prove this by way of a more general definition that applies to arbitrary surfaces. Let Σ be a compact orientable surface, embedded in $\mathbf{R}^3$. For $x \in \Sigma$, we let $B'_d(x)$ denote the set of all points of Σ whose distance from x (as measured on Σ) is at most d.

Definition 7.1.7 *A set $X \subset \Sigma$ is (γ_0, γ_1)-*flat *for some positive constants γ_0, γ_1, if there is a Σ-disc D such that*

(i) $X \subseteq D$.

(ii) For all points $x \in X$ and $s \geq 0$ such that $B'_s(x) \subset X$, the surface area of $B'_s(x)$ is at least $\gamma_0 s^2$ and at most $\gamma_1 s^2$.

(iii) For all Σ-discs D' such that $D' \subseteq X$, if the boundary of D' has length s, then the surface area of D' is at most $\gamma_1 s^2$.

*We say that Σ is (r, γ_0, γ_1)-*locally flat *if it is orientable, and for all $x \in \Sigma$ the set $B'_r(x)$ is (γ_0, γ_1)-flat.*

Now we say that a graph is *locally well-formed* if it is drawn on a locally flat surface, and each face has, geometrically, about the same (small) size.

Definition 7.1.8 *A graph H drawn on Σ is* locally well-formed *with parameters $\Delta, \ell, \gamma_0, \gamma_1, \rho_0, \rho_1$ if there exists an $r > 0$ so that*

(i) it has maximum degree Δ,

(ii) The maximum number of edges on a face in the drawing of H is ℓ, and

(iii) Σ is $(r \log n, \gamma_0, \gamma_1)$-locally flat,

(iii) for each face Φ of G there is an $x \in \Sigma$ so that $B'_{\rho_0 r}(x) \subset \Phi \subset B'_{\rho_1 r}(x)$.

Note that a geometrically well-formed graph, as defined above, is locally well-formed for an appropriate choice of parameters. We now want to show that every locally well-formed graph is densely embedded. To show this, the following routine lemma is useful: in Definition 7.1.1 it is enough to require semi-expansion for cuts that produce only two components.

Lemma 7.1.9 *H is an α-semi-expander if and only if the condition of Definition 7.1.1 holds for all sets X for which $H[X]$ and $H \setminus X$ are both connected.*

Proof. We proceed by induction on the number of connected components of $H[X]$ and $H \setminus X$. Assume $H[X]$ is not connected, and let $\Gamma_1, \ldots, \Gamma_p$ be components. Then each of the sets $\Gamma_1, \ldots, \Gamma_p$ must satisfy Definition 7.1.1 by the induction hypothesis. From this we get

$$|\delta(X)| = \sum_i |\delta(\Gamma_i)| \geq \alpha \sum_i \sqrt{|\Gamma_i|} \geq \alpha \sqrt{|\cup_i \Gamma_i|} = \alpha \sqrt{|X|}.$$

If H connected but $H \setminus X$ is not, and each connected component has size at most $\frac{1}{2}|V(H)|$, then the above argument applies to the components of $H \setminus X$. Otherwise, removing the edges between H and the single large component of $H \setminus X$ produces a cut both of whose sides are connected; the semi–expansion condition holds in this case by assumption. ■

Proposition 7.1.10 *If H is locally well-formed with parameters $\Delta, \ell, \gamma_0, \gamma_1, \rho_0, \rho_1$, then there are positive constants α and λ such that H is densely embedded with parameters α, λ, Δ, and ℓ.*

Proof. Let H be locally well-formed with the given parameters, and with an associated $r > 0$ as in Definition 7.1.8. Then for any $v \in V$, if $s \leq \rho_1^{-1} \log n$, the set $B_s(v)$ is contained in $B'_{r \log n}(v)$ and hence in a Σ-disc. Now let $X \subset B_s(v)$; we wish to show that it satisfies the semi-expansion requirement in $H[B_s(v)]$. By Lemma 7.1.9, we may assume that both $H[X]$ and $H[B_s(v) \setminus X]$ are connected. Thus $\delta(X)$ lies on a single face of $H[X]$. Let $q = |\delta(X)|$; then there is a closed curve L on Σ of length at most $\rho_1 r q$ that bounds a Σ-disc containing X. Thus, X is contained in a disc of area at most $\gamma_1 \rho_1^2 r^2 q^2$. But each face in $H[X]$ has area at least $\gamma_0 \rho_0^2 r^2$, so X has at most $\gamma_1 \rho_1^2 \gamma_0^{-1} \rho_0^{-2} q^2$ faces, and hence at most ℓ times this many vertices. ∎

In their series of papers proving, among other things, that the disjoint paths problem for a fixed number of terminal pairs is solvable in polynomial time [102], Robertson and Seymour make use of another notion of "denseness" of surface embeddings — namely *representativity*. It turns out that our definition of densely embedded graphs could also have been expressed in these terms. We say that a drawing of a graph G on Σ is *c-representative* [99, 101] if any non-null-homotopic closed curve on Σ meets the drawing of G at least c times. In this terminology, we could have replaced the condition that each $G[B_r(v)]$ ($r \leq \lambda \log n$) be contained in a Σ-disc by the condition that the drawing of G be $\Omega(\log n)$-representative. More precisely,

Proposition 7.1.11 *If G satisfies parts (i) and (iii) of Definition 7.1.2, and the drawing of G is $(\lambda \log n)$-representative, then there is a constant λ' such that G is densely embedded with parameters α, λ', Δ, and ℓ. Conversely, if G is densely embedded with parameters α, λ, Δ, and ℓ, then there is a constant λ' such that the drawing of G is $(\lambda' \log n)$-representative.*

Proof. The converse statement is easier. If G is densely embedded with parameters α, λ, Δ, and ℓ, then any closed curve $\mathcal{R}$ on Σ meeting G at fewer than $\ell^{-1}\lambda \log n$ vertices meets it only at vertices contained in $B_{\lambda \log n}(v)$ for some $v \in V$. Thus $\mathcal{R}$ is contained in a Σ-disc and is null-homotopic.

Now suppose G satisfies parts (i) and (iii) of Definition 7.1.2, and the drawing of G is $(\lambda \log n)$-representative. We must show that for some λ', every $G[B_{\lambda' \log n}(v)]$ is drawn in a Σ-disc. Choose $\lambda' < \frac{1}{4}\lambda$ and let $U = B_{\lambda' \log n}(v)$ for some $v \in V$. We claim that every simple cycle of $G[U]$ is null-homotopic in Σ. For suppose not, and choose the shortest non-null-homotopic cycle Q contained in $G[U]$. Say for simplicity that Q contains an even number of vertices, $v_0, \ldots, v_k, \ldots, v_{2k} = v_0$, and let Q_0 and Q_1 denote the two sub-paths of Q with ends equal to v_0 and v_k. Now suppose there were some path P in $G[U]$ with ends equal to v_0 and v_k of length less than k; then one of $Q_0 \cup P$ or $Q_1 \cup P$ would contain a non-null-homotopic simple cycle of $G[U]$ of length less than $2k$, contradicting our choice of Q. Thus $k \leq 2\lambda' \log n$, and so $|Q| \leq 4\lambda' \log n < \lambda \log n$. Now since G is strongly embedded, there is a simple closed curve $\mathcal{R}$ on Σ, meeting G at precisely the vertices of Q, that is homotopic to Q in Σ; but since $\mathcal{R}$ meets G fewer than $\lambda \log n$ times, it is null-homotopic in Σ. This contradicts our assumption that Q is non-null-homotopic.

Thus $G[U]$ contains only null-homotopic simple cycles. By Theorems (11.2) and (11.10) of [99], this implies that $G[U]$ is contained in a Σ-disc. ■

7.2 Smooth Cycles

In the sections to follow, we will construct a simulated network in an arbitrary densely embedded Eulerian graph, following roughly the outline of the previous chapter. We will be working with connected, planar subsets of G; and a property that will prove crucial to the construction is that their outer cycles be *smooth*. By this we mean that the distance between two vertices on the outer cycle of such a set, as measured along this outer cycle, is not significantly greater than the distance between these vertices as measured through the interior of the set.

To be more concrete, let H denote an arbitrary graph, and Q a simple cycle of H. For $u, v \in Q$, let $d_Q(u, v)$ denote the shortest-path distance from u to v on Q — that is, the length of the shorter of the two u-v paths on Q.

Definition 7.2.1 *We say that Q is ε-*smooth *if for all $u, v \in Q$ we have $\varepsilon d_Q(u, v) \leq d(u, v)$.*

Definition 7.2.2 *If U and W are two subsets of $V(H)$, we say that U is ε'-close to W if for each $u \in U$ there is a $w \in W$ such that $d(u, w) \leq \varepsilon'|W|$.*

What we are after is a result of the following form. Let H be embedded on some compact surface Σ_1, and let Q be a non-null-homotopic cycle of H. We want to find a cycle Q', also non–null–homotopic, such that (i) Q' is (sufficiently) smooth, and (ii) Q' is (sufficiently) close to Q. These two conditions are in tension — to satisfy the first, one could simply take a shortest non-null-homotopic cycle in H; any such cycle is 1-smooth. But such a cycle might not be close Q. To satisfy the second, one could take Q itself; but Q might not be sufficiently smooth.

The following result shows that one can always find a cycle Q' that satisfies parametrized versions of conditions (i) and (ii).

Theorem 7.2.3 *For each $\varepsilon > 0$ the following holds. Let Σ_1 be a compact surface (possibly with boundary), H a graph embedded on Σ_1, and Q a simple cycle of H that is non-null-homotopic on Σ_1. Then in polynomial time one can find an $\frac{\varepsilon}{1+\varepsilon}$-smooth simple cycle Q' such that*

(i) $|Q'| \leq |Q|$,

(ii) Q' is ε-close to Q, and

(iii) Q' is also non-null-homotopic on Σ_1.

Proof. For $u, v \in Q$, let $[u, v]_Q$ denote the shorter of the two u-v paths contained in Q (ties broken arbitrarily), and let $\bar{\varepsilon} = \frac{\varepsilon}{1+\varepsilon}$. If Q is not $\bar{\varepsilon}$-smooth, then there are $u, v \in Q$ such that

$$\bar{\varepsilon} d_Q(u, v) > d(u, v). \tag{7.1}$$

Moreover, we can efficiently find such a u and v so that there is a shortest u-v path P_{uv} in H that is vertex-disjoint from Q (for example, the pair u, v satisfying (7.1) for which $|P_{uv}|$ is minimum).

Now one of the two simple cycles $[u, v]_Q \cup P_{uv}$ and $(Q \setminus [u, v]_Q) \cup P_{uv}$ is not null-homotopic on Σ_1; and each is shorter than Q. We thus update Q, replacing it with the cycle from among these two that is not null-homotopic. We will say that the portion of Q that was deleted when adding P_{uv} is "marked," and the remaining portion of Q is "unmarked."

We now iterate this process of "slicing off" parts of Q using short paths through H. Since the length of the cycle decreases with each iteration, this process must terminate in a cycle Q' that is $\bar{\varepsilon}$-smooth. Moreover, each iteration maintains the invariant that the current cycle is non-null-homotopic on Σ_1. Thus, we only have to verify that the final cycle is ε-close to Q.

This is clearly true after the first iteration: since $|P_{uv}| < \bar{\varepsilon} d_Q(u, v) < \varepsilon|Q|$, every vertex on the updated cycle can reach a vertex of Q by a path of length at most $\varepsilon|Q|$. Now, let Q_i denote the cycle obtained after i iterations of slicing off. As long as portions of Q remain that are still unmarked, we say that we are in the "first phase"; other phases will be defined below. In the first phase, Q_i consists of alternating intervals $Q_{i1}, P_{i1}, Q_{i2}, \ldots, Q_{ir}, P_{ir}$, where $Q_{ij} \subset Q$ is unmarked, and the interval Q'_{ij} of Q lying between Q_{ij} and $Q_{i,j+1}$ is marked (and replaced by P_{ij}). We show by induction on the number of iterations that $|P_{ij}| \leq \bar{\varepsilon}\left|Q'_{ij}\right|$ — as was true after the first iteration.

This is done by the following case analysis. In the $(i+1)^{\text{st}}$ iteration, we find a new path; there are three cases to consider.

1. One end of P lies on Q_{ij} and the other on Q_{ik}, where possibly $j = k$. Then the property clearly continues to hold, since $|P|$ is at most $\bar{\varepsilon}$ times the number of current cycle vertices cut off, which is in turn at most the number of original vertices of Q between the endpoints of P.

2. One end of P lies on P_{ij} and the other on Q_{ik} (so P_{ij} is lengthened). Suppose that the amount of original cycle cut off *in addition to* Q'_{ij} is equal to x, and the amount of P_{ij} that is cut off by P is y. Then if $P_{i+1,j}$ denotes P_{ij} after this iteration, we have

$$\begin{aligned}
|P_{ij}| &\leq \bar{\varepsilon}\left|Q'_{ij}\right| \\
|P| &\leq \bar{\varepsilon}(x+y) \\
|P_{i+1,j}| &= |P_{ij}| + |P| - y \\
&\leq \bar{\varepsilon}(\left|Q'_{ij}\right| + x + y) - y \\
&\leq \bar{\varepsilon}(\left|Q'_{ij}\right| + x)
\end{aligned}$$

3. One end of P lies on P_{ij} and the other lies on P_{ik} (so P glues some of the new paths together). There are two subcases.

   (i) $j = k$. Then $|P_{ij}|$ goes down while $|Q_{ij}|$ is not affected, so the property still holds.

   (ii) $j \neq k$. Again suppose that the amount of original cycle cut off in addition to Q'_{ij} and Q'_{ik} is equal to x, the amount of P_{ij} cut off by P is y, the amount of P_{ik} cut off by P is z, and the new interval is denoted $P_{i+1,j}$. Then

$$\begin{aligned}
|P_{ij}| &\leq \bar{\varepsilon}\left|Q'_{ij}\right| \\
|P_{ik}| &\leq \bar{\varepsilon}|Q'_{ik}| \\
|P| &\leq \bar{\varepsilon}(x+y+z) \\
|P_{i+1,j}| &= |P_{ij}| + |P| + |P_{ik}| - y - z \\
&\leq \bar{\varepsilon}(\left|Q'_{ij}\right| + x + y + z + |Q'_{ik}|) - y - z
\end{aligned}$$

$$\leq \quad \bar{\varepsilon}(\left|Q'_{ij}\right| + x + |Q'_{ik}|)$$

If the iterations come to an end before the end of the first phase, then indeed Q' is ε-close to Q — any vertex on P_{ij} can reach Q by a path of length at most $\bar{\varepsilon}\left|Q'_{ij}\right| \leq \bar{\varepsilon}|Q|$. Otherwise, consider the iteration in which the first phase comes to an end. By analogous arguments, we obtain a cycle Q^1 such that $|Q^1| \leq \bar{\varepsilon}|Q|$ and every vertex on Q^1 can reach Q by a path of length at most $\bar{\varepsilon}|Q|$.

We say that all vertices of Q^1 are initially "unmarked." Each phase now proceeds exactly like the previous one, except that it begins with a cycle whose length has been reduced by at least a factor of $\bar{\varepsilon}$. Thus when the process terminates, all vertices on Q' will be able to reach Q by a path of length at most

$$|Q| \cdot \sum_{i=1}^{\infty} \bar{\varepsilon}^i = \varepsilon|Q|.$$

Thus Q' is ε-close to Q. ■

7.3 Clusters and Enclosures

We are now ready to begin defining the simulated network. One encounters a number of difficulties in extending the construction of Chapter 6 to densely embedded graphs in general. Some of these are easily taken care of — for example, we cannot define "subsquares" of G anymore; but we can use balls of the form $B_r(v)$ instead, and we have seen above that these behave in much the same way. We similarly may choose a maximal set of mutually distant balls and grow enclosures around them. Two major problems are the following. (1) We used the natural crossbars inside a mesh for routing; do these enclosures have similar crossbars inside them? (2) How do we join the clusters together into a simulated network?

To build crossbars inside the enclosures, we make use of a construction based on the Okamura-Seymour theorem [84], which we describe in this section. To define the high-capacity simulated network $\mathcal{N}$, we want to grow the enclosures out until they touch. However, at this point their boundaries might not be "smooth" enough to allow us to build crossbars inside them; additionally, there is no reason for enclosures that do touch to have $\Omega(\log n)$ edges in their common boundary.

Nevertheless, it is still possible to build a simulated network $\mathcal{N}$, as follows. We grow enclosures that have smooth boundaries, and are large enough that they contain large crossbars, but we keep them mutually distant from one another. Then we define the notion of a *Voronoi partition* of G to allow us to determine which clusters are "neighbors"; we define the simulated network $\mathcal{N}$ by putting $\Omega(\log n)$ parallel edges between neighboring clusters (whether or not they have that many edges in the common boundary of their Voronoi regions).

We show that the collection of these parallel edges "represents" the graph G well enough that it can be used as the network $\mathcal{N}$. In particular, we need to show that all paths accepted by the simulated network can be routed in G. For this we make use of a theorem of Schrijver [104]; we show that there exist $\Omega(\log n)$ paths in G between the neighboring enclosures, such that all paths between all pairs are mutually disjoint.

We make no attempt to optimize constants here. Set $\lambda_0 = \lambda$, and choose positive constants $\lambda_1, \lambda_2, \ldots$ so that $\lambda_{j+1} \ll \lambda_j$ (the exact relationship between these constants is easy to determine from the analysis below). A connection (s_i, t_i) is *short* if $d(s_i, t_i) \leq \lambda_2 \log n$ and *long* otherwise. As before, we handle long and short connections separately; for now we concentrate on preprocessing the graph as described above for handling long connections.

To start, we wish to choose a maximal set of mutually distant vertices around which to grow clusters. Let G^r denote the graph obtained from G by joining u and v if $d(u, v) \leq r$. We first run Luby's randomized maximal independent set algorithm [79] in $G^{\lambda_3 \log n}$.

