Efficient Strategies for Topics in Internet Algorithmics

by

Amitabha Bagchi

A dissertation submitted to The Johns Hopkins University in conformity
with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

October, 2002

Abstract

In this dissertation we posit a coherent area of research called *Internet Algorithmics*. We study a number of different algorithmic problems which are motivated by or are relevant to the various processes which activate the Internet or are activated by it.

The first problem is that of auctioning. We investigate algorithmic strategies that allow sellers to maximize their profits without compromising fairness. We give simple online algorithms to maximize seller profits and give bounds on their competitive ratios, also proving that these competitive ratios are the best achievable by any randomized algorithm.

Subsequently we address the problem of online dictionaries in all its generality. We propose biased skiplists, deterministic and randomized, as an efficient solution to this problem. These structure are easy to implement and can be efficiently updated for storing sorted lists of items. We prove bounds on the update times of these structures which are comparable to the best known times for solutions to the dictionary problem.

Finally we study fault-tolerant routing in interconnection networks. We present a model of fault-tolerance given a bounded number of edge failures and define a set of flow problems which formalize this model. The main problem we study is the *k-Disjoint Flows Problem* in which we are given an undirected network $G = (V, E)$ with edge capacities and a set of terminal pairs T with demands d_i, $1 \leq i \leq |T|$. The problem is to find a subset of the pairs of maximum total demand such that each chosen pair can be connected by k

disjoint paths, each path carrying d_i/k units of flow and no capacity constraint is violated. For this problem, and its variants, we present simple online greedy algorithms. Further, we prove that these algorithms are competitive.

Advisor: Michael T. Goodrich
Reader: Christian Scheideler

Acknowledgements

The first debt of gratitude is due to Mike Goodrich who was willing to demonstrate his faith in me by funding me for several years. More important than some Computer Science stuff, what I have learned from Mike is how an advisor should treat his students.

One person without whom none of this would have been possible is Amitabh Chaudhary who has been my partner in crime, and research, for these six years, always managing to best me by putting off his own work a little more than I'd put off mine.

Tripurari Singh, who once ran after me in slippers to answer a data structures question, has been a pillar of strength; understanding, supportive, giving and affectionate. Sameer Jadhav's patience in the face of my repetitive garrulousness, his fine taste in music, his gentlemanly demeanour, have all made him the best roommate anyone could ask for.

I have benefitted tremendously from Christian Scheideler's commitment to and enjoyment of his work, as well as his incredible perspicacity, for which I am grateful. Adam Buchsbaum's help and support have been invaluable; collaborating with him has been a pleasure. Another very fine person and brilliant researcher I had the good fortune of interacting with is Leslie Hall. Scott Smith's support at a crucial point in my grad school career is something I will not forget.

I feel lucky at having found an area of work which interests me and stimulates me. I am grateful to Lokesh Gupta for dragging me into it and to Kamal Jain and Vijay Vazirani, along with Lokesh, for showing me how exciting this way of thinking could be. But it is to S. Arun-Kumar that I owe the (Dutch?) courage it took for me to jump in.

Through my six years at Hopkins I have made some great friends in the department. Rich Wicentowski was one of the first, always entertaining and unaffected. John Henderson followed. He along with Mihai Pop took it upon themselves to drill some sense into me, and almost succeeded. Lunch with Grace Ngai was always fun. Subsequently came the Center for Algorithms Engineering (CAE) crowd with their immensely entertaining ways and their fantastically creative humour. Thank you Jeremy Mullendore, Michael Shin, Ankur Bhargava, Mark Thober, Joel Sandin and Tahir Butt. And also, thank you.

Every grad student needs friends who can help him hang on to his sanity through the long grind. Anthony Macris' erudition and generosity made my first year at Hopkins a really good time. Panchali Mukherji with her wit and talent for poetry was always a pleasure to talk to. I enjoyed Bhavani Raman's intellect, sense of humour and levelheadedness. Aparna Balachandran, the hostess with the mostest, has been a good friend these last two years. And, of course, through it all, intermittently, there has been Pankaj Kakkar, always entertaining and free with his affection.

At some point these acknowledgments have to be abbreviated and so, although I would love to write paragraphs about all the people whose names follow, I will have to content myself by just mentioning them.

First of all, Deep Sen Gupta, my oldest friend.

Jithesh Parameswaran and Swaminathan Jayaraman who we lost in a tragic road accident.