Let M denote the resulting MIS. For any $x \in V$, some vertex within $\lambda_5 \log n$ of it will enter M on the first iteration if the largest number chosen in $B_{2\lambda_3 \log n}(x)$ is chosen by a vertex in $B_{\lambda_5 \log n}(x)$. This happens with constant probability, by Lemma 7.1.3. Moreover, if $d(x, y) \geq \lambda_2 \log n$, then these events are independent for x and y. Thus,

Lemma 7.3.1 *Let $x, y \in V$ be such that $d(x, y) \geq \lambda_2 \log n$. Then with constant probability there are $u, v \in M$ such that $d(x, u) \leq \lambda_5 \log n$ and $d(y, v) \leq \lambda_5 \log n$.*

We use the following procedure to build the clusters and the enclosures around each node v in M. Let $K_v = B_{\lambda_5 \log n}(v)$.

(i) By Lemma 7.1.4, we can choose a radius r between $2\lambda_5 \log n$ and $3\lambda_5 \log n$ so that

$$|\pi(B_r(v))| \leq 9\beta\lambda_5 \log n.$$

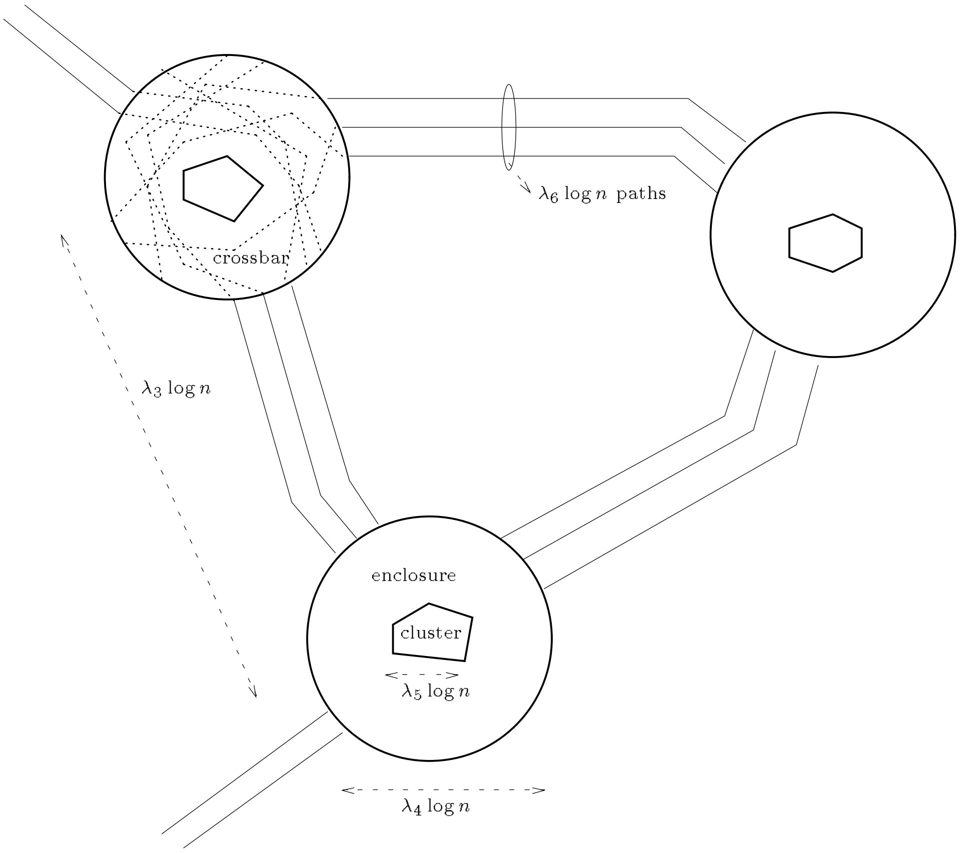

Figure 7-1: Building the simulated network

Set $C_v = B_r(v)$.

(ii) Now extend C_v to a simple set as in Lemma 7.1.5; by choosing the λ_i appropriately, we can ensure that $\lambda_3 > 2\xi\lambda_5$, and hence no C_v will be grown enough by this process that it overlaps any other.

(iii) We now apply the ε-smoothing algorithm of Theorem 7.2.3 to the facial cycle Q_v of $G[C_v]$ containing $\pi(C_v)$. Here

$$H_v = G[B_{\lambda_3 \log n}(v) \setminus K_v^o]$$

plays the role of H, and the cylinder formed by removing the portions of Σ on which $G[K_v^o]$ and $G \setminus B_{\lambda_3 \log n}(v)$ are drawn plays the role of Σ_1. Now for a constant ε, the resulting cycle Q'_v is ε-smooth in this subgraph H_v of G, and it is also $\frac{\varepsilon}{1-\varepsilon}$-close to Q_v.

We choose

$$\varepsilon < \frac{1}{18\beta + 1}.$$

All vertices on the resulting smooth cycle Q'_v will thus be within a distance of

$$\frac{1}{18\beta} \cdot |Q_v| \leq \frac{1}{2}\lambda_5 \log n$$

of Q_v. Hence Q'_v encloses the set $B_{\frac{3}{2}\lambda_5 \log n} \supset K_v$, and is contained in the set $B_{\frac{7}{2}\lambda_5 \log n}$. It follows that any path with both ends on Q'_v that leaves the subgraph H_v must have length at least

$$\lambda_5 \log n \geq \frac{1}{9\beta}|Q_v| \geq \frac{1}{9\beta}|Q'_v|,$$

and hence Q'_v is ε-smooth in the graph G (and not just in the subgraph H).

The smooth cycle Q'_v encloses a set S of vertices containing K_v. Update C_v to be this set S.

We now grow an *enclosure* $D_v \supset C_v$ by the same three-step process, except that we now use the constant λ_4 in place of λ_5, and the set C_v^o in place of K_v^o Thus, we have clusters of radius $\approx \lambda_5 \log n$, enclosures of radius $\approx \lambda_4 \log n$, and they are separated by a mutual distance of $\approx \lambda_3 \log n$.

Following the algorithm of Chapter 6, we now must build crossbar structures in the enclosures to replace the natural crossbars of the mesh. We build crossbars using Frank's extension (Theorem 2.2.4) of the Okamura–Seymour theorem [84, 37]. To be precise,

Definition 7.3.2 *If $X \subset V$, we say a* crossbar *anchored in X is a set of edge-disjoint paths, each with at least one end in X, such that every pair of paths meets in at least one vertex.*

Let $\sigma_v = \pi(C_v)$, and recall that by Q'_v we mean the facial cycle of $G[C_v]$ containing σ_v. Analogously, let $\tau_v = \pi(D_v)$, and φ_v the facial cycle of $G[D_v]$ containing τ_v. We wish to build a crossbar in $G[D_v \setminus C_v^o]$, anchored in $\sigma_v \cup \tau_v$, of size at least a constant fraction of $|\sigma_v \cup \tau_v|$. For a large enough constant κ depending on ε, we choose a set σ'_v of $|\sigma_v|/\kappa$ vertices on σ_v spaced about κ apart, and a set τ'_v of $|\tau_v|/\kappa$ on τ_v spaced about κ apart.

Lemma 7.3.3 *There is a crossbar anchored in $\sigma'_v \cup \tau'_v$, such that each vertex of $\sigma'_v \cup \tau'_v$ is the endpoint of a distinct path of the crossbar.*

Proof. Consider the planar graph $G[D_v \setminus C_v^o]$. This graph has only two large faces — the outer face bounded by φ_v, and the inner face left by the deletion of C_v^o, and bounded by the cycle Q'_v. We find a shortest path P^* in the planar dual graph whose endpoints are equal to these two large faces, and we delete the edges used by this path. Denote the resulting graph by G_v; note that it has only a single large face, bounded by a cycle φ'_v which contains $\sigma'_v \cup \tau'_v$. We observe that

$$\begin{aligned} |\varphi'_v| &\leq |\varphi_v| + |Q'_v| + 2\ell|P^*| \\ &\leq 9\beta\lambda_4 \log n + 9\beta\lambda_5 \log n + 7\ell\lambda_4 \log n \\ &\leq 9\beta\ell(2\lambda_4 + \lambda_5) \end{aligned}$$

where the last inequality is merely for the sake of having a simpler expression to use in the discussion below.

We claim that the cycle φ'_v is ε_1-smooth in the graph G_v, where

$$\varepsilon_1 = \min(\frac{\varepsilon}{1 + \ell + \varepsilon\ell}, (27\beta\ell)^{-1}).$$

To see this, suppose that P is a path in G_v with endpoints $u, w \in \varphi'_v$. There are four cases to consider.

(i) If u and w both belong to Q'_v, or both belong to φ_v, then this follows from the ε-smoothness of these two cycles in $G[D_v \setminus C_v^o]$.

(ii) If one belongs to each, then $|P| \geq (\lambda_4 - 4\lambda_5) \log n$, while $|\varphi'_v| \leq 9\beta\ell(2\lambda_4 + \lambda_5)$, and again the bound follows.

(iii) If both u and w lie on the short dual path P^*, then since P^* is a shortest path in the planar dual, we have $|P| \geq \ell^{-1} d_{\varphi'_v}(u, w)$.

(iv) Finally, suppose one end lies on P^* and the other does not — let us assume that the other end is on Q'_v. Let P' denote the subpath of P^* lying between Q'_v and the endpoint of P, and P'_v denote the (shorter) subpath of Q'_v lying between the endpoints of P and P^* on Q'_v. Since P^* is a shortest path in the planar dual, we have $|P| \geq \ell^{-1}|P'|$; by the ε-smoothness of Q'_v we have $|P \cup P'| \geq \varepsilon |P'_v|$. Combining these two inequalities, we obtain

$$|P| \geq \frac{\varepsilon}{1+\ell+\varepsilon\ell} |P' \cup P'_v|,$$

and so the bound holds in this case as well.

Write $X = \sigma'_v \cup \tau'_v$. Let us assume for simplicity that $|X|$ is odd. Now for $u \in X$, define u^+ to be the vertex on φ'_v that is $\frac{1}{2}\kappa$ steps clockwise from u, and write $X^+ = \{u^+ : u \in X\}$. Now $|X \cup X^+|$ is even, so we can pair each $u \in X \cup X^+$ with its unique "antipodal" point $\tilde{u} \in X \cup X^+$ under the clockwise ordering of φ'_v. Note that vertices in X are paired with vertices in X^+, and vice versa. We now define a disjoint paths problem in G_v, with the set of terminal pairs $\mathcal{T}_v$ equal to $\{(u, \tilde{u}) : u \in X\}$. Note that all terminals are at least $\frac{1}{2}\kappa$ from one another on the cycle φ'_v.

We now want to show that $\mathcal{T}_v$ is realizable in G_v. First, say that a cut is *non-trivial* if it separates at least one pair of terminals. We are dealing with a disjoint paths problem in a planar graph with all terminals on the outer face. Moreover, every node of G_v not on the outer face has even degree. Thus, by Theorem 2.2.4, the following *strict cut condition* is sufficient for the realizability of $\mathcal{T}_v$ — every non-trivial cut has more capacity than the number of terminal pairs it separates.

By Lemma 2.1.2, it is enough to consider non-trivial cuts of the form $\delta(U)$ with both $G_v[U]$ and $G_v \setminus U$ connected. For such a set U, there must be two vertices $v, w \in \pi(U)$ such that $v, w \in \varphi'_v$. Suppose that the distance from v to w in G_v is d; then by Lemma 7.1.6, we have $|\delta(U)| \geq \ell^{-1}d$. Since the facial cycle φ_v is ε_1-smooth, the number of terminal pairs disconnected by $\delta(U)$ is at most $2\varepsilon_1^{-1}d/\kappa$. Thus taking $\kappa > 2\ell\varepsilon_1^{-1}$ ensures that the strict cut condition will be satisfied.

Finally, observe that the edge-disjoint paths in a realization of $\mathcal{T}_v$ provide the crossbar required by the lemma, since each pair of paths must meet at some vertex of G_v. ■

In the crossbar just constructed, let Y_v^u denote the path with an endpoint equal to u, where $u \in \sigma'_v \cup \tau'_v$.

7.4 Defining the Simulated Network

We now define the *simulated network*; the nodes of this network are the clusters, which we represent by the vertices in M. We define a neighbor relation on the clusters using the notion of a *Voronoi partition*; two clusters will be joined by an edge in $\mathcal{N}$ if they are neighbors in this sense.

Let us first introduce the Voronoi partition at a general level. Let H be a graph and $S \subset V(H)$. We fix a lexicographic ordering $\preceq$ on the elements of S. For $s \in S$, let

$$\mathcal{U}_s = \{v \in G : \forall s' \in S : d(v,s) \leq d(v,s') \text{ and } \forall s' \preceq s : d(v,s) < d(v,s')\}.$$

That is, $\mathcal{U}_s$ is the set of vertices that are at least as close to s as to any other s', with ties broken based on $\preceq$.

Definition 7.4.1 *The* Voronoi partition $\mathcal{V}(H,S)$ *of H with respect to S is the partition $\{\mathcal{U}_s : s \in S\}$.*

The following fact is immediate.

Lemma 7.4.2 *For each $s \in S$, $H[\mathcal{U}_s]$ is connected.*

Proof. Suppose $v \in \mathcal{U}_s$; we claim that any shortest s-v path P is contained in $\mathcal{U}_s$. For suppose not, and let $v' \in P$ be the closest vertex to s that lies in $\mathcal{U}_{s'}$ for some $s' \neq s$. Then $d(s',v) \leq d(s,v)$, and in fact $d(s',v) < d(s,v)$ if $s \preceq s'$. It follows that $v \in \mathcal{U}_{s'}$, a contradiction. ■

We can now build a graph $\mathcal{N}(H,S)$ on the vertices in S, joining two if their Voronoi cells share an edge.

Definition 7.4.3 *The* neighborhood graph *of S in H, denoted $\mathcal{N}(H,S)$, is the graph with vertex set S, and an edge (s,s') iff there is an edge of H with endpoints in $\mathcal{U}_s$ and $\mathcal{U}_{s'}$.*

The simulated graph we use will be the neighborhood graph $\mathcal{N}(G,M)$ with every edge given capacity $\approx \lambda_6 \log n$. Let $\mathcal{V}$ and $\mathcal{N}$ denote $\mathcal{V}(G,M)$ and $\mathcal{N}(G,M)$ respectively, and $\mathcal{N}(\gamma)$ the graph $\mathcal{N}$ in which each edge is given capacity γ.

We now establish some basic facts about $\mathcal{N}$. First, by the maximality of M, we have

Lemma 7.4.4 *For all $v \in M$, $\mathcal{U}_v \subset B_{\lambda_3 \log n}(v)$.*

Proof. Suppose $u \in \mathcal{U}_v$ but $d(v, u) > \lambda_3 \log n$. Then $d(v', u) > \lambda_3 \log n$ for *all* $v' \in M$; this contradicts the fact that M is a maximal independent set in $G^{\lambda_3 \log n}$. ■

Definition 7.4.5 *For $v \in M$, and a number $p > 0$, we define $g_v(p)$ to be the number of $u \in M$ for which $\mathcal{U}_u \subset B_{p\lambda_3 \log n}(v)$. We define $g'_v(p)$ to be the number of $u \in M$ for which $\mathcal{U}_u$ has a non-empty intersection with $B_{p\lambda_3 \log n}(v)$. Finally, we define*

$$g(p) = \max_{v \in M} g_v(p)$$

$$g'(p) = \max_{v \in M} g'_v(p)$$

Using Lemmas 7.1.3 and 7.4.4, we show

Lemma 7.4.6 $g(p) \leq 4\bar{\alpha}^{-1}\beta p^2 = O(p^2)$.

Proof. By the definition of M, we see that $B_{\frac{1}{2}\lambda_3 \log n}(v) \subset \mathcal{U}_v$ for every $v \in M$. Thus,

$$|\mathcal{U}_v| \geq \frac{1}{4}\bar{\alpha}^{-1}\lambda_3^2 \log^2 n.$$

Now by Lemma 7.1.3, we have

$$|B_{p\lambda_3 \log n}(v)| \leq \beta p^2 \lambda_3^2 \log^2 n$$

for every $v \in G$; from this it follows that $g(p) \leq 4\bar{\alpha}^{-1}\beta p^2$. ■

Lemma 7.4.7 $g'(p) \leq g(p+2)$.

Proof. Let $v \in M$ be a vertex maximizing $g'_v(p)$. If $u \in M$ is such that $\mathcal{U}_u$ has a non-empty intersection with $B_{p\lambda_3 \log n}(v)$, then by Lemma 7.4.4, $u \in B_{(p+1)\lambda_3 \log n}(v)$, and $\mathcal{U}_u \subset B_{(p+2)\lambda_3 \log n}(v)$. ■

Lemma 7.4.8 *The degree of a vertex in $\mathcal{N}$ is at most $\Delta' \leq g(3)$.*

Proof. Consider a vertex $v \in M$. By Lemma 7.4.4, any $u \in M$ that is a neighbor of v with respect to $\mathcal{V}(G, M)$ has the property that $\mathcal{U}_u$ intersects $B_{\lambda_3 \log n}(v)$. Thus, the number of such neighbors u is at most $g'(1) \leq g(3)$. ■

We can now make explicit the sense in which the network $\mathcal{N}(\Theta(\log n))$ "represents" the graph G sufficiently well; this will serve as the analogue of Lemma 6.1.6. As in Chapter 6, if u is a vertex in a cluster C_v, we define $\psi(u) = v$; we say that $\psi(u)$ is undefined if $u \notin \cup_{v \in M} C_v$.

Lemma 7.4.9 *There is a constant γ such that the following holds. If $\mathcal{T}'$ be a realizable subset of $\mathcal{T}$, with the property that all terminals in $\mathcal{T}'$ belong to $\cup_{v \in M} C_v$, then $\psi(\mathcal{T}')$ is realizable in $\mathcal{N}(\gamma \log n)$.*

Proof. Set $\gamma = 9\beta\Delta(\lambda_5 g(5) + \lambda_3)$. For each s_i-t_i path P in the optimal routing, construct the following path for $(\psi(s_i), \psi(t_i))$ in $\mathcal{N}$ — when P crosses from $\mathcal{U}_w$ into $\mathcal{U}_{w'}$, add an edge from w to w'. Now consider how much demand in our constructed routing uses the edge (w, w'). This demand corresponds to paths in the original routing that used an edge in $\delta(\mathcal{U}_w, \mathcal{U}_{w'})$. We can't bound the size of this set directly, since it could be quite "meandering." Thus we invoke the following argument. $\delta(\mathcal{U}_w, \mathcal{U}_{w'}) \subset B_{2\lambda_3 \log n}(x)$ for some vertex $x \in G$; thus there is some r between $2\lambda_3 \log n$ and $3\lambda_3 \log n$ so that $|\pi(B_r(x))| \leq 9\beta\lambda_3 \log n$, and hence $|\delta(B_r(x))| \leq 9\Delta\beta\lambda_3 \log n$. This is an upper bound on the total demand crossing $\delta(\mathcal{U}_w, \mathcal{U}_{w'})$ associated with terminal pairs with both ends more than r away from x. But closer than this, there are at most $g'(r) \leq g(5)$ clusters, each of which is the origin of at most $9\Delta\beta\lambda_5 \log n$ total demand in $\mathcal{T}'$. ■

7.5 Building the Simulated Network

The goal of this part is, for a constant λ_6, to construct $\lambda_6 \log n$ disjoint paths between each pair of enclosures D_v, D_w where (v, w) is an edge in $\mathcal{N}$. This will allow us to convert a routing in the simulated network $\mathcal{N}(\lambda_6 \log n)$ into actual disjoint paths in G. Recall that the outer facial cycle of $G[D_v]$ is denoted φ_v, and it contains a set τ'_v of vertices evenly spaced at distance κ.

Theorem 7.5.1 *There exist vertex-disjoint paths in G, each with ends in sets τ'_v and otherwise disjoint from all D_v, such that for $(v, w) \in E(\mathcal{N})$, there are at least $\lambda_6 \log n$ such paths with one end in τ'_v and the other in τ'_w.*

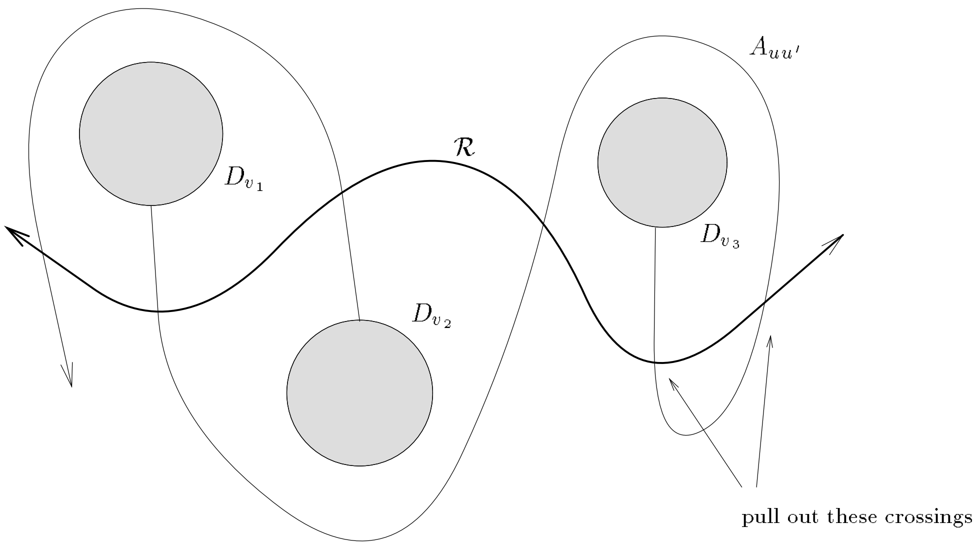

Figure 7-2: Pulling out a crossing

Proof. The proof is based on Theorem 2.3.3, due to Schrijver [104]. Recall that a curve is G-normal if it meets the drawing of G only at vertices; we define its G-length to be the number of times it meets the drawing. For each v, v' that are neighbors in $\mathcal{N}$, we draw a simple G-normal curve $\mathcal{A}_{vv'}$ on Σ with endpoints v and v'. We can ensure that all these curves are disjoint, since each $\mathcal{U}_v$ is connected, and for each $(v, v') \in E(\mathcal{N})$, there is at least one edge of G with endpoints in $\mathcal{U}_v$ and $\mathcal{U}_{v'}$. Choose a small constant $\lambda_6 \leq |\tau'_v|/\Delta'$ (say, less than $\frac{1}{12}\alpha^2\bar{\alpha}\Delta'^{-1}\Delta^{-2}\lambda_3^{-1}\lambda_4^2 g(6)^{-1}$; the reason for this will become clear below). We construct $\lambda_6 \log n$ copies of each $\mathcal{A}_{vv'}$, all running "parallel" to one another. By pushing them apart appropriately, we can assume that each curve runs through a different vertex in τ'_v and $\tau'_{v'}$. Let $A_{uu'}$ denote the G-normal curve that runs through $u \in \tau'_v$ and $u' \in \tau'_{v'}$.

Define G' to be the graph obtained by deleting D_v^o for each $v \in M$. We now cut Σ along the facial cycles φ_v to obtain Σ', a surface with boundary. Note that G' is properly embedded on Σ'. The theorem is now a consequence of the following claim. ■

Claim 7.5.2 *There exist vertex-disjoint paths $P_{uu'}$ in G' such that $P_{uu'}$ is homotopic to $A_{uu'}$.*

Proof. If curves $\mathcal{R}$ and $\mathcal{R}'$ are homotopic, we write $\mathcal{R} \sim \mathcal{R}'$. From Theorem 2.3.3, we can obtain the following somewhat simpler sufficient condition for the existence of the desired paths: for each essential curve $\mathcal{R}$ on Σ', one has

$$\mathrm{cr}(\mathcal{R}, G') > \sum_{(u,u')} \mathrm{mincr}(\mathcal{R}, A_{uu'}). \tag{7.2}$$

Note that in verifying this inequality, we may assume $\mathcal{R}$ is G'-normal and has no self–crossings.

For an essential curve $\mathcal{R}$, define its *index* to be

$$\mathrm{cr}(\mathcal{R}, G') - \sum_{(u,u')} \mathrm{mincr}(\mathcal{R}, A_{uu'}).$$

So it is enough to consider curves $\mathcal{R}$ whose index is minimum in their homotopy class, and to show that such $\mathcal{R}$ in fact have positive index.

Set $r = \frac{1}{2}\Delta^{-1}\alpha\bar{\alpha}^{1/2}\lambda_4 \log n$. We claim that no G-normal closed curve of G-length less than $2r$ can enclose a set of the form D_v. For if it did, then D_v could be disconnected by an edge cut of size less than $2\Delta r = \alpha\bar{\alpha}^{1/2}\lambda_4 \log n$, which is not possible since $|D_v| \geq \bar{\alpha}\lambda_4^2 \log^2 n$. From this it follows that any G'-normal closed curve of G'-length less than $2r$ must be null-homotopic in Σ'.

We now consider two cases, based on the G'-length of $\mathcal{R}$.

Case 1. $\mathrm{cr}(\mathcal{R}, G') \leq r$. Then $\mathcal{R}$ must have both endpoints on the same facial cycle (it is too short to touch two such cycles, and if it were a closed curve it would have to be null-homotopic, by the above argument.) But then it is easy to produce curves $\{A'_{uu}\}$ for which $\mathrm{cr}(\mathcal{R}, G') > \sum_{(u,u')} \mathrm{cr}(\mathcal{R}, A'_{uu'})$ since φ_v is ε-smooth.

Case 2. $\mathrm{cr}(\mathcal{R}, G') > r$. Again, we just have to exhibit simple curves $A'_{uu'} \sim A_{uu'}$ lying on Σ' so that

$$\mathrm{cr}(\mathcal{R}, G') > \sum_{(u,u')} \mathrm{cr}(\mathcal{R}, A'_{uu'}), \tag{7.3}$$

If the set $\{A_{uu'}\}$ satisfies (7.3), we are done; otherwise, we show how to modify this set of curves so that it does. See Figure 7-2.

If the set $\{A_{uu'}\}$ does not satisfy Inequality (7.3), then there is some interval $\mathcal{R}'$ of $\mathcal{R}$ of G'-length r for which (7.3) is violated. Let us consider such an $\mathcal{R}'$.

Observe that each curve $A_{uu'}$, as initially constructed, can be assumed to have G-length at most $3\lambda_3 \log n$, since u and u' belong to enclosures in neighboring Voronoi cells. Thus at most $\Delta' g(6)\lambda_6 \log n$ of these curves can meet $\mathcal{R}'$, since at most $\Delta' g(6)$ pairs of clusters have at least one end close enough to $\mathcal{R}'$. Now suppose the total number of crossings of these curves with $\mathcal{R}'$ exceeds

$$\frac{(3\lambda_3 \log n)(\Delta' g(6)\lambda_6 \log n)}{r} < \mathrm{cr}(\mathcal{R}', G).$$

Then some curve $A_{uu'}$ meets $\mathcal{R}'$ more than $3\lambda_3 \log n/r$ times, and hence the interval of $A_{uu'}$ between some pair of consecutive crossings with $\mathcal{R}'$ has G'-length less than r.

Suppose that this pair of consecutive crossings occurs at vertices w and w'. Let $\mathcal{R}''$ denote the G'-normal curve formed from this interval of $A_{uu'}$ and the portion of $\mathcal{R}'$ between w and w'. $\mathcal{R}''$ has G'-length less than $2r$, and so it must be null-homotopic by the argument given above.

Now, since $\mathcal{R}$ has minimum index over all curves in its homotopy class, the portion of $A_{uu'}$ between w and w' meets G' at least as many times as the portion of $\mathcal{R}'$ between w and w'. We can therefore modify $A_{uu'}$ so that it runs along $\mathcal{R}'$ for this interval. This does not increase the G'-length of $A_{uu'}$; and it decreases the number of crossings of $\mathcal{R}$ with $A_{uu'}$.

Thus this process terminates; when it does, we have a set of curves $\{A'_{uu'}\}$ for which Inequality 7.3 holds. ■

Let us denote one such path with ends u and u' by $Z_{uu'}$. Moreover, we have $u \in \tau'_v$ and $u' \in \tau'_{v'}$, and they are ends of paths Y^u_v and $Y^{u'}_{v'}$ respectively. Denote by $\bar{Z}_{uu'}$ the concatenation of the three paths Y^u_v, $Z_{uu'}$, and $Y^{u'}_{v'}$.

7.6 The On-Line Algorithm

With the simulated network $\mathcal{N}$ in place, the routing algorithms themselves are essentially the same as those for the mesh; here we just describe what must be modified.

First of all, the analogue of Lemma 6.2.6 is the following.

Lemma 7.6.1 *Let $r \leq \lambda_1 \log n$, $U \subset B_r(v)$ for some $v \in V$, and T' a set of terminal pairs in U. Then the maximum weight of a subset of T' that is realizable in $B_{8\xi^2 r}(v)$ is within a constant factor of the maximum weight of a subset of T' that is realizable in G.*

Proof. First choose a radius r' between $2r$ and $3r$ for which $|\pi(B_r(v)| \leq 9\beta r'$. Then construct a simple set extension of $B_{r'}(v)$ as in Lemma 7.1.5; and ε-smooth its outer facial cycle to obtain a set $U' \supset U$ contained in $B_{4\xi r}(v)$. Let $U'' \subset B_{8\xi^2 r}$ denote a simple set extension of $B_{8\xi r}(v)$, as in Lemma 7.1.5. For a constant κ', we can pick a set S of vertices on the outer facial cycle of U' spaced κ' apart, and use Frank's theorem [37] as in Lemma 7.3.3 to construct a set of edge-disjoint paths connecting "antipodal" pairs in S, such that all paths stay within $U'' \setminus U'$. Note that we must take care to ensure that the parity condition is met, since the outer facial cycle of U'' can contain odd-degree vertices. To do this we remove sub-paths of this cycle between consecutive pairs of the (necessarily even number of) odd-degree vertices; U'' is large enough that the strict cut condition will remain satisfied.

Consider the set of paths in a realization of a maximum-weight subset of $\mathcal{T}'$ in G, and let $\mathcal{T}''$ denote the subset of pairs whose paths meet $\pi(U')$. We break each at their first and last intersections with $\pi(U')$. We now connect a subset of these paths to their respective closest vertices in S, following the outer facial cycle of U' — the subset is chosen to be maximal subject to the constraint that no more than 1 unit of demand is connected to any single vertex in S. It is easy to argue that any such maximal set contains a constant fraction of the demand in $\mathcal{T}''$. We now use the crossbar of the previous paragraph to connect all of these pairs together; in this way we are routing an amount of demand that is within a constant fraction of the maximum achievable in G, using paths that do not leave U''. ■

The algorithm for short connections is now as follows. We run a randomized version of Luby's algorithm, this time in $G^{\lambda_1 \log n}$. Let M' denote the resulting MIS. With constant probability, both ends of a short connection are within $\frac{1}{16\xi^2}\lambda_1 \log n$ of the same $v \in M'$, as in Lemmas 6.2.7 and 7.3.1. We now let U_v denote $B_{\lambda_1 \log n/16\xi^2}(v)$ and only route connections both of whose ends lie in the same U_v. To route such connections, we run the greedy algorithm of Theorem 5.1.3 in each $B_{\frac{1}{2}\lambda_1 \log n}(v)$; by Lemma 7.6.1, this is within $O(\log n)$ of optimal in each U_v.

For long connections, we run the same simulation as in the case of the mesh, this time in the graph $\mathcal{N}(\frac{1}{2}\lambda_6 \log n)$. The analogue of Lemma 6.2.4 holds exactly as before, as does Lemma 6.2.5 — making use of the paths $\bar{Z}_{uu'}$, since all such paths incident to the same enclosure must cross.

Thus the optimum for long connections in G is bounded by the fractional optimum in

$\mathcal{N}(\gamma \log n)$, which is at most a constant factor more than the fractional optimum in $\mathcal{N}(\frac{1}{2}\lambda_6 \log n)$, which is at most an $O(\log n)$ factor more than the total demand routed by the on-line algorithm in G. Thus we have

Theorem 7.6.2 *The on-line algorithm is an $O(\log n)$-approximation in any densely embedded Eulerian graph G.*

7.7 The Off-Line Algorithm

The off-line algorithm too is essentially the same, now working with the larger simulated network $\mathcal{N}'(\lambda_6 \log n)$. The only non–trivial change required is in the proof of Lemma 6.3.1. Here, we are no longer able to talk about "overfull rectangles"; however, we can define a *round cut* to be a set of the form $B_r(w) \cap C_v$. Since a given $u \in C_v$ is only contained in $O(r^2)$ round cuts of radius r, the following lemma establishes that round cuts can be used instead of rectangles in the analogue of Lemma 6.3.1 for densely embedded graphs.

Lemma 7.7.1 *Let G'_v denote $G[C_v]$ with an additional vertex z_v joined by an edge to each vertex in σ'_v. Then there is a constant ξ_1 such that for every $U \subset G'_v$ not containing z_v, there is a round cut $R \supseteq U$ satisfying $|\delta(R)| \leq \xi_1 |\delta(U)|$.*

Proof. Set $\xi'_1 = \kappa + \bar{\alpha}^{-1/2}\alpha^{-1}$ and $\xi_1 = 4\Delta\xi'_1(\beta + \varepsilon^{-1})$. Let $U \subset G'_v$ be a set not containing z_v, and write $p = |\delta(U)|$; we construct a round cut R containing U for which $|\delta(R)| \leq \xi_1 p$.

Now if we contract $G'_v \setminus U$ to a single vertex, we obtain a planar graph with maximum face size κ (as opposed to ℓ; this is due to the large spacing of the vertices of σ'_v). So by Lemma 7.1.6, the maximum distance between two points on $\pi(U)$ is at most κp.

Next we claim that every vertex in U must be within distance $\bar{\alpha}^{-1/2}\alpha^{-1}p$ of $\pi(U)$. The reason for this is analogous to the proof of Lemma 7.1.5 — if not, then U would contain a ball of more than this radius, which would contain more than $\alpha^{-2}p^2$ vertices; a contradiction since $|\delta(U)| = p$.

Thus for any $u \in \pi(U)$, we have $U \subset B_r(u)$, where

$$r = (\kappa + \bar{\alpha}^{-1/2}\alpha^{-1})p = \xi'_1 p.$$

Now by Lemma 7.1.4, there is an r' between r and $2r$ such that

$$|\delta(B_{r'}(u))| \leq 4\beta\Delta\xi'_1 p.$$

Now let $R \supset U$ denote the round cut $B_{r'}(u) \cap C_v$. Every edge of $\delta(R)$ is an edge of either $\delta(B_{r'}(u))$ or of $\delta(C_v \cap R)$. The former quantity was just shown to be at most $4\beta\Delta\xi'_1 p$. To bound the latter quantity, note that any two vertices in $\pi(C_v \cap R)$ are at most $2r' \leq 4\xi'_1 p$ apart; since the facial cycle containing $\pi(C_v)$ is ε-smooth, this means that $\pi(C_v \cap R)$ contains at most $4\varepsilon^{-1}\xi'_1 p$ vertices, and hence

$$|\delta(C_v \cap R)| \leq 4\Delta\varepsilon^{-1}\xi'_1 p.$$

The claim now follows since

$$|\delta(R)| \leq 4\Delta\xi'_1(\beta + \varepsilon^{-1})p = \xi_1 p.$$

■

Thus we have

Theorem 7.7.2 *There is a randomized (off-line)* UFP *algorithm in densely embedded Eulerian graphs that produces a constant-factor approximation with high probability.*

Chapter 8

Minimizing Congestion

... Here the street is narrow:
The throng that follows Caesar at the heels,
Of senators, of praetors, common suitors,
Will crowd a feeble man almost to death ...
— *William Shakespeare,* Julius Caeser.

For the previous three chapters, our concern has been with maximizing the total amount of demand routed, when given a graph with edge capacities, and terminal pairs with specified demands. Here, we consider a different objective function that also serves to measure the "quality" of a routing: we wish to route all the demand, and exceed the capacity constraints by as little as possible.

To be more concrete, let us recall the definitions given in Chapter 1. We say that a *routing* of $\mathcal{T}$ in G is a choice of an s_i-t_i path P_i for each $(s_i, t_i) \in \mathcal{T}$. The *congestion* of a given routing, $\nu(\mathcal{T}, \{P_i\})$, is defined to be the maximum, over all edges $e \in G$, of the total demand crossing e. That is,

$$\nu(\mathcal{T}, \{P_i\}) = \max_e \sum_{i: e \in P_i} \rho_i.$$

We use $\nu(\mathcal{T})$ to denote the minimum congestion of a routing of $\mathcal{T}$ in G. In Chapter 1, we indicated some of the ways in which the issue of congestion minimization arises.