The Hopkins Desi crowd over the years: Shrikant Jagannathan, Mohammad Noorul Islam, Farhan Hameed, Mona Sharan, Jayasree Iyer, Monish Rajpal, Srinadh Madhavapeddi, Shankar Narayan, Maithilee Mitra, Vaibhava Goel, Veeraraghavan Venkataramani, Vijay Parthasarathi, Karthik Viswanathan, Manuj Pathak, Amit Paliwal, Vikrant Kasarabada and Noshir Pesika.

Other people who made Baltimore a more livable place: Teddy Chao, Cynthia Sarafidis, Rachel Ablow, Kip Lubliner, Jennifer Smith, Pedro Ponce, Tishani Doshi, Sameena Mulla, Roger Begrich, Aaron Goodfellow, Charles Schafer.

The lot at IBM Research in India where I spent two summers: Vijay Kumar, Vinayaka Pandit, Tanveer Afzal Faruquie, Manish Kurhekar, Raghavendra Udupa, Johra Shahabuddin, Pradeep Dubey, Gaurav Aggarwal.

And also those friends who I have not met as often as I wanted to but whose presence in their far off places has been nourishing: Richa Sahai, Trisha Gupta, Vineeta Nayyar, Tarun Bhatia, Ashish Saksena, Sonal Priyanka, Amlan Sen, Kapil Kumar, Tripthi Pillai, Udai Tennati, Vijendra Chauhan.

At the end I have to mention those three people who are most important to me, to whom I dedicate this thesis. Those three people who have supported me in all I have done, never tried to make me be something I wasn't, and never asked for anything in return: My father, my mother and my sister. Thank you for everything.

Amitabha Bagchi
Baltimore,
27th September 2002

Contents

List of Figures

List of Tables

For

Ajoy, Indira and Anu

Chapter 1

Introduction

At the time of writing of this first sentence of this first paragraph of this thesis, the number of active Internet hosts [1] is 185,057,967 according to Telcordia Technologies' Netsizer project [87]. To put this number in context, the United States was home to an estimated 284,796,887 people on July 1, 2001 according to the US Government's Census Bureau [88]. Samuel Morse, had he been alive today, might have exclaimed, "What hath Man wrought!"

The growth of the Internet from a modest sized research project into a major economic, cultural, social and political force that touches the life of a large fraction of the earth's population is a part of contemporary folklore. The extensively distributed nature of the Internet, the anarchic and libertarian ideas that underlie it, the economic forces that seek to mold it, the varied nature of the engineering challenges it poses, all these come together to form a dazzlingly complex mass of issues which need to be addressed by the research community. In this thesis we posit the existence of a diverse but coherent subset of these issues: the problems that can be formulated in algorithmic terms, solutions for which can be developed and analyzed using the methods developed by algorithm design community. We name the study of this subset of issues *Internet Algorithmics*, a term that has been in use for the last couple of years.

[1] defined as computer devices with IP addresses

1.1 Algorithmic Beginnings

In an interview given in 1990, Paul Baran, the central figure in the development of the packet switching technology that is arguably the foundation of the modern Internet, talks about the imperatives driving his research at the RAND corporation under the shadow of the hair-trigger nuclear ballistic missile systems of the Cold War near the end of the 1950s:

> "... long-distance communications networks at that time were extremely vulnerable and not able to survive attack. That was the issue. ... a most dangerous situation was created by the lack of a survivable communication system." (quoted in [1], pp. 10)

In view of this statement consider the following simple definitions taken from Douglas West's text book on Graph Theory [91]:

Definition 1.1.1 *A* graph G *with* n vertices *and* m edges *consists of a* vertex set $V(G) = \{v_1, \ldots, v_n\}$ *and an* edge set $E(G) = \{e_1, \ldots, e_m\}$, *where each edge consists of two vertices called its* endpoints.

Definition 1.1.2 *A graph G is* connected *if it has a u, v-path for each pair $u, v \in V(G)$.*

Definition 1.1.3 *A* vertex cut *of a graph G is a set $S \subseteq V(G)$ such that $G \setminus S$ is not connected. A graph G is* k-connected *if every vertex cut has at least k vertices. The* connectivity *of G is the maximum k such that G is k-connected.*

Baran's problem, formulated in these terms, was two-fold:

- How to design a communication network with high connectivity that spans the United States.
- How to ensure that communications do not fail when the nodes of this network are destroyed.

Both these problems fall under the rubric of Algorithmic Graph Theory, and it is their solution, packet switching, that essentially gave birth to the Internet as we know it today.