We will sometimes speak of an X–Y *routing* (or a routing from X to Y) where X and Y are two sets of vertices in a graph G. By this we mean a set of paths in G such that each path has one end in a distinct vertex of X and its other end in some vertex of Y.

The simulated network construction of the previous chapter, though designed with the UFP in mind, also allows us to give a constant–factor approximation for the problem of minimizing congestion in any densely embedded graph. (For this result, we do not require the graph to be Eulerian.) We describe this algorithm in Section 8.2, after first discussing some previous work in Section 8.1.

8.1 Preliminaries

Algorithmically, the problem of minimizing congestion is much better understood than the UFP (which seeks to maximize the amount of demand routed). One of the primary reasons for this is that the natural fractional version of this problem gives much more information about the unsplittable optimum than was the case for the UFP. Recalling the terminology of Section 2.4, we define a *fractional routing* of $\mathcal{T}$ in G as follows. For each pair $(s_i, t_i) \in \mathcal{T}$, we choose

(i) a set of s_i-t_i paths $P_i^1, \ldots, P_i^{q_i}$, and

(ii) associated non-negative weights $y_i^1, \ldots, y_i^{q_i} \in \mathbf{R}$ such that $\sum_j y_i^j = \rho_i$.

The congestion of this routing is now defined to be

$$\nu^f(\mathcal{T}, \{P_i\}) = \max_e \sum_{(i,j):e \in P_i^j} y_i^j.$$

We use $\nu^f(\mathcal{T})$ to denote the minimum congestion of a fractional routing of $\mathcal{T}$ in G.

The randomized rounding method of Raghavan and Thompson is clearly applicable in this setting: one computes a fractional routing for $\mathcal{T}$; and then for (s_i, t_i) one chooses the path P_i^j with probability y_i^j [94]. Since we are no longer worried about respecting the capacity constraints, there is no need to impose any "high–capacity" requirements — the algorithm can be used in any graph. The performance guarantee of the algorithm can be analyzed using tail bounds of the type in Theorem 2.5.1; one can show [94] that with high probability, the resulting unsplittable routing has congestion $\nu^f(\mathcal{T}) + o(\nu^f(\mathcal{T})) + O(\log |E|)$. For our purposes here, it is enough to state a somewhat weaker version of this performance guarantee.

Theorem 8.1.1 (Raghavan–Thompson) *There are constants b_0 and b_1 (independent of G and $\mathcal{T}$) such that the following holds. If all edge capacities are* 1*, and all demands are at most* 1*, then the above algorithm produces a routing of congestion at most $b_0 \nu^f(\mathcal{T}) + b_1 \log |E|$ with high probability,*

Thus, there are never large gaps between the fractional and integral optima — this is in contrast to Theorem 2.4.4, which applied to the UFP. Indeed, the fractional and integral optima for congestion are always within an $O(\log |E|)$ factor of one another in general, and within a constant factor when $\nu^f(\mathcal{T})$ is at least $\Omega(\log |E|)$.

It is interesting to note, though it not relevant to the material here, that good on–line approximations have recently been developed by Aspnes, Azar, Fiat, Plotkin, and Waarts [8].

Theorem 8.1.2 (Aspnes et al.) *There is an on–line $O(\log n)$–approximation to $\nu(\mathcal{T})$ in any n–node graph.*

Thus, the principal question left open by the Raghavan–Thompson algorithm is the following. The algorithm provides a constant–factor approximation whenever the fractional optimum is sufficiently large; but it is only an $O(\log |E|)$–approximation in general. Thus, for example, if we are given a set $\mathcal{T}$ of terminal pairs that is in fact *realizable* in G (and hence routable with congestion 1), the algorithm of Theorem 8.1.1 will still produce a routing of congestion $\Omega(\log |E|)$ with high probability. It is therefore natural to ask whether one can obtain a constant–factor approximation for $\nu(\mathcal{T})$ for all ranges of the fractional optimum; it appears that techniques beyond pure randomized rounding will be necessary to resolve this question.

In this chapter, we address this question for the case of densely embedded graphs, and provide a constant–factor approximation for $\nu(\mathcal{T})$ in such graphs, independent of the value of $\nu^f(\mathcal{T})$. Combined with Theorem 8.1.1, this settles an open question of Karp, Leighton, Rivest, Thompson, Vazirani, and Vazirani [57] on the existence of a constant–factor approximation for congestion on the two–dimensional mesh. The algorithm makes use of randomized rounding within the *simulated network* defined in the previous two chapters, and is thus quite similar to the UFP approximation algorithms just developed. However, it is less difficult, in large part because we are no longer faced with problem of choosing *which* subset of terminals to route.

8.2 The Approximation Algorithm

We will not make use of all parts of the simulated network construction; we begin by recalling those aspects that we do use, and consolidating the notation. First of all, let G be an n-node densely embedded graph with parameters α, λ, Δ, and ℓ; by doubling every edge of G, we lose only a factor of 2 in the approximation ratio, and we may therefore suppose that G is Eulerian.

As in Chapter 7, M is a set of vertices of G, maximal subject to the condition that no two vertices of M are closer than $\lambda_3 \log n$ to one another. $\mathcal{V}$ is the Voronoi partition of G with respect to M, and $\mathcal{N}$ is the neighborhood graph of M in G. Recall that $\mathcal{U}_v$ denotes the "Voronoi region" associated with $v \in M$ in the partition $\mathcal{V}$. The cluster/enclosure structures inside each Voronoi region are no longer necessary; however, we will be using the inter–cluster paths that pass through them. Thus, we recall that for each $v \in M$, there is an ε-smooth cycle

$$\varphi_v \subset B_{3\lambda_4 \log n}(v) \setminus B_{2\lambda_4 \log n}(v).$$

For a large enough constant κ, we choose a set of vertices τ'_v spaced (approximately) κ apart on φ_v. Finally, by Lemma 7.3.3 and Theorem 7.5.1, there exist edge–disjoint paths in G, such that

(i) for each pair (v, w) that is an edge of $\mathcal{N}$, at least $\lambda_6 \log n$ paths pass through both a vertex in both τ'_v and τ'_w,

(ii) at most one path passes through any one vertex in $\cup_{v \in M} \tau'_v$,

(iii) if two paths pass through nodes in the same set τ'_v, then they meet at some node.

Deleting vertices from τ'_v if necessary, we may assume that every vertex in τ'_v lies on one of these paths. If $u \in \tau_v$ and $u' \in \tau_{v'}$ lie on a common path, then we use $\bar{Z}_{uu'}$ to denote this path. Let

$$\mathcal{Z}_{vv'} = \{\bar{Z}_{uu'} : u \in \tau_v, u' \in \tau_{v'}\},$$

and

$$\mathcal{Z}'_v = \bigcup_{w:(v,w) \in E(\mathcal{N})} \mathcal{Z}_{vw}.$$

By analogy with the construction of clusters and enclosures from Section 7.3, we construct a simple set F_v around each $v \in M$, such that, for constants ε and γ_1,

(i) $G[F_v]$ has an ε-smooth outer facial cycle R_v that contains $\pi(F_v)$,

(ii) $|R_v| \leq \gamma_1 \log n$, and

(iii) $R_v \subset B_{3\lambda_3 \log n}(v) \setminus B_{2\lambda_3 \log n}(v)$.

We define a function $\psi : \mathcal{T} \to M$ by setting $\psi(u) = v$ if $u \in \mathcal{U}_v$. We can now describe the routing algorithm. Recall that a connection is *long* if $d(s_i, t_i) \geq \lambda_2 \log n$. Below, we present the algorithm for long connections; short connections are handled recursively as in the algorithm of Section 7.7, and we will briefly elaborate on this at the end. Thus, for the time being, we assume that all connections are long. The algorithm is as follows.

(i) Apply the randomized rounding algorithm to the graph $\mathcal{N}$, with terminal pairs

$$\{(\psi(s_i), \psi(t_i) : (s_i, t_i) \in \mathcal{T}\}.$$

The result is, for each (s_i, t_i), a sequence

$$\mu_i = \mu_i^0 \mu_i^1 \cdots \mu_i^{q_i}$$

of neighboring vertices of M, such that $\mu_i^0 = \psi(s_i)$ and $\mu_i^{q_i} = \psi(t_i)$.

(ii) For an edge (v, w) of $\mathcal{N}$, let $\mathcal{T}_{vw}$ denote the set of terminal pairs that are routed on the edge (v, w) by step (i). We assign terminal pairs to paths in $\mathcal{Z}_{vw}$ so that the maximum demand across any one path is minimized.

By using the fact that all pairs of paths incident to a given set τ'_v cross, we thus obtain for each pair (s_i, t_i), a path P'_i that has one end in $\tau'_{\psi(s_i)}$ and the other end in $\tau'_{\psi(t_i)}$.

(iii) For each $v \in M$, we run the algorithm of Corollary 3.2.3 in the graph $G[F_v]$ to produce a routing of $\psi^{-1}(v)$ to τ'_v whose congestion is within a constant factor of optimal. Now suppose that $s_i \in \psi^{-1}(v)$ is routed by this algorithm to $u \in \tau'_v$ on a path Q_w; and let $\bar{Z}_{uu'}$ denote the path in $\mathcal{Z}'_v$ that passes through u. Then $Q_w \cup \bar{Z}_{uu'}$ contains a path P''_{s_i} which meets the path P'_i returned by step (ii).

(iv) For each $(s_i, t_i) \in \mathcal{T}$, we concatenate the path P_i found in step (ii) with the two paths P''_{s_i}, P''_{t_i} found in step (iii). This is the path we use for routing (s_i, t_i) in G.

The remainder of this section is devoted to showing that the above algorithm provides a constant–factor approximation to $\nu(\mathcal{T})$. In order to do this, it is useful to work with the following two lower bounds on this optimum. First, for $v \in M$, let $\nu_{F,v}$ denote the minimum congestion of a routing of the terminals in $\psi^{-1}(v)$ to the cycle R_v, in the subgraph $G[F_v]$. Define

$$\nu_F^* = \max_{v \in M} \nu_{F,v}.$$

Now we have

Lemma 8.2.1 $\nu(\mathcal{T}) \geq \nu_F^*$.

Proof. Let $v \in M$ be such that $\nu_F^* = \nu_{F,v}$. Now consider any routing of $\mathcal{T}$; since all terminals in $\psi^{-1}(v)$ are ends of long connections, the path associated with each must leave the subgraph $G[F_v]$. Consequently, a prefix of each of these paths constitutes a routing of $\psi^{-1}(v)$ to the cycle R_v that bounds $G[F_v]$. ■

A simple consequence of this is the following.

Lemma 8.2.2 *The total demand of terminals in $\psi^{-1}(v)$, for any $v \in M$, is at most $\nu_F^* \gamma_1 \Delta \log n \leq \nu(\mathcal{T})\gamma_1 \Delta \log n$.*

Proof. Since $|R_v| \leq \gamma_1 \log n$, there are at most $\gamma_1 \Delta \log n$ edges incident to nodes in R_v. Each edge carries at most ν_F^* units of demand, and every unit of demand in $\psi^{-1}(v)$ must traverse one of these edges (since all this demand is associated with long connections. ■

Recall that in step (i) of the approximation algorithm, we considered the routing problem in which the underlying graph is $\mathcal{N}$ and the set of terminal pairs is $\{(\psi(s_i), \psi(t_i)) : (s_i, t_i) \in \mathcal{T})\}$ Let $\nu_{\mathcal{N}}^*$ denote value of the optimum congestion for this problem.

Lemma 8.2.3 $\nu(\mathcal{T}) \geq \frac{\Omega(\nu_{\mathcal{N}}^*)}{\log n}$.

Proof. Using the minimum–congestion routing of $\mathcal{T}$ in G, we construct a routing of congestion $O(\nu(\mathcal{T}) \cdot \log n)$ in $\mathcal{N}$. The construction of the routing in $\mathcal{N}$ is straightforward — for each

$i = 1, \ldots, k$, we use the sequence of Voronoi regions through which the s_i–t_i path in G passes to define the $\psi(s_i)$–$\psi(t_i)$ path in $\mathcal{N}$.

Now we must bound how much demand is sent across an edge (v, w) of $\mathcal{N}$. First, as in the proof of Lemma 7.4.9, $\delta(\mathcal{U}_v, \mathcal{U}_w) \subset B_{2\lambda_3 \log n}(x)$ for some vertex $x \in G$; thus there is some r between $2\lambda_3 \log n$ and $3\lambda_3 \log n$ so that $|\pi(B_r(x))| \leq 9\beta\lambda_3 \log n$, and hence $|\delta(B_r(x))| \leq 9\Delta\beta\lambda_3 \log n$. Since each edge in $\delta(B_r(x))$ carries at most $\nu(\mathcal{T})$ units of demand, the total demand sent across (v, w) that is associated with terminal pairs having at least one end outside $B_r(x)$ is at most $\nu(T) \cdot 9\Delta\beta\lambda_3 \log n$. But there are only $g'(r) \leq g(5)$ Voronoi regions that do not lie outside $B_r(x)$, and by Lemma 8.2.2, each is the origin of at most $\nu(T) \cdot \gamma_1 \Delta \log n$ units of demand. The bound follows from this. ∎

We now provide an upper bound on the congestion of the routing produced by the above algorithm, in terms of the two quantities ν_F^* and $\nu_{\mathcal{N}}^*$. First, the following fact is useful.

Lemma 8.2.4 *There is a constant κ_0 such that the following holds. Let $v \in M$. There exists a routing from R_v to τ_v' in $G[F_v]$ such that at most κ_0 paths in the routing traverse any one edge of $G[F_v]$.*

Proof. It will follow from the calculations below that

$$\kappa_0 = 6\ell\gamma_1\bar{\alpha}^{-1/2}(\alpha\varepsilon\kappa\lambda_6)^{-1}$$

is a sufficiently large constant. We attach a source s^* to each vertex in R_v by an edge of unit capacity, attach a sink t^* to every vertex in τ_v' by an edge of capacity $|R_v|$, and give every edge in $G[F_v]$ a capacity of κ_0. We now show that there is a feasible flow from s^* to t^* of value $|R_v|$, which will imply the lemma.

It is enough to verify the cut condition: for any set X containing s^*, we have $c(X) \geq |R_v|$. First note that $X \cap \tau_v' = \phi$, for otherwise $\delta(X)$ would contain an edge of capacity $|R_v|$. Now we consider the following two cases.

(i) $R_v \not\subset X$. Let $R_v^1, \ldots, R_v^q$ be the intervals of R_v that are contained in X. Then there are $|R_v| - \sum_j |R_v^j|$ edges of unit capacity incident to s^* in $\delta(X)$. Now, in $G[F_v] \setminus \{s\}$, X has q connected components, one containing each of $R_v^1, \ldots, R_v^q$. Since R_v is ε-smooth, Lemma 7.1.6 implies that the component of X containing R_v^j has at least $\varepsilon\ell^{-1}|R_v^j|$ edges in its boundary, and hence contributes a capacity of at least $\kappa_0\varepsilon\ell^{-1}|R_v^j| \geq |R_v^j|$ to $c(X)$. It follows that the cut condition holds in this case.

(ii) $R_v \subset X$. Now if there is some vertex u of τ_v' that does not lie within distance $r = \frac{1}{6}\varepsilon\kappa\lambda_6 \log n$ of some vertex in $\pi(X)$, then $G[F_v]\setminus X$ contains $B_r(u)$. By the semi–expansion of $G[B_{\lambda \log n}(v)]$, it follows that $|\delta(X)| \geq \alpha\bar{\alpha}^{1/2}r$, and hence $c(X) \geq \gamma_1 \log n \geq |R_v|$.

Otherwise, every vertex of τ_v' lies within distance r of some vertex of $\pi(X)$. We know that τ_v' consists of at least $\lambda_6 \log n$ vertices spaced κ apart on an ε-smooth cycle φ_v. Thus, we may choose vertices $u, u' \in \tau_v'$ whose distance, measured along φ_v, is $\frac{1}{2}\kappa\lambda_6 \log n = 3\varepsilon^{-1}r$. By the ε-smoothness of Q, we have $d_G(u, u') \geq 3r$; and hence there are two vertices of $\pi(X)$ that are at separated by a distance of at least r in G. By Lemma 7.1.6, it follows that $|\delta(X)| \geq \ell^{-1}r$, and hence

$$c(X) \geq \kappa_0\ell^{-1}r \geq \gamma_1 \log n \geq |R_v|.$$

■

In analyzing the approximation algorithm, we observe that only steps (ii) and (iii) produce paths in G — step (i) simply computes "global routes" in $\mathcal{N}$, and step (iv) concatenates paths already constructed. We therefore provide upper bounds on the congestion of the paths produced by steps (ii) and (iii).

Lemma 8.2.5 *The total congestion of the paths produced in step (ii) is*

$$O\left(\frac{\nu_{\mathcal{N}}^*}{\log n} + \rho^*\right) = O(\nu(\mathcal{T})).$$

Proof. By Theorem 8.1.1, the total demand of terminals assigned to the edge (v, w) in $\mathcal{N}$ is at most

$$b_0\nu_{\mathcal{N}}^* + \rho^* b_1 \log |E(\mathcal{N})| \leq b_0\nu_{\mathcal{N}}^* + \rho^* b_1 \log n.$$

Since the number of paths in $\mathcal{Z}_{vw}$ is at least $\lambda_6 \log n$, we can assign terminal pairs to paths in $\mathcal{Z}_{vw}$ so that none of these paths carries more than

$$\frac{b_0 \nu^*_{\mathcal{N}}}{\lambda_6 \log n} + b_1 \lambda_6^{-1} \rho^* + \rho^*$$

units of demand. ■

Lemma 8.2.6 *The total congestion of the paths produced in step (iii) is $O(\nu^*_F) = O(\nu(\mathcal{T}))$.*

Proof. By Lemma 8.2.4, any routing from $\psi^{-1}(v)$ to R_v can be converted into a routing from $\psi^{-1}(v)$ to τ'_v with only a constant–factor increase in the congestion. It follows that the minimum congestion of a ψ^{-1}–τ'_v routing is $O(\nu^*_F)$. Now, we produce paths whose congestion is within a constant factor of the minimum; and since every edge of G appears in only a constant number of sets of the form F_v, the congestion of the set of paths we produce in step (iii) is $O(\nu^*_F)$. ■

By the previous two lemmas, we have

Theorem 8.2.7 *The above algorithm produces a routing for the long connections in $\mathcal{T}$ whose congestion is within a constant factor of optimal.*

Finally, we discuss how to handle short connections. As this is completely analogous to what was done in Section 7.7, we will be somewhat brief. We choose a set of vertices M' which is maximal subject to the constraint that no two are within distance $\frac{1}{2}\lambda_1 \log n$ of each other. For each $v \in M'$, we let $\mathcal{T}'_v$ denote the set of terminal pairs with both ends in $B_{\lambda_1 \log n}(v)$; we run the above algorithm recursively in $G[B_{8\xi^2\lambda_1 \log n}(v)]$, with terminal pairs $\mathcal{T}'_v$. By the construction used to prove Lemma 7.6.1, it is easy to show that the minimum congestion of a routing of $\mathcal{T}'_v$ in $G[B_{8\xi^2\lambda_1 \log n}(v)]$ is within a constant factor of the minimum congestion of a routing of $\mathcal{T}'_v$ in G. Moreover, every edge of G appears in only a constant number of the sets $B_{8\xi^2\lambda_1 \log n}(v)$, and by the definition of M' every short connection belongs to some set $\mathcal{T}'_v$.

Thus, we are within a constant–factor of optimal in routing connections that are classified as “long” in the first recursive call. Connections that are classified as “short,” as in the previous chapter, belong to balls of radius $O(\log \log n)$; we can compute the optimum routing for them by brute force in one additional recursive call. We overlay all the paths produced in these three levels of recursion; this costs at most a factor three in the approximation ratio. Thus we have

Theorem 8.2.8 *Let G be a densely embedded graph, and $\mathcal{T}$ a set of terminal pairs. There is a polynomial–time algorithm that with high probability produces a routing of $\mathcal{T}$ in G whose congestion is within a constant factor of optimal.*

Chapter 9

Node–Disjoint Paths

My man, from sky to sky's so far,
We never crossed before;
Such leagues apart the world's ends are,
We're like to meet no more.
— *A.E. Housman,* A Shropshire Lad.

In all the previous chapters, the underlying constraint has been related to the notion of *edge–disjointness* — each edge can only carry a limited amount of demand. Here, we investigate some of the problems that arise when one instead requires paths that are *node–disjoint* — that is, paths such that no two pass through the same node.

Finding node-disjoint paths between designated pairs of vertices in a planar graph is fundamental to a number of routing and embedding problems. In particular, in many models of VLSI layout, effective layout strategies rely on heuristics for finding node–disjoint paths; many current routing software systems rely extensively on such an approach (see e.g. [91]). Node–disjointness constraints can arise as well in the context of virtual circuit routing in networks, when one requires paths to be node–disjoint for the sake of fault–tolerance or because of bottlenecks involving switches rather than links. At the same time, the basic node–disjoint paths problem is NP-hard even for the two–dimensional mesh [65], and essentially nothing is known in the way of reasonable approximation algorithms for the problem.

Although the node–disjoint and edge–disjoint paths problems are quite similar at the level of general graphs, their underlying structures are very different in the case of planar graphs, and even in the case of the two–dimensinoal mesh. This is true even in a technical sense; the approximation algorithm of Chapter 6 could be analyzed, for the most part, with respect to the *fractional* optimum. But this is far from true in the node–disjoint case, where simple examples show that the optimum fractional and integral solutions can be very different in value. In particular, one now encounters obstacles to the existence of paths that have nothing to do with saturated cuts; they can best be described as "topological" in nature. We will discuss this further in Section 9.1.

Thus, it appears that a new range of techniques must be explored in dealing with node–disjoint paths problems in planar graphs, and this will be our motivation in this chapter. In particular, we will concentrate on the objective functions $\alpha(\mathcal{T})$ and $\chi(\mathcal{T})$, defined now for the case of node–disjoint paths. It is interesting to note that for approximating the node–disjoint analogue of $\nu(\mathcal{T})$, the approach developed for the edge–disjoint case turns out to be sufficient; the algorithm of Chapter 8, with some small modifications, provides a constant–factor approximation for this problem in densely embedded graphs.

9.1 Node–Disjoint Paths on the Mesh

Let us discuss a very basic problem that arises in this setting. Recall that a set of terminal pairs is said to be a *permutation* if each vertex of the graph appears in exactly one pair. If G denotes the $n \times n$ two–dimensional mesh graph, and we are given a permutation $\mathcal{T}$ in which every pair needs to cross the center column, then the standard *bisection bound* says that $\alpha(\mathcal{T}) \leq n$, and hence $\chi(\mathcal{T}) \geq \frac{1}{2}n$.

The lack of understanding of these problems is such that the following has remained open. Again, let G denote the mesh.

> (1) For every permutation on G, does there exist a set of $\Omega(n^{1-\varepsilon})$ pairs that can be simultaneously routed on node–disjoint paths, for every $\varepsilon > 0$?
>
> (1′) Can every permutation on G be routed in $O(n^{1+\varepsilon})$ rounds, for every $\varepsilon > 0$?

Again, by way of contrast, these questions are straightforward to resolve in the affirmative when one only requires edge–disjoint paths.

These latter questions turn out to be closely related to open problems of Aggarwal, Klawe, Lichtenstein, Linial, and Wigderson [2] in the setting of *multi–layer* VLSI layout. In multi–layer VLSI layouts, typically a *pin* (a node, in our terminology) goes through several layers at the same location, and the routing among pins at each layer is planar. If a routing wire needs to change a layer, then a *via* (or *contact cut*) has to be made between two layers, and this usually affects the yield of the output chip (as well as affecting the general quality of the circuit). Consequently, it is practically useful to route as many connections on a single layer as possible (in order to reduce the vias); for more details see [2, 27, 96]. These constraints thus result directly in problems involving $\alpha(\mathcal{T})$ and $\chi(\mathcal{T})$, with *layers* now playing the role of "rounds."

One typically reduces the number of layers by introducing spacing between the terminals. In the model of [2, 27], for a parameter d, consider a $dn \times dn$ mesh G_d, with an $n \times n$ grid of terminals at a *uniform spacing* of d. By this we mean that terminals are placed at the set of nodes of G_d whose row and column numbers are multiples of d; thus, $d-1$ paths can now be routed between adjacent terminals. The goal in this model is to find the tightest trade–off possible between the spacing d and the number of layers required in the worst case.

Cutler and Shiloach [27] showed that spacing $d = O(n^2)$ suffices to route all pairs in a single layer; and [2] showed that this value of d was tight up to constant factors. In terms of a general trade-off between d and the minimum number of layers required, the techniques of [27, 2] are only sufficient to show that, for spacing d, $O(\left(\frac{n}{\sqrt{d}}\right)^{4/3})$ layers are sufficient. However, guided by $(1')$, a reasonable question was the following.

> $(2')$ With spacing d, are $O(\left(\frac{n}{\sqrt{d}}\right)^{1+\varepsilon})$ layers sufficient to route any permutation on the mesh, for every $\varepsilon > 0$?

This contains the result of [27] in the case $d = \Theta(n^2)$ and question $(1')$ in the case $d = 1$.

The main results of this chapter provide affirmative answers to questions (1), $(1')$, and $(2')$. Specifically, we prove

Theorem 9.1.1 *Every permutation on the mesh contains a set of $\Omega(n/\log n)$ terminal pairs that can be simultaneously routed on node–disjoint paths. Moreover, every permutation on the mesh can be routed in $O(n \log^2 n)$ rounds.*

Theorem 9.1.2 *With spacing d, every permutation on the mesh contains a set of $\Omega(n\sqrt{d}/\log n)$ pairs that can be routed in one layer, and the entire permutation can be routed in $O(nd^{-\frac{1}{2}}\log^2 n)$ layers.*

This strengthens the bounds of [27, 2] on the number of layers required for routing a permutation. Within the context of VLSI, where the number of layers is typically a fixed small number, the "dual" maximization versions of these theorems may in fact be of greater interest. Namely, these are the first algorithms that can provably extract from any permutation on the mesh a routable set of terminal pairs whose size is within polylogarithmic factors of the bisection bound. Moreover, the approach we develop is fundamentally different from that of Cutler and Shiloach [27], which was essentially the only previous method applicable in this setting.

We mentioned above that questions (1) and (1′) are easy in the case of edge–disjoint paths; in fact, one can construct a set of paths meeting the bisection bound using only one bend in each path. On the other hand, the paths in the Cutler–Shiloach construction, as well as our constructions in Theorems 9.1.1 and 9.1.2, use many bends. Thus, it is natural to ask how many rounds are required with a routing that uses only a constant number of bends. Getting $o(n^2)$ rounds is not immediate, and it appears that no previous construction in the literature achieves it. We show

Theorem 9.1.3 *Every permutation on the mesh can be routed in $O(n^{7/5})$ rounds, with each path in the routing using only* 6 *bends.*

Within the context of the above results, it is worth mentioning what we feel is probably the most natural open question here. The algorithm of Theorem 9.1.1 gives a routing of a set $\mathcal{T}$ for which the number of rounds used is close to the best possible in the worst case; however, it is not an *approximation algorithm*, since we do not have a result relating the number of rounds we require for a given set $\mathcal{T}$ to the optimum value $\chi(\mathcal{T})$. It would be very interesting to obtain such an algorithm, since it would appear to require new techniques for analyzing $\chi(\mathcal{T})$ in cases when the fractional solution gives essentially no information.

Some Basic Examples

For the remainder of this chapter, let G denote the $n \times n$ two–dimensional mesh.

Let us indicate some ways in which the node–disjoint paths problem on the mesh differs fundamentally from its edge–disjoint counterpart. First of all, letting $v_{i,j}$ denote the node at row i and column j of the mesh, consider the set $\mathcal{T}$ of consisting of all pairs of the form $(v_{i,1}, v_{n,i})$, $i = 1, \ldots, n$. This set of terminal pairs satisfies the usual *cut condition*, stating that there is no way to delete k nodes and separate $k+1$ pairs of terminals, and in fact the set $\mathcal{T}$ is routable in one round on edge–disjoint paths. However, since any pair of paths in a routing of $\mathcal{T}$ must cross, $\mathcal{T}$ requires n rounds when paths must be node–disjoint.

This example does not fundamentally rely on the terminals being positioned near the boundary of the mesh. Consider an infinite grid graph, let $p = m^2$ for some parameter m, and $\mathcal{T}$ denote the set of all pairs of the form $(v_{im+j,0}, v_{jm+i,p})$, with $1 \leq i, j \leq m$. Then it is not hard to show by a crossing number argument (using a lemma of [67]) that $\chi(\mathcal{T}) = \Theta(\sqrt{p})$, even though $\mathcal{T}$ is again routable in a single round using edge–disjoint paths.

As a final example, we show that while Theorem 9.1.3 implies that 6 bends are sufficient to route any permutation in $O(n^{7/5})$ rounds, there exist partial permutations that require $\Omega(n^2)$ rounds in any one–bend routing.

Proposition 9.1.4 *There exists a partial permutation $\mathcal{T}$ on G for which there is no one–bend routing using fewer than $\frac{1}{8}n^2$ rounds.*

Proof. Assume for simplicity that n is even; so $n = 2m$. Let $\mathcal{T}$ consists of all pairs of the form $(v_{i,j}, v_{m+j,m+i})$, where $1 \leq i, j \leq m$. Let

$$X = \{v_{i,j} : 1 \leq i \leq m < j \leq 2m\}$$

$$Y = \{v_{i,j} : 1 \leq j \leq m < i \leq 2m\}.$$

Then in any one–bend routing of $\mathcal{T}$, at least $\frac{1}{8}n^2$ of the paths pass though either X or Y, and every pair of such paths must cross at some node. ■

It is also worth noting that the algorithm of Schrijver [104] (see Section 2.3) can be applied to provide a polynomial–time algorithm for the related, but simpler, problem in which all terminals lie on the boundaries of a designated set of faces, and we require node–disjoint paths whose homotopies with respect to this set of faces is specified. Indeed, for the two–dimensional mesh, efficient algorithms for this "homotopic routing" problem were given by Cole and Siegel [25] and Leiserson and Maley [71] for the mesh; Schrijver's work can be viewed as a generalization of this work to arbitrary graphs on arbitrary surfaces. This approach based on homotopy actually provides a polynomial–time algorithm for the node–disjoint paths problem in planar graphs, when all terminals lie on the boundaries of a fixed number of faces — for in this case, there are only a polynomial number of sets of possible homotopies to try. Of course, in the problem we are considering, there are an exponential number of sets of possible homotopies (and the problem is NP-complete).

Sketch of the Algorithm

We will show that it is possible, with a constant–factor loss in the bound, to reduce the problem for an arbitrary set of terminals to one in which there are two $k \times k$ subsquares of G, denoted A and B, which are spaced $4k$ apart from one another; and every terminal pair has one end in A and the other in B. If there are p terminal pairs, we will show how to route $p/(k \log k)$ of them simultaneously; applying this process greedily gives the bound of Theorem 9.1.1.

The approach of Cutler and Shiloach [27] requires, in order to route q terminals in a single round, that the inter–terminal spacing be at least q. Using this, one can therefore only be sure of routing a set of size $p^{1/3}$, resulting in an overall bound of $n^{4/3}$ for $\chi(\mathcal{T})$.

We start by considering the following simple notion: suppose we tried choosing at most one terminal from each row of A and at most one from each row of B, and then joining as many of these terminals as possible in the region between A and B. There are a number of difficulties with this; in particular, there may be no way to choose terminals in this way so that nearly p/k can be joined in node–disjoint fashion. Let us call this above approach *row–selection* — one tries to pick a unique terminal from each row so that the selected terminals in A can be routed out of A on their rows, and then joined with their selected counterparts in B, in such a way that no paths cross.

Our main technique is the following generalization of row–selection: although there may be no good way to choose unique terminals from the rows of A and B, we show that there exists a "rotation" of A and a "rotation" of B, so that the resulting set of rows allows for such a good selection. Now, meshes are discrete objects, so the notion of rotation has to be defined carefully

— we use monotone random walks of varying biases as a basic way of "rotating" A and B.

Having chosen a candidate pair of rotations, we decompose A and B into *pseudo–rows*. The task of *pseudo–row selection* — choosing a unique terminal from each pseudo–row — is based on analyzing the bipartite graph H on the set of pseudo–rows, in which a pseudo-row of A is joined to a pseudo-row of B if there is some terminal pair with one end in each. We show that one can choose rotations of A and B so that this bipartite graph contains a large *monotone matching* — one in which no edges cross. The pairs corresponding to the edges of this matching can be routed in a single round.

With inter–terminal spacing of $d > 1$, a very similar approach is applicable. The main difference is that we can now afford to choose edge sets of the bipartite graph H which are not necessarily monotone matchings; we can handle a limited amount of non-monotonicity using the technique of Cutler–Shiloach within each pseudo–row.

9.2 Preliminary Facts

In this section, we develop some basic lemmas that will be useful in analyzing our algorithms. In Section 9.3, we consider the basic problem of a permutation on the mesh, and prove Theorem 9.1.1. In Section 9.4, we give our algorithm for routing with inter-terminal spacing in the model of [27, 2]. Finally, in Section 9.5, we discuss algorithms for routing with only a constant number of bends per path.

Lemma 9.2.1 *Let $x_1, \ldots, x_d$ be positive integers for which $\sum_i x_i = a$ and $\sum_i x_i^2 = b$. Then $d \geq a^2 b^{-1}$.*

Proof. $a^2 = (\sum_i x_i)^2 \leq d \sum_i x_i^2 = db$. ■

The following is easily proved by induction on b.

Lemma 9.2.2 *For non-negative integers a and b, one has*

$$\int_0^1 x^a (1-x)^b \, dx = (a+b+1)^{-1} \binom{a+b}{b}^{-1}.$$

Lemma 9.2.3 *Consider the following random experiment: we first choose a bias α for a coin uniformly from the interval $[0, 1]$, and then flip the coin n times. The probability that the number of heads is k is equal to $\frac{1}{n+1}$.*

Proof. The probability is equal to

$$\binom{n}{k} \int_0^1 \alpha^k (1-\alpha)^{n-k} \, d\alpha,$$

which by Lemma 9.2.2 is equal to $\frac{1}{n+1}$. ■

There is also a purely combinatorial proof of this, which we leave as an exercise for the reader.

Monotone Matchings

We make use of a notion that we call a *monotone matching*. Here we define and develop some basic facts about monotone matchings at a general level.

In this section, let H denote a bipartite graph with bipartition (U, V), $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$; and let e denote the number of edges of H. As usual, by a *matching* in H we mean a set of edges that have no endpoints in common, and we let $\mu(H)$ denote the size of the largest matching in H.

Definition 9.2.4 *A matching in H is said to* monotone *if it consists of edges $(u_{i_1}, v_{j_1}), \ldots, (u_{i_k}, v_{j_k})$ such that the sequence $i_1, \ldots, i_k$ is increasing, and the sequence $j_1, \ldots, j_k$ is either increasing or decreasing. We say the matching is* increasing *in the former case and* decreasing *in the latter.*

We give two lower bounds on the size of the largest monotone matching in H.

Lemma 9.2.5 *H has a monotone matching of size at least $\sqrt{\mu(H)}$.*

Proof. Let M be a matching of maximum size in H, and order its edges so that their endpoints in U have increasing indices. Consider the resulting sequence of indices in V. By a lemma due to Erdös and Szekeres [33], this sequence must have either an increasing sequence or a decreasing sequence of length at least $\sqrt{|M|} = \sqrt{\mu(H)}$. The edges corresponding to this sequence constitute a monotone matching. ■

Figure 9-1: Decomposing into pseudo-rows

Lemma 9.2.6 *H has an increasing monotone matching of size at least $e/2n$.*

Proof. Let M_k denote the set of all edges (u_i, v_j) for which $j = i + k \bmod n$. Let M'_k denote the subset of M_k in which $j \geq i$, and M''_k the subset of M_k in which $j < i$. Now, each M'_k and M''_k is a monotone matching, and since they are all disjoint, one has size at least $e/2n$. ∎

An example due to Don Coppersmith shows that for every $d \leq n$, there exist d-regular bipartite graphs on $2n$ nodes with no monotone matching of size greater than $O(\max(d, \sqrt{n}))$.