This example shows us how, not unlike other fields of technological endeavor, the development of the Internet has been grounded in sound mathematical formulations and made possible by algorithmic realizations of those formulations. The formulation of political concerns in mathematical terms, like the Internet, has outlived the Cold War. Today, more than ever before, there is a need to understand the processes that surround the Internet in sound mathematical terms, and to devise provably efficient and optimal solutions, to the extent possible, of the algorithmic problems which they represent. In this thesis we present three such solutions.

Before we proceed however, it is necessary to understand the nature of the beast.

1.2 Design Principles and Design Features of the Internet

The evolution of the Internet appears to be an anarchic process. Nevertheless, certain design principles have been at the basis of this evolution. Theoretical models for studying the Internet will have to incorporate these principles, or at least the features they seek to provide. The principles enunciated here are taken from the National Research Council's insightful stock taking exercise "The Internet's Coming of Age" published in 2001 [26].

1.2.1 Design Principle 1: Hourglass Architecture

The International Standards Organization's (ISO) Open Systems Interconnection (OSI) Reference Model has been tremendously influential in the development of the Internet. It is pictured in Figure 1.1.[2]

[2]For a clear exposition of this model see, for example, [86].

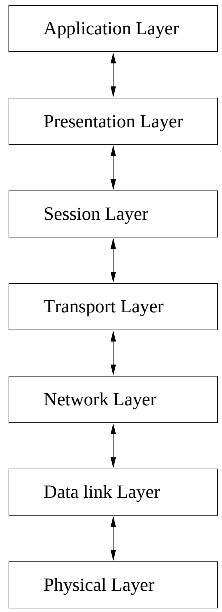


Figure 1.1: The seven layers of the ISO OSI Reference Model.

The key insight of this model is the notion of abstracting away from the underlying transport infrastructure. It allows several different research communities to interact. In a manner of speaking, it allows the Tower of Babel to be built.

The uniform width of the layers depicted in Figure 1.1 hide an important principle of the Internet's Architecture, the so called "Hourglass" principle. Simply stated, this means that at the network layer there should be just one protocol while above and below this level the space of applications, middleware and transport, on one side, and the space of network communication technologies, on the other side, is allowed to grow, pushed forward by research and innovation. This principle is succinctly stated in RFC 1958:

> "It is generally felt that in an ideal situation there should be one, and only one, protocol at the Internet level. This allows for uniform and relatively seamless operations in a competitive, multi-vendor, multi-provider public network. There can of course be multiple protocols to satisfy different requirements at other levels, and there are many successful examples of large private networks

with multiple network layer protocols in use." [44]

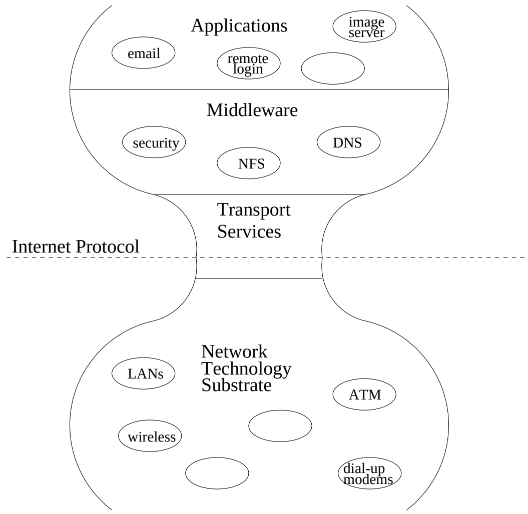


Figure 1.2: The Hourglass Principle. Based on a figure from [26]

Figure 1.2, based on a figure from [26], clarifies this principle further.[3] There are debates about what should lie at the waist of the hourglass. But the notion of keeping this central portion lightweight is an almost uniform consensus.

1.2.2 Design Principle 2: End-to-End Architecture

In any network it is essential to decide what part of the communication function will be performed by the communication subsystem and what part will be done at the communicating ends. The importance of making this differentiation, the "end-to-end argument," was first highlighted by Saltzer and others [76].

The communication layer of the Internet essentially functions as a delivery network, it's only job being transmitting packets from one computer to the

[3] Figure 1.2 also hints at the fact that four layers above the Transport layer are, in effect, actually seen as two layers, Applications and Middleware.

other as efficiently and flexibly as possible. All other functions are done at the edges of the network, in the hosts themselves.

One of the most important consequences of this design feature is that applications do not suffer because of network failures. By keeping the functionality of the communication subnet to a minimum, this design principle also facilitates scalability.

1.2.3 Design Feature 1: Scalability

The design principles listed above have made adding more and more machines to the Internet a relatively easy and decentralized process, quite unlike the expansion of a telephone network. Perhaps an apt analogy would be to a road network, simply making a road from a village to the nearest town or village which has a road, or to the nearest road, successfully adds that village to the entire network.