Finally, it is also useful to establish "covering" versions of these lemmas. We state these here.

Lemma 9.2.7 *The edges of H can be covered by $O(\Delta\sqrt{n})$ monotone matchings.*

Lemma 9.2.8 *The edges of H can be covered by $2n$ increasing monotone matchings.*

9.3 The bound for $\chi(\mathcal{T})$

For vertices $x, y \in G$, let $d(x, y)$ denote the shortest-path distance between them. As indicated above, we will first consider the following special case: there are $k \times k$ subsquares A and B of G, with $d(A, B) \geq 4k$, each terminal pair has one end in A and the other in B, and we must use paths that stay within $d(A, B)$ of $A \cup B$. Below we show how a bound for this special case this can be used to prove Theorem 9.1.1.

We represent A as a union of *pseudo-rows*, as follows. Let v_1 denote the lower left vertex of A. We choose $\alpha \in [0, 1]$ uniformly at random, and perform the following random walk for k time steps starting from v_1:

(i) With probability $1 - \alpha$, move one step to the right.

(ii) With probability α, move one step up and then one step to the right.

Let L_1^α denote the path traversed by this walk; note that it is contained in A. For $-k \leq i \leq k$, define L_i^α to be a copy of this path translated $(i-1)$ steps upward and then intersected with A. The sets L_i^α will be called *pseudo-rows.*

For $x \in A$, let $R_x^\alpha = \{i : x \in L_i^\alpha\}$. The following is immediate from the construction, but worth noting.

Lemma 9.3.1 *If $v \in A$ belongs to both L_i^α and L_j^α, then $|j-i| = 1$. Thus $|R_v^\alpha| \leq 2$ (v belongs to at most two of these sets); and in particular, $E|R_v^\alpha| \leq 2$.*

We say that $x, y \in A$ are *collinear* if there is some pseudo-row that contains them both; in this case we write $x \sim y$.

For a vertex $x \in G$, let $a(x)$ denote its row number and $b(x)$ denote its column number. For two vertices x and y, let $d_\infty(x,y)$ denote the L_∞ distance between them; that is,

$$d_\infty(x,y) = \max\left(|a(x)-a(y)|, |b(x)-b(y)|\right).$$

Lemma 9.3.2 *The probability that $x, y \in A$ are collinear is at most $\frac{2}{d_\infty(x,y)}$.*

Proof. Suppose without loss of generality that $b(x) \leq b(y)$. We have

$$\begin{aligned}
Pr(x \sim y) &= \int_0^1 Pr(x \sim y \mid \alpha = t)\, dt \\
&\leq \int_0^1 \sum_i Pr(x, y \in L_i^t)\, dt \\
&= \int_0^1 \sum_i Pr(x \in L_i^t) \cdot Pr(y \in L_i^t \mid x \in L_i^t)\, dt \\
&= \int_0^1 Pr(y \in L_i^t \mid x \in L_i^t) \cdot \sum_i Pr(x \in L_i^t)\, dt \\
&= \int_0^1 Pr(y \in L_i^t \mid x \in L_i^t) \cdot E|R_x^t|\, dt \\
&\leq 2\int_0^1 Pr(y \in L_i^t \mid x \in L_i^t)\, dt.
\end{aligned}$$

If it is not the case that $b(y) - b(x) \geq a(x) - a(y) \geq 0$, then this last integral is 0. Otherwise, it is simply the probability, over a uniformly distributed bias, that a given number of heads will come up when a coin with that bias is flipped $d_\infty(x,y)$ times. By Lemma 9.2.3, this is $\frac{1}{1+d_\infty(x,y)}$. The lemma follows. ∎

Lemma 9.3.3 *There is an absolute constant c' so that if there are p terminal pairs with ends in A and B, then there is a realizable subset of size at least $p/(c'k\log k)$.*

Proof. We first choose a random α and decompose A into pseudo-rows. We do the same for B, using a random β. Now by Lemma 9.3.1, we can choose half the pseudo-rows of A and half the pseudo-rows of B so that all pseudo-rows are disjoint and at least $q = p/4$ of the terminal pairs have both ends in these chosen pseudo-rows. Let $\mathcal{T}'$ denote a set of q such pairs.

Define a *collineation* to be a pair i, j such that s_i and s_j are collinear in A and t_i and t_j are collinear in B. We wish to bound the expected number of collineations among the pairs in $\mathcal{T}'$; this is simply the sum

$$\sum_{i,j} Pr(s_i \sim s_j) \cdot Pr(t_i \sim t_j).$$

This can be viewed as the dot product of two vectors of dimension $\binom{q}{2}$, one with coordinates $Pr(s_i \sim s_j)$ and the other with coordinates $Pr(t_i \sim t_j)$. We observe that this dot product will be maximized if $Pr(s_i \sim s_j) = Pr(t_i \sim t_j)$ for all pairs i, j. For the dot product is certainly maximized, once the lengths of the vectors is fixed, by having them parallel to one another; and in this case, if their lengths differ, one can increase the dot product by changing the coordinates of the shorter one to be equal to the longer one.

Thus, the expected number of collineations is bounded by the largest possible sum

$$\sum_{x,y\in S} Pr(x \sim y)^2,$$

where S is a subset of A of size q. We can write this sum as

$$\frac{1}{2}\sum_{x\in S}\sum_{y\in S\setminus\{x\}} Pr(x\sim y)^2 \le \frac{1}{2}\sum_{x\in S}\sum_{y\in S\setminus\{x\}} \frac{4}{d_\infty(x,y)^2},$$

with the inequality following from Lemma 9.3.2. This inner sum is maximized by having the vertices y fill in a ball centered at x, in which case its value is bounded by

$$\sum_{i=1}^{\frac{1}{2}\sqrt{q}} \frac{32i}{i^2} \leq 32 + 16 \ln q \leq c \ln q$$

for an absolute constant c. Thus, summing over all x, we see that the expected number of collineations in $\mathcal{T}'$ is at most $\frac{1}{2}cq \ln q$.

Thus, there exist α, β and decompositions of A and B into pseudo-rows for which there is a subset $\mathcal{T}'$ of q terminal pairs whose number of collineations is at most $\frac{1}{2}cq \ln q$. For this set $\mathcal{T}'$, let us build the following bipartite multigraph H. There is one vertex u_i on the left for each chosen pseudo-row A, and one vertex v_j on the right for each chosen pseudo-row B. We now put w_{ij} parallel edges from u_i to v_j if there are w_{ij} terminals with one end in the pseudo-row u_i and the other end in v_j. From H, we build a simple graph H' which contains an edge (u_i, v_j) iff $w_{ij} > 0$.

We claim that the number of edges of H' is at least $q/(2c \ln q)$. To see this, note that the number of collineations in $\mathcal{T}'$ is precisely

$$\sum_{i,j} \binom{w_{ij}}{2},$$

whence we have $\sum_{i,j} w_{ij}^2 \leq 2cq \ln q$. Since $q = |T'| = \sum_{i,j} w_{ij}$, the claim follows from Lemma 9.2.1.

Finally, the number of vertices in each half of the bipartition of H' is at most k, so by Lemma 9.2.6, H' has a monotone matching of size at least

$$\frac{q}{4ck \ln q} \geq \frac{p}{16ck \ln q} \geq \frac{p}{32ck \log k}.$$

We can route one terminal for each edge of the monotone matching, which implies the lemma with $c' = 32c$. Note that we need $d(A, B) = \Omega(k)$ in order to ensure that we correctly align the matched pseudorows of A with the corresponding pseudorows of B. ■

Lemma 9.3.4 *All terminal pairs with ends in A and B can be routed in $O(k \log^2 k)$ rounds.*

Proof. We produce the partition into realizable sets greedily, using Lemma 9.3.3. The bound follows since the number of remaining terminals goes down by a $(1 - \frac{c'}{k \log k})$ fraction for each realizable set we produce. ∎

Proof of Theorem 9.1.1. For an integer τ, let $\mathcal{T}_\tau$ denote the set of terminal pairs (s_i, t_i) for which $\frac{1}{2}\tau \leq d(s_i, t_i) \leq \tau$. We prove that $\mathcal{T}_\tau$ can be routed in $O(\tau \log^2 \tau)$ rounds; the overall bound on the number of rounds then follows since, for an absolute constant c'',

$$\begin{aligned}
\chi(\mathcal{T}) &\leq \sum_{i=0}^{\lceil \log n \rceil} \chi(\mathcal{T}_{2^i}) \\
&\leq \sum_{i=0}^{\lceil \log n \rceil} c'' 2^i \log^2(2^i) \\
&\leq 4c'' n \log^2 n.
\end{aligned}$$

To prove that $\chi(\mathcal{T}_\tau) = O(\tau \log^2 \tau)$, we show that $\mathcal{T}_\tau$ can be partitioned into a constant number of sets, and each of these sets can be partitioned into pairs of well–separated squares; we then invoke the algorithm of Lemma 9.3.4 on all the pairs in a single set simultaneously.

Choose $\tau' = \frac{1}{16}\tau$, and define S_{ij} to be the $\tau' \times \tau'$ subsquare of G whose upper–left corner lies in row $i\tau'$ and column $j\tau'$. Each pair of squares $(S_{ij}, S_{\ell m})$ defines a subset of $\mathcal{T}_\tau$, consisting of those terminal pairs with one end in S_{ij} and the other in $S_{\ell m}$. We call such a pair *active* if it contains a terminal pair in $\mathcal{T}_\tau$. By definition, any active pair $(S_{ij}, S_{\ell m})$ satisfies

$$6 \leq |\ell - i| + |m - j| \leq 16.$$

Thus, by Lemma 9.3.4, we can route all the terminal pairs in the set defined by any active pair in $O(\tau \log^2 \tau)$ rounds.

We say that active pairs $(S_{ij}, S_{\ell m})$ and $(S_{i'j'}, S_{\ell' m'})$ *interfere* with one another if there is some vertex within distance τ of both $S_{ij} \cup S_{\ell m}$ and $S_{i'j'} \cup S_{\ell' m'}$. Let us build a graph $\mathcal{K}$ on the set of active pairs, joining two by an edge if they interfere with one another. This graph has constant degree, and hence can be colored with a constant number of colors. But we can use the same set of rounds for routing *all* the active pairs in a single color class, and thus $\chi(\mathcal{T}_\tau) = O(\tau \log^2 \tau)$.

The proof of the other statement of Theorem 9.1.1, that every permutation contains a routable set of size $\Omega(n/\log n)$, is strictly analogous. We first argue, using the interference graph $\mathcal{K}$ and Lemma 9.3.3 instead of Lemma 9.3.4, that $\mathcal{T}_\tau$ contains a routable set of size $\Omega(|\mathcal{T}_\tau|/\tau \log \tau)$. The bound now follows, since for some $\tau \in \{2^i : 0 \leq i \leq \lceil \log n \rceil\}$ we have $|\mathcal{T}_\tau| = \Omega(n\tau)$. ■

9.4 Terminals with Spacing

The proof of Theorem 9.1.2 is essentially that of Theorem 9.1.1, with one additional step added. As before, we can specialize to the case of two $kd \times kd$ squares A and B that contain p terminal pairs. To construct pseudo-rows, we first partition A and B into disjoint $d \times d$ subsquares, contract these subsquares, and then use the previous construction.

Lemma 9.4.1 *There is an absolute constant c' so that if there are p terminals with ends in A and B, then there is a realizable subset of size at least $p\sqrt{d}/(c'k \log k)$.*

Proof. We follow the proof of Lemma 9.3.3 up through the construction of the bipartite graph H' with at most k vertices in each half of its bipartition, and with at least $q/(c \ln q)$ edges. Now define a *slice* in H' to be a set of the form $U' \cup V'$, where U' is a set of $\sqrt{d}$ consecutive vertices on the left side of H' and V' is a set of $\sqrt{d}$ consecutive vertices on the right side. The crucial point is that there are only d edges of H' with both ends in a given slice, and so using the technique of [2, 27], we can route one terminal pair for each of these edges in a single round.

We now construct a graph H'' by identifying blocks of $\sqrt{d}$ consecutive vertices of H' on the left and on the right. (We can essentially think of edges of H'' as corresponding to particular slices of H'.) Let us give each edge (u^*, v^*) in H'' a weight equal to the number of edges of H' that have ends in the super-nodes u^* and v^*; thus these weights are integers between 0 and d. By the weighted analogue of Lemma 9.2.6, H'' has a monotone matching of weight at least

$$\frac{\text{weight}(H'')}{2|V(H'')|} = \frac{|E(H')|}{2|V(H'')|} \geq \frac{q\sqrt{d}\log k}{4ck \ln q} \geq \frac{p\sqrt{d}}{32ck \log k}.$$

We can route a number of terminals equal to the weight of any monotone matching in H'', as argued above; this implies the lemma. ■

Proof of Theorem 9.1.2. Again reducing to the case of $kd \times kd$ squares A and B, we use a greedy approach based on Lemma 9.4.1. Each time we reduce the number of remaining

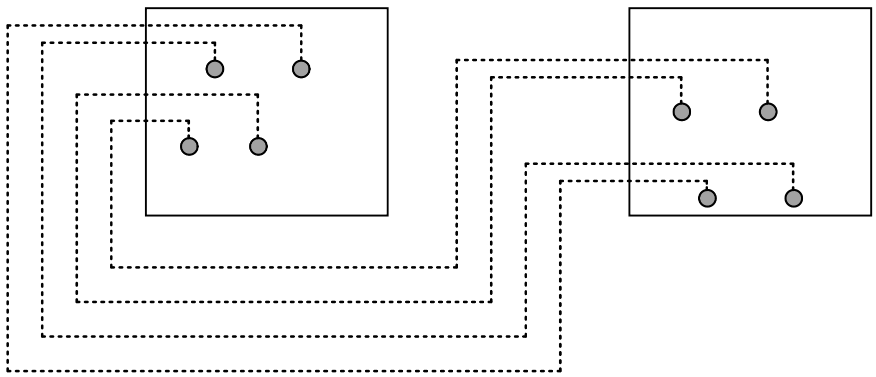

Figure 9-2: Routing with a constant number of bends

terminals by a factor of at least $(1 - \frac{c'\sqrt{d}}{k \log k})$; thus we route all terminal pairs in $O(kd^{-\frac{1}{2}} \log^2 k)$ rounds. ■

9.5 Constant-Bend Routing

As in the previous sections, let us first consider the case in which all terminal pairs have ends in $k \times k$ squares A and B, with $d(A, B) \geq 4k$. Here, we additionally assume that A and B are each at a distance of at least $\Omega(k^{7/10})$ from the boundary of the mesh. Let us suppose that the lowest column number of a node in A is less than or equal to the lowest column number of a node in B.

The main difficulty we encounter here is the following: since we wish to use only a constant number of bends, the approach based on pseudo–rows appears to be inherently inapplicable, as it involves partitioning A and B into paths with many bends. Thus, in this case, we build a graph H with a vertex u_i for each row of A, and a vertex v_j for each row of B. We give the edge (u_i, v_j) a weight w_{ij} equal to the number of terminal pairs with one end in each row.

To give a sense of the algorithm, we first mention a very easy algorithm for routing in $O(k^{3/2})$ rounds. This is as follows. First replace each edge (u_i, v_j) of H with w_{ij} parallel edges. Now by Lemma 9.2.7, the edges of H can be covered by $O(k^{3/2})$ monotone matchings. Moreover, we may assume that each matching in the cover contains at most $O(k^{1/2})$ edges. We claim that all the terminal pairs corresponding to a single one of these matchings can be routed in a single round. For if the matching is increasing, then the pairs can be routed by having each terminal leave A along its row through the east wall, and enter B along the matching row through the west wall; and if the matching is decreasing, then the pairs can be routed by having each terminal leave A along its row through the west wall and enter B along the matching row through the west wall. (See Figure 9-2 for a version of the latter case.) Moreover, since A and B are at least $\Omega(k^{1/2})$ distance from the boundary of the mesh, there is room for all the paths corresponding to a single matching to be routed in the region outside $A \cup B$.

To get a bound of $O(k^{7/5})$, we need a slight strengthening of this technique. We divide the edges into $\log k$ classes, by assigning each edge to the smallest power of 2 greater than its weight. Let H_w denote the subgraph of H consisting only of edges that are assigned the weight w, and let $\mathcal{T}_w$ denote the set of terminals corresponding to edges of H_w.

Lemma 9.5.1 *$\mathcal{T}_w$ can be routed in at most $2kw$ rounds, using at most four bends.*

Proof. Using Lemma 9.2.8, we can partition the edges of H_w into $2k$ (increasing) monotone matchings. Each edge of H_w corresponds to w terminals, so all terminals can be routed in $2kw$ rounds. It is easy to construct paths for these terminals using at most four bends. ∎

Lemma 9.5.2 *$\mathcal{T}_w$ can be routed in at most*

$$\max[O(k^{3/2}/w^{1/4}), O(k^{13/10})]$$

rounds, using at most six bends.

Proof. The maximum degree of a vertex in H_w is k/w, so we can partition the edges of H_w into k/w matchings. Let us say that a matching M is *well-spaced* if for edges (u_i, v_j) and (u_k, v_ℓ) of M, the difference between i and k is at least $\sqrt{w}$, and the distance between j and ℓ is at least $\sqrt{w}$. By an easy coloring argument, we can partition any matching in H_w into $O(\sqrt{w})$ well-spaced matchings, and thus we can partition the edges of H_w into $O(k/\sqrt{w})$ well-spaced matchings. Finally, each of these well-spaced matchings is only on $O(k/\sqrt{w})$ vertices, so by Lemma 9.2.7 we can partition each one into $O(\sqrt{k}/w^{1/4})$ monotone matchings.

Thus, we now have a total of $O(k^{3/2}/w^{3/4})$ monotone matchings covering the edges of H_w, and each has the property that the endpoints of its edges are mutually at distance $\sqrt{w}$. We use a different set of rounds for routing each monotone matching.

Now, how many rounds are needed for a single monotone matching M? (Recall that each edge of M represents up to w terminals.) Let A_i (resp. B_i) denote the set of vertices in the i^{th} row of A (resp. B). Since the edges of M are anchored in rows of A and B that are mutually $\sqrt{w}$ apart, we can assign to each edge $e = (u_i, v_j)$ a "band" of rows of width $\Omega(\sqrt{w})$ around row A_i and a similar band around row B_j. The pairs corresponding to an edge e are themselves a matching on the vertex set $A_i \cup B_j$. Thus these pairs can be partitioned, again by Lemma 9.2.7, into $O(\sqrt{w})$ monotone matchings, and each of *these* monotone matchings can be routed in a single round in the bands surrounding rows A_i and B_j.

Finally, we can do this simultaneously (i.e. using the same set of rounds) for all edges of a single well-spaced monotone matching, provided there is sufficient room between the squares A and B and the boundary of the mesh for routing all the paths. If there is sufficient room, then the number of rounds used is $O(k^{3/2}/w^{1/4})$. Otherwise, we must break up the rounds that use too many paths into sub-rounds that only route $O(k^{7/10})$ terminals at a time; the resulting bound is $O(k^{13/10})$ rounds. ■

Lemma 9.5.3 *The set of terminals with ends in A and B can be routed in $O(k^{7/5})$ rounds, using at most six bends.*

Proof. Let $r(\mathcal{T})$ denote the total number of rounds required for routing $\mathcal{T}$ with six bends. Then

$$\begin{aligned}
r(\mathcal{T}) &\leq \sum_{w=2^i} r(\mathcal{T}_w) \\
&\leq \sum_{w=2^i \leq k^{2/5}} r(\mathcal{T}_w) + \sum_{k^{2/5} \leq w=2^i \leq k^{4/5}} r(\mathcal{T}_w) + \sum_{w=2^i \geq k^{4/5}} r(\mathcal{T}_w) \\
&\leq O(k) \sum_{w=2^i \leq k^{2/5}} w + O(k^{3/2}) \sum_{k^{2/5} \leq w=2^i \leq k^{4/5}} w^{-1/4} + O(k^{13/10}) \sum_{w=2^i \geq k^{4/5}} 1.
\end{aligned}$$

Since all three of the terms in this last sum are easily seen to be $O(k^{7/5})$, the bound follows. ■

Proof of Theorem 9.1.3. We use the notion of an interference graph $\mathcal{K}$ on pairs of subsquares as in the proof of Theorem 9.1.1. The only change is that we can now only work with $k \times k$ squares that have either no row or no column within $O(k^{7/10})$ of the boundary. (It is enough to have this guarantee for either rows *or* columns, since we could just as well have built our graph H on the set of columns of one of the squares.) To handle this, we define G' to be the subgraph of G induced by all nodes within distance $O(n^{2/5})$ of the boundary, or within distance $O(n^{7/10})$ of a corner of the mesh. Since $|G'| = O(n^{7/5})$, we can use a different round for each terminal pair with an end in G'; we then run the algorithm above (using the interference graph $\mathcal{K}$) in $G \setminus G'$. $\blacksquare$

Chapter 10

Conclusion

... Given the already–secreted true experience of the regeneratively–evolving world–design effort against fire flood pestilence violent atmospheric disturbance and providing seventeen cubic feet of air per minute per person free of toxic or disagreeable odors or dust, we feel that metals broadly speaking and synthetics narrowly speaking will interlink into continuously improving world–around extra–corporeal networks, networks within which only individual man presents himself as an inherent island of physical discontinuity, sad to say, sad to say, physical discontinuity and torpor, total velocities of which known practices have proved inadequate to solve. ...

Quite extraordinary, said Emma, what did it mean?

Thank you, said the Dead Father, it meant I made a speech.

— *Donald Barthelme,* The Dead Father.

The focus of this work has been the development of approximation algorithms for disjoint paths problems, and along these lines there are a number of fundamental directions for further research. Clearly, the development of improved approximation guarantees is of interest; but the framework of approximation also provides us with a concrete way to elucidate structural properties of disjoint paths in graphs, in particular through their relation to fractional multicommodity flow. There are a number of optimization problems involving disjoint paths in graphs that we have not touched on here; for many of these, essentially nothing non–trivial is known about their combinatorial structure. The development of efficient approximation algorithms for these problems would undoubtedly be of value in the search for such structure.

With this in mind, we will use this chapter to outline a number of directions for further work. In particular, this will give us a chance to mention a number of basic NP-hard problems for which there appear to be no relevant algorithmic techniques currently available. In Section 10.1, we address the question of strengthening the approximation ratios presented here. In Section 10.2, we present some open problems in the on–line model, and also discuss some of the limitations of the model as it currently stands. Finally, in Section 10.3, we describe a number of possible extensions of this line of work to more complicated optimization problems.

Most of the open questions we will be discussing hold not only for the general unsplittable flow problem, but also for the maximum disjoint paths problem. For the sake of concreteness, we will phrase things (for the most part) in terms of the MDP; the extension of these questions to the UFP, when applicable, is straightforward.

10.1 Improved Approximation Ratios

Probably the most natural question raised by this work is the following: what is the best approximation ratio that can be obtained for the maximum disjoint paths problem in arbitrary graphs? The only algorithm we have analyzed here that is applicable in arbitrary graphs is the bounded greedy algorithm, which has an approximation ratio of $O(\max[\sqrt{|E(G)|}, \mathrm{diam}(G)])$. However, while the constant–factor approximation of Chapters 6 and 7 applies only to the class of densely embedded graphs, there is currently no evidence to indicate that constant–factor approximation algorithms should not exist for broader classes of graphs.

In particular, both of the following situations are possible.

(1) It is possible that there is an $\varepsilon > 0$ so that there is no polynomial–time $O(n^{\varepsilon})$–approximation algorithm for the maximum disjoint paths problem on an arbitrary n–node graph unless $\mathrm{P} = \mathrm{NP}$. Such strong lower bounds are known for other basic optimization problems such as finding a maximum independent set in an arbitrary graph (see e.g. [7]). However, all that is currently known for the maximum disjoint paths problem is that it is MAX-SNP-hard [44]; hence by the well–known result of Arora et al. [7], there is a constant $c > 1$ such that there is no polynomial–time c–approximation for the MDP unless $\mathrm{P} = \mathrm{NP}$.

It is worth noting, in light of the previous work discussed in Chapter 2, as well as the algorithms of Chapters 6 and 7, that the best approximation ratio obtainable for the maximum disjoint paths problem might conceivably be much better in the class of Eulerian graphs than in the class of all graphs. At present, however, we have no complexity–theoretic evidence to support this intuition.

(2) It is also possible that there is a constant–factor approximation for the maximum disjoint paths problem in an arbitrary graph. For Eulerian graphs, we do not have an example to refute the stronger possibility that, for any graph G with terminal pairs $\mathcal{T}$, the maximum fractional flow never exceeds the maximum size of a realizable subset of $\mathcal{T}$ by more than a constant factor. (For non-Eulerian graphs Theorem 2.4.4, which uses the *brick wall graph*, implies that there can be a large gap between these quantities.)

In relation to point (2) above, a number of interesting questions arise from considering a graph G with a set of terminal pairs $\mathcal{T}$ that is *fractionally* realizable. Essentially, the problem here is to determine what one can say about the realizability of $\mathcal{T}$ by disjoint paths. Questions related to this are discussed in a survey by Sebő [109], which also contains many references. We note first of all that the example of the brick wall graph from Chapter 2 shows that, in general, it is possible for the set $\mathcal{T}$ to be fractionally realizable, and for the maximum realizable subset of $\mathcal{T}$ to consist of a single pair of terminals.

However, as Seymour observed [112], things *seem* to be much better behaved if we consider the relation between fractional flow and *half–integral flow*. By the latter term we mean, as in Chapter 2, a fractional flow in which each terminal pair can only be routed on at most two flow paths, each of which is assigned a weight of $\frac{1}{2}$ or 1. Seymour's conjecture, that a fractionally realizable set of terminals can always be realized by a half–integral flow, was disproved by Lomonosov [76]. But, for example, we know of no evidence to refute the following possibility: every fractionally realizable set of terminals can be partitioned into a constant number of sets, each of which is realizable by a half–integral flow. And should this turn out not to be the case, one can ask the analogous question for a "$\frac{1}{c}$–integral flow," defined by the obvious analogy with half–integral flows. Thus, for example, do there exist absolute constants c_0 and c_1 such that any fractionally realizable set of terminals can be partitioned into c_0 sets, each of which realizable by a $\frac{1}{c_1}$–integral flow?

Any such result would imply a constant–factor approximation for the minimum congestion problem in an arbitrary (uniform–capacity) graph. To see this, we simply argue as follows. Take a fractional routing of $\mathcal{T}$ of minimum congestion. Partition $\mathcal{T}$ into c_0 sets, each of which has a $\frac{1}{c_1}$–integral routing of the same congestion. For each terminal pair (s_i, t_i), choose any of the (at most) c_1 paths assigned to it by this routing, and use this as the path for (s_i, t_i) in an unsplittable routing. The resulting unsplittable routing has congestion at most $c_0 c_1$ times the fractional optimum.

Before leaving the subject of improved approximation ratios, it is worth highlighting two more concrete problems that have already appeared in earlier chapters.

Routing in Rounds. For a set $\mathcal{T}$ of terminal pairs on the two–dimensional mesh G, is there a constant–factor approximation for the minimum partition of $\mathcal{T}$ into rounds? The result of Chapter 6 provides an $O(\log n)$–approximation; an improved approximation ratio would be interesting in particular because it presumably would have to use a technique not based on the greedy set cover algorithm.

A Tight Bound for Single–Source Unsplittable Flow. As discussed in Chapter 3, we would like to know whether every fractionally feasible single–source flow problem has an unsplittable routing of congestion at most 2. If so, it would show that the following trivial example gives (asymptotically) the worst possible gap: Let G consist of vertices s and t, with k parallel edges of unit capacity between them; let $\mathcal{D}$ consist of $k+1$ copies of the vertex t, with each terminal in $\mathcal{D}$ having demand $1 - \frac{1}{k+1}$. More generally, is the worst possible gap at most $1 + \rho^*$, where ρ^* as usual denotes the maximum demand of a terminal in $\mathcal{D}$?

10.2 The On–Line Model

The on–line model, in which terminal pairs arrive over time and must be processed immediately, is designed to capture the issues arising in "dynamic" problems in network routing, where one must establish routes without knowledge of the future. As such, it is of value to investigate ways in which the model can best represent such problems.

Currently, for example, the objective function being used is simply the total demand routed. But in many of the applications in which these problems arise, connections have varying priority;

and one does not want to route a low–priority connection if it will block a later, high–priority, connection — even if this high–priority connection has a smaller amount of demand. It is not clear how best to incorporate this into the on–line model, or how the design of on–line algorithms would be affected by such a constraint.

At a more general level, there is the question of *fairness* in routing connections. We observe that many of the current randomized on–line algorithms — including some of the algorithms discussed here — have performance guarantees that hold *in expectation.* That is, the expected amount of demand that they route is close to optimal. As a result, it is possible for these algorithms to choose arbitrarily to reject a large fraction of the terminal pairs and not be heavily penalized in the objective function. One can argue that it would be desirable to modify the on–line model so that such algorithms are not viable, and there are several possible ways to do this. One way would be to try defining some explicit notion of *fairness* to connection requests, that on–line algorithms must observe. Alternately, it is possible that one would obtain much more robust randomized on–line algorithms if one simply required approximation ratios to hold *with high probability.* Namely, an on–line c–approximation would be required to have the property that with probability $1 - o(1)$, it is within $2c$ of optimal. (Here the $o(1)$ term could hide a dependency on the input size.)

We note that this qualitative distinction between expected and high–probability performance guarantees has not typically arisen for randomized *off–line* algorithms. For example, the off–line algorithm we presented in Chapters 6 and 7 has, in its basic version, a performance guarantee that holds in expectation; but by running it $\Theta(k)$ times independently, it can be made to hold with probability $1 - 2^{-k}$. The point is that on–line algorithms, by definition, cannot be run many times independently.

One additional question is whether the "worst–case" nature of the on–line model is too strong to be representative of the types of traffic patterns that arise in practice. A possibility worth investigating is the requirement that the input be chosen subject to some underlying probability distribution, rather than by an arbitrary adversary that operates with knowledge of the algorithm's behavior. This has been suggested within the context of on–line algorithms in general by a number of researchers (see e.g. Raghavan [92]); and it has been investigated within the framework of on–line routing by Kamath, Palmon, and Plotkin [54]. This latter paper obtains very strong approximation results under the assumption that requests for connections between each pair of vertices arrive subject to an unknown Poisson process, and that each request only lasts for a period of time that is determined by an exponentially distributed random variable.

If one believes that the worst–case on–line model in its full generality is too pessimistic, but that one does not want to design algorithms that rely crucially on probabilistic assumptions about the input, then it makes sense to consider the notion of "hybrid" on–line models. Thus, for example, it would be interesting to explore models in which the traffic pattern between all pairs of vertices is specified by a Poisson process; but for some small number k, an adversary can disregard this probabilistic assumption for k of the pairs of vertices, and generate connection requests arbitrarily. This is related to the notion of "k–resilient" algorithms — those which must continue to operate correctly even in the presence of up to k "failures." (In this case, a failure would correspond to a violation of one of the probabilistic assumptions about the distribution of input.) It also captures the notion of a network traffic pattern that has been modeled probabilistically, but which is susceptible to a limited number of large, unpredictable variations.

An On-Line Single-Source Problem. Finally, it is worth mentioning one interesting open question that arises in the "pure" on–line model that we have concentrated on in this work. This is simply the on–line, single–source maximum disjoint paths problem, which we can define in more detail as follows. Let G be an arbitrary undirected graph, s a vertex of G, and $\mathcal{D}$ a set of vertices in G. We wish to design an on–line algorithm that is given G and s initially; it receives the terminals in $\mathcal{D}$ one at a time, and must construct disjoint paths from s to as many of them as possible. (As usual, it must irrevocably route or reject each terminal in $\mathcal{D}$ when it arrives.)

We do not know of any reasonable on–line algorithm for this problem even when we are guaranteed that the entire set $\mathcal{D}$ can be linked to s by edge–disjoint paths. This is of particular interest by way of contrast with standard "augmenting paths" algorithms for constructing single–source disjoint paths — in the current setting, approaches based on re–routing are explicitly ruled out.

10.3 Extensions

Finally, we discuss three directions for further work, of which only the last was covered at all in the previous chapters.

Disjoint Paths of Bounded Length. Many of the constraints that arise in routing problems put upper bounds on the maximum *length* of any path used in the solution. This can have the effect of making disjoint paths problems that were otherwise tractable NP-hard.

Probably the most basic example of this is the following. Let G be an undirected graph, with vertices s and t, and let k and L be natural numbers. The problem is to decide whether there exist k mutually edge–disjoint s–t paths, each of which uses at most L edges. This is NP-hard for $k = 2$ and variable L [73], and for $L = 5$ and variable k [52]. Related to this is the well–known minimum–cost flow problem, which is solvable in polynomial time (see [32, 115, 48]) For our purposes, we can view minimum–cost flow as addressing the following problem: for a given k, find k edge–disjoint s–t paths in G whose *total* length is minimum.

Thus, a fundamental open problem concerns the following "max-min" objective function. Given a graph G, vertices s and t, and a number k, find a set of k edge–disjoint s–t paths whose maximum length is minimum. If we compute a minimum–cost flow (on k paths) from s to t, then it is easy to see that this provides a k–approximation for the above problem. Thus we ask: is there a $o(k)$–approximation for this problem? Work of Lovász, Neumann–Lara, and Plummer [78] and Galil and Yu [40] is related to these questions.

There are also some basic questions related to the "min-sum" objective function. We noted in Chapter 2 that work of Robertson and Seymour [102] provides a polynomial–time algorithm for deciding whether a set $\mathcal{T}$ of terminal pairs is realizable, for fixed $|\mathcal{T}|$. Dean [28] and Galil and Yu [40] ask whether the following is also decidable in polynomial time, again for fixed $|\mathcal{T}|$: can $\mathcal{T}$ be realized using paths of total length at most L, for some given value of L?

Re-routing. In many network routing problems, connections that are already established cannot be "re–routed" without incurring a great cost; this fact provides much of the justification for the on–line model of routing. Consequently, it could be interesting to look at optimization problems in which some form of this "re–routing cost" is the objective function.

We state one such problem here, which we have chosen because it does not seem to be hopelessly difficult. Let G be graph, s a vertex of G, and $\mathcal{D}$ a subset of the vertices of G, such that $\mathcal{D}$ can be linked to s by edge–disjoint paths. Let us partition $\mathcal{D}$ into sets A and B, and let us choose a set $\mathcal{P}$ of edge–disjoint paths from s to the vertices in A. Thus, the vertices in A are terminals that have already been routed; the vertices in B are the terminals that have yet to be routed. The problem is to find a set of edge–disjoint paths from s to $\mathcal{D} = A \cup B$, such that as many of the paths as possible in the intermediate routing $\mathcal{P}$ are preserved.

We do not known whether this problem is NP-complete; though if this should turn out to be the case, there is still a natural approximation problem to be dealt with.

Node–Disjointness Constraints. Finally, we recall the main issue addressed in Chapter 9 — the design of approximation algorithms for *node–disjoint* paths problems. The algorithm presented in Chapter 9 provides an absolute performance guarantee that holds for all permutations on the two–dimensional mesh; however, it is not a proper approximation algorithm since it does not provide a solution that is close to optimal for *every* set of terminal pairs on the mesh. Thus the natural open question is to find an approximation algorithm for the node–disjoint paths problem, even on the two–dimensional mesh, that is within a polylogarithmic factor of optimal on every set of terminal pairs.

Designing approximation algorithms for problems with node–disjointness constraints is particularly interesting because it requires the development of techniques that do not rely solely on the fractional solution; we have seen that the fractional and integral optima can be far apart for these problems. This is an interesting issue for approximation algorithms in general — good approximation ratios are linked, an overwhelming majority of the time, to a corresponding bound for a fractional version of the problem being approximated. Exploring methods that are applicable even when this link to the fractional solution is broken promises to be an interesting direction for future work.

Bibliography

[1] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswami, B. Schieber, M. Sudan, "Efficient routing and scheduling algorithms for optical networks," *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, 1994, pp. 412–423.