Despite this there are several pressures which make scaling up the Internet a challenge. We will discuss these in more detail in a subsequent section.

1.2.4 Design Feature 2: Distributed Design and Decentralized Control

To a great extent, individual entities of the Internet have the freedom to interact with the network as they wish. Routing decisions are not controlled centrally. The network itself is hierarchical but relatively democratic in its functioning. Additionally, control of the running and development of the Internet does not reside with any single entity. In its ingenuous but lucid style, RFC 1958 captures this feature the best:

> "Fortunately, nobody owns the Internet, there is no centralized control, and nobody can turn it off." [44]

This decentralization was not a given in the early years of the Internet and has been defended against many attacks (e.g. [1], Chap. 5). However, the Internet as we now know it has probably grown beyond the point where any single entity can control it. Additionally, the processes for designing new protocols, established by the Internet Engineering Task Force, are open to anybody and designed to be vendor-neutral.[4]

As the Internet continues to grow, some of these principles themselves have come under severe pressure from market forces and competing ideologies. A theoretical perspective could be useful in underscoring the validity of these principles or helping debunk them in the face of new ways of thinking about the Internet. Although this thesis does not encapsulate any such research, we feel that the theory of computing community in general, not just the algorithms community, can have a telling impact in this area.

Preliminary to describing the various areas in which the theoretical research could contribute, let us attempt to classify the issues which arise in the area of Internet research.

1.3 Taxonomy of Internet Algorithmics

The study of Internet Algorithmics can be broadly classified under three rubrics:

- **Growth**: Managing and facilitating the increase in the size and quality of the Internet. Size includes number of users, amount of data transferred, transfer speeds etc.

- **Diversification**: Enhancing the quality of the current uses of the Internet and increasing the number of uses of the Internet.

[4] While vendor-neutrality is a desired goal, in practice this is sometimes compromised by the clout certain big players wield.

- **Security and Abuse Prevention**: Making the Internet and its content and processes provably resistant to different kinds of attack. Detecting and helping prevent abuse or misuse.

- **Analysis**: Studying the Internet as a system.

These categories are not exclusive. As we will see later, each class of problems has implications for the other and it is sometimes difficult to decide which of these categories best defines a particular problem. The thrust of this thesis is in the first two areas. In the Sections 1.4 and 1.5 we will endeavor to elaborate on what kind of problems fall under these two rubrics. We will try and investigate one set of issues under each of these heads in some detail and try to point out the sort of theoretical work needed to address them. We will also tie in these issues with the technical contribution of this thesis, described in detail in Chapters 2, 3 and 4.

1.4 Growth

Internet traffic is estimated by some to double every six months through the first decade of the 21st century [70]. Other's feel that the doubling period is closer to one year [65]. The numbers of active hosts is also estimated differently by different sources [87, 45, 57]. Given the complexity and the international spread of the Internet, perhaps the most accurate statement that can be made is the following:

> "The Internet is getting big, and it's happening fast." [9]

No growth is free of pain. The exponential growth of the Internet is no exception to this rule. We do not intend to give a comprehensive survey of all the issues which need to be addressed if the growth of the Internet is to be realized in a smooth manner. We discuss two major problems, the need for more capacity and the need to scale naming systems, pointing out the relevance of this thesis' technical contribution to their solutions.

1.4.1 Growing Pains 1: The Need for More Capacity

We have discussed earlier how the open and layered design of the Internet makes it a naturally scalable system. However, there are a number of pressures which oppose untrammeled growth.

The most obvious concern that accompanies the growing number of hosts and users is the demand for more capacity. Some see the increase in the kinds of devices attached to the Internet as a major driver on demand for capacity. The growing popularity of Digital Subscriber Line (DSL) connections is another source of demand. It is also worth pointing out that the large populations of developing countries are for the most part not connected to the Internet currently. But the social potential of the Internet is clearly recognized in most of these nations and when they surmount the infrastructural issues they are facing (see e.g. [28, 6]), they will join the Internet community in tremendous numbers.

Whatever the source might be, the growth in demand for capacity threatens to outstrip the growth in providers capabilities to provide it. Better use of bandwidth plays a key role in keeping pace with demand. That this engenders a rich problem space for Algorithms researchers needs no repetition. Innovative solutions to this problem have been driven by the Algorithms community, the most notable being the example of Akamai [46].

1.4.2 Easing the Pain 1: Traffic Engineering and MPLS