[2] A. Aggarwal, M. Klawe, D. Lichtenstein, N. Linial, A. Wigderson, "Multi-layer grid embeddings," *Proc. 26th IEEE Symp. on Foundations of Computer Science*, 1985.

[3] A. Aggarwal, M. Klawe, D. Lichtenstein, N. Linial, A. Wigderson, "A lower bound on the area of permutation layouts," *Algorithmica*, 6(1991), pp. 241–255.

[4] A. Aggarwal, M. Klawe, P. Shor, "Multi-layer grid embeddings for VLSI," *Algorithmica*, 6(1991), pp. 129–151.

[5] A. Aggarwal, J. Kleinberg, D. Williamson, "Node-Disjoint Paths on the Mesh, and a New Trade-Off in VLSI Layout," *Proc. 28th ACM Symp. on Theory of Computing*, 1996.

[6] N. Alon, P.D. Seymour, R. Thomas, "A separator theorem for non–planar graphs," *J. American Math. Soc.*, 3(1990), pp. 801–809.

[7] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, "Proof verification and the hardness of approximation problems," *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, 1992.

[8] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin and O. Waarts, "On-line load Balancing with applications to machine scheduling and virtual circuit routing" *Proc. 25th ACM Symp. on Theory of Computing*, 1993. pp. 623-631.

[9] Y. Aumann, Y. Rabani, "Improved bounds for all-optical routing," *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms*, 1995, pp. 567–576.

[10] B. Awerbuch, Y. Azar, S. Plotkin, "Throughput-competitive online routing," *Proc. 34th IEEE Symp. on Foundations of Computer Science*, 1993, pp. 32–40.

[11] B. Awerbuch, Y. Bartal, A. Fiat, A. Rosén, "Competitive non-preemptive call control," *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms.*

[12] B. Awerbuch, R. Gawlick, F.T. Leighton, Y. Rabani, "On-line admission control and circuit routing for high performance computing and communication," *Proc. 35th IEEE Symp. on Foundations of Computer Science*, 1994, pp. 412–423.

[13] B. Awerbuch, F. T. Leighton, "A simple local–control approximation algorithm for multicommodity flow," *Proc. 34th IEEE Symp. on Foundations of Computer Science*, 1993, pp. 459–469.

[14] B. Awerbuch, F. T. Leighton, "Improved approximations for the multi–commodity flow problem and local competitive routing in networks," *Proc. 26th ACM Symp. on Theory of Computing*, 1994, pp. 489–496.

[15] Y. Bartal, A. Fiat, S. Leonardi, "Lower bounds for on-line graph problems with application to on-line circuit and optical routing," *Proc. 28th ACM Symp. on Theory of Computing*, 1996.

[16] S. Ben-David, A. Borodin, R. Karp, G. Tardos, A. Wigderson, "On the power of randomization in on-line algorithms," *Proc. 22nd ACM Symposium on Theory of Computing*, 1990, pp. 379–386.

[17] A. Blum, A. Fiat, H. Karloff, Y. Rabani, manuscript.

[18] B. Bollobas, *Extremal Graph Theory*, Academic Press, 1978.

[19] A. Broder, A. Frieze, S. Suen, E. Upfal, "Optimal construction of edge-disjoint paths in random graphs," *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, 1994, pp. 603–612.

[20] A. Broder, A. Frieze, E. Upfal, "Existence and construction of edge-disjoint paths on expander graphs," *SIAM J. Computing*, 23(1994), pp. 976–989.

[21] A. Broder, A. Frieze, E. Upfal, manuscript.

[22] Y.M. Chen, G. Sasaki, "Irregular torus networks: deadlock avoidance and throughput analysis," *Proc. INFOCOM*, 1993, pp. 312–320.

[23] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, *Annals of Math. Stat.*, 23(1952), pp. 493–509.

[24] V. Chvátal, "A greedy heuristic for the set covering problem," *Mathematics of Operations Research*, 4(1979), pp. 233–235.

[25] R. Cole, A. Siegel, "River routing every which way but loose," *Proc. 25th IEEE Symp. on Foundations of Computer Science*, 1984.

[26] S. Cosares, I. Saniee, "An optimization problem related to balancing loads on SONET rings," *Telecommunications Systems*, 3(1994), pp. 165–181.

[27] M. Cutler, Y. Shiloach, "Permutation layout," *Networks*, 8(1978), pp. 253–278.

[28] N. Dean, "Open problems," in *Graph Structure Theory*, N. Robertson, P.D. Seymour, eds., AMS Press, 1991.

[29] G. Ding, A. Schrijver, P.D. Seymour, "Disjoint paths in a planar graph — a general theorem," *SIAM J. Discrete Math*, 5(1992), pp. 112–116.

[30] E.A. Dinic, "Algorithm for solution of a problem of maximum flow in networks with power estimation," *Soviet Math. Dokl.* 11(1970), pp. 1277–1280.

[31] J. Edmonds, "Minimum partition of a matroid into independent subsets,", *J. Res. Nat. Bur. Standards B*, 69(1965), pp. 67–72.

[32] J. Edmonds, R.M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM*, 19(1972), pp. 248–264.

[33] P. Erdös, G. Szekeres, "A combinatorial problem in geometry," *Compositio Math.*, 2(1935), pp. 463–470.

[34] L.R. Ford, D.R. Fulkerson, "Maximal flow through a network," *Canadian J. Math*, 8(1956), pp. 399–404.

[35] L.R. Ford, D.R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[36] S. Fortune, J. Hopcroft, J. Wyllie, "The direcetd subgraph homeomorphism problem," *Theoretical Computer Sci.*, 10(1980), pp. 111–121.

[37] A. Frank, "Edge-disjoint paths in planar graphs," *J. Comb. Theory Ser. B*, 39(1985), pp. 164–178.

[38] A. Frank, "On disjoint homotopic paths in the plane," in *Polyhedral Combinatorics*, W. Cook, P.D. Seymour, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science (vol. 1), 1990, pp. 163–170.

[39] A. Frank, "Packing paths, cuts, and circuits — a survey," in *Paths, Flows, and VLSI-Layout*, B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, Eds., Berlin: Springer-Verlag, 1990, pp. 49–100.

[40] Z. Galil, X. Yu, "Short length versions of Menger's theorem," *Proc. 27th ACM Symp. on Theory of Computing*, 1995.

[41] J. Garay, I. Gopal, "Call preemption in communication networks," *Proc. INFOCOM*, 1992, pp. 1043–1050.

[42] J. Garay, I. Gopal, S. Kutten, Y. Mansour, M. Yung, "Efficient on–line call control algorithms," *Proc. 2nd Israel Conference on Theory of Computing and Systems*, 1993.

[43] M. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[44] N. Garg, V. Vazirani, M. Yannakakis, "Primal-dual approximation algorithms for integral flow and multicut in trees, with applications to matching and set cover," *Proc. International Colloq. on Automata, Languages, and Programming*, 1993, pp. 64–75.

[45] N. Garg, V. Vazirani, M. Yannakakis, "Approximate max-flow min-(multi)cut theorems and their applications," *Proc. 25th ACM Symp. on Theory of Computing*, 1993, pp. 698–707.

[46] R. Gawlick, C. Kalmanek, K.G. Ramakrishnan, "On–line routing for permanent virtual circuits," *Proc. INFOCOM*, 1995, pp. 278–288.

[47] J. Gilbert, J. Hutchinson, R.E. Tarjan, "A separation theorem for graphs of bounded genus," *J. Algorithms*, 5(1984), pp. 391-407.

[48] A. Goldberg, É, Tardos, R.E. Tarjan, "Network flow algorithms," in *Paths, Flows, and VLSI-Layout*, B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, Eds., Berlin: Springer-Verlag, 1990, pp. 101–164.

[49] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer–Verlag, 1987.

[50] D. Hochbaum, D. Shmoys, "Using dual approximation algorithms for scheduling problems: theoretical and practical results," *Journal of the ACM*, 34(1987), pp. 144–162.

[51] W. Hoeffding, "On the distribution of the number of successes of independent trials," *Annals of Math. Stat.*, 27(1956), pp. 713–721.

[52] A. Itai, Y. Perl, Y. Shiloach, "The complexity of finding maximum disjoint paths with length constraints," *Networks*, 12(1982), pp. 277–286.

[53] D.S. Johnson, "Approximation algorithms for combinatorial problems," *J. Computer and System Sci.*, 9(1974), pp. 256–278.

[54] A. Kamath, O. Palmon, S. Plotkin, "Routing and Admission Control in General Topology Networks with Poisson Arrivals," *Proc. 7th ACM-SIAM Symp. on Discrete Algorithms*, 1996.

[55] N. Karmarkar, "A new polynomial–time algorithm for linear programming," *Combinatorica*, 4(1984), pp. 373–396.

[56] R.M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, R.E. Miller, J.W. Thatcher, Eds., New York: Plenum Press, 1972. pp. 85–103.

[57] R.M. Karp, F.T. Leighton, R. Rivest, C. Thompson, U. Vazirani, V. Vazirani, "Global wire routing in two–dimensional arrays," *Algorithmica*, 2(1987), pp. 113–129.

[58] L. Khachiyan, "A polynomial algorithm for linear programming," *Soviet Math. Dokl.*, 244(1979), pp. 1093–1096.

[59] M. Klawe, F.T. Leighton, "A tight lower bound on the area of planar permutation layouts," *SIAM J. Discrete Math.*, 5(1992), pp. 558–563.

[60] P. Klein, A. Agrawal, R. Ravi, S. Rao, "Approximation through multicommodity flow," *Proc. 31st IEEE Symp. on Foundations of Computer Science*, 1990, pp. 726–737.

[61] J. Kleinberg, S. Rao, É. Tardos, "Routing with low congestion in densely embedded graphs," manuscript.

[62] J. Kleinberg, R. Rubinfeld, "Short paths in expander graphs," submitted for publication.

[63] J. Kleinberg, É. Tardos, "Approximations for the Disjoint Paths Problem in High–Diameter Planar Networks," *Proc. 27th ACM Symp. on Theory of Computing*, 1995, pp. 26–35.

[64] J. Kleinberg, É. Tardos, "Disjoint Paths in Densely Embedded Graphs," *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 1995, pp. 52–61.

[65] M.E. Kramer, J. van Leeuwen, "The complexity of wire routing and finding the minimum area layouts for arbitrary VLSI circuits," *Advances in Computing Research 2: VLSI Theory*, F.P. Preparata, Ed., London: JAI Press, 1984. pp. 129–146.

[66] R. Krishnan, N.F. Maxemchuk, "Is there life beyond linear toplogies? A comparison of DQDB and the Manhattan Street Network," *Proc. INFOCOM*, 1993, pp. 690–698.

[67] F.T. Leighton, *Layouts for the Shuffle–Exchange Graph and Lower Bound Techniques for VLSI*, Ph.D. Thesis, MIT Math Dept., 1981

[68] F.T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, S. Tragoudas, "Fast approximation algorithms for the multicommodity flow problem," *Proc. 23rd ACM Symp. on Theory of Computing*, 1991, pp. 101–111.

[69] F.T. Leighton, S. Rao, "An appoximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms," *Proc. 29th IEEE Symp. on Foundations of Computer Science*, 1988.

[70] F.T. Leighton, S. Rao, "Circuit switching: a multicommodity flow based approach," *Proc. 9th International Parallel Processing Symposium*, 1995.

[71] C. Leiserson, F.M. Maley, "Algorithms for routing and testing routability of planar VLSI layouts," *Proc. 17th ACM Symp. on Theory of Computing*, 1985.

[72] J.K. Lenstra, D.B. Shmoys, É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Proc. 28th IEEE Symp. on Foundations of Computer Science*, 1987.

[73] C. Li, T. McCormick, D. Simchi–Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Appl. Math.*, 26(1990), pp. 105–115.

[74] N. Linial, E. London, Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Proc. 35th IEEE Symp. on Foundations of Computer Science*, 1994.

[75] R. Lipton, R. Tarjan, "A separator theorem for planar graphs," *SIAM J. Applied Math.* 36(1979), pp. 177–189.

[76] M. Lomonosov, "Combinatorial approaches to multiflow problems," *Discrete Appl. Math.*, 11(1985), pp. 1–94.

[77] L. Lovász, "On the ratio of optimal integral and fractional covers," *Discrete Math.*, 13(1975), pp. 383–390.

[78] L. Lovász, V. Neumann-Lara, M. Plummer, "Mengerian theorems for paths of bounded length," *Periodica Mathematica Hungarica*, 9(1978) pp. 269–276.

[79] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM J. Computing*, 15(1986), pp. 1036–1053.

[80] W. Massey, *Algebraic Topology: An Introduction*, New York: Springer-Verlag, 1967.

[81] N.F. Maxemchuk, "Regular mesh topologies in local and metropolitan area networks," *AT&T Technical Journal*, 64(1985), pp. 1659–1685.

[82] K. Menger, "Zur allgemeinen Kurventheorie," *Fundam. Math.*, 19(1927), pp. 96–115.

[83] M. Middendorf, F. Pfeiffer, "On the complexity of the disjoint paths problem," *Combinatorica*, 13(1993). pp. 97–107.

[84] H. Okamura, P. Seymour, "Multicommodity flows in planar graphs," *J. Comb. Theory Ser. B*, 31(1981), pp. 75–81.

[85] H. Okamura, "Multicommodity flows in graphs," *Discrete Applied Mathematics*, 6(1983), pp. 55–62.

[86] D. Peleg, E. Upfal, "Disjoint paths on expander graphs," *Combinatorica*, 9(1989), pp. 289–313.

[87] H. Perfect, "Applications of Menger's graph theorem," *J. Math. Analysis Appl.* 22(1968), pp. 96–111.

[88] S. Plotkin, D. Shmoys, É. Tardos, "Fast approximation algorithms for fractional packing and covering problems," *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, 1991.

[89] S. Plotkin, É. Tardos, "Improved bounds on the max-flow min-cut ratio for multicommodity flows," *Proc. 25th ACM Symp. on Theory of Computing*, 1993, pp. 691–697.

[90] M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, Ellis Horwood Ltd., 1993.

[91] W.R. Pulleyblank, "Two Steiner tree packing problems," invited talk at ACM Symposium on Theory of Computing, 1995.

[92] P. Raghavan, "A statistical adversary for on–line algorithms," in *On-Line Algorithms*, D. Sleator and L. McGeoch, Eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science (vol. 7), 1992.

[93] P. Raghavan, "Probabilistic construction of deterministic algorithms: approximating packing integer programs," *J. Computer and System Sciences*, 37(1988), pp. 130–143.

[94] P. Raghavan, C.D. Thompson, "Randomized rounding: a technique for provably good algorithms and algorithmic proofs," *Combinatorica*, 7(1987), pp. 365–374.

[95] P. Raghavan, E. Upfal, "Efficient all-optical routing," *Proc. 26th ACM Symp. on Theory of Computing*, 1994, pp. 134–143.

[96] R. Rivest, "The placement and interconnect system," *Proc. 19th Design and Automation Conference*, 1982.

[97] N. Robertson, P.D. Seymour, "Graph Minors II. Algorithmic aspects of tree–width. *J. Algorithms*, 7(1986), pp. 309–322.

[98] N. Robertson, P.D. Seymour, "Graph Minors VI. Disjoint paths across a disc," *J. Combinatorial Theory Ser. B*, 41(1986), pp. 115–138.

[99] N. Robertson, P.D. Seymour, "Graph Minors VII. Disjoint paths on a surface," *J. Combinatorial Theory Ser. B*, 45(1988), pp. 212–254.

[100] N. Robertson, P.D. Seymour, "Graph Minors IX. Disjoint crossed paths," *J. Combinatorial Theory Ser. B*, 49(1990), pp. 40–77.

[101] N. Robertson, P.D. Seymour, "Graph Minors XI. Circuits on a surface," *J. Combinatorial Theory Ser. B*, 60(1994), pp. 72–106.

[102] N. Robertson, P.D. Seymour, "Graph Minors XIII. The disjoint paths problem," *J. Combinatorial Theory Ser. B*, 63(1995), pp. 65–110.

[103] A. Schrijver, "The Klein bottle and multicommodity flow," *Combinatorica* 9(1989), pp. 375–384.

[104] A. Schrijver, "Disjoint circuits of prescribed homotopies in a graph on a compact surface," *J. Combinatorial Theory Ser. B*, 51(1991), pp. 127–159.

[105] A. Schrijver, "Disjoint homotopic paths and trees in a planar graph," *Discrete and Computational Geometry*, 6(1991), pp. 527–574.

[106] A. Schrijver, "Finding k disjoint paths in a directed planar graph," *SIAM J. Computing*, 23(1994), pp. 780–799.

[107] A. Schrijver, personal communication, 1995.

[108] A. Schrijver, P.D. Seymour, P. Winkler, "The ring loading problem," submitted for publication.

[109] A. Sebő, "The cographic multiflow problem: an epilogue," in *Polyhedral Combinatorics*, W. Cook, P.D. Seymour, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science (vol. 1), 1990, pp. 203–234.

[110] P.D. Seymour, "Directed circuits on a torus," *Combinatorica*, 11(1991), pp. 261–273.

[111] P.D. Seymour, "Disjoint paths in graphs," *Discrete Math.*, 29(1980), pp. 293–309.

[112] P.D. Seymour, "On odd cuts and planar multicommodity flows," *J. London Math. Soc.*, 42(1981), pp. 178–192.

[113] Y. Shiloach, "A polynomial solution to the undirected two paths problem," *Journal of the ACM*, 27(1980), pp. 445–456.

[114] D. Sleator, R. Tarjan, "Amortized efficiency of list update and paging rules," *Comm. ACM*, 23(1985), pp. 202–208.

[115] É. Tardos, "A strongly polynomial minimum–cost circulation algorithms," *Combinatorica*, 5(1985), pp. 247–255.

[116] C. Thomassen, "2–linked graphs," *European J. Combinatorics*, 1(1980), pp. 371–378.

[117] T. Todd, "The token grid: multidimensional media access for local and metropolitan networks," *Proc. INFOCOM*, 1992, pp. 2415–2421.

[118] D. Wagner, K. Weihe, "A linear time algorithm for multicommodity flow in planar graphs," *Proc. First European Symposium on Algorithms*, 1993, pp. 384–395.

[119] D.J.A. Welsh, *Matroid Theory*, London: Academic Press, 1976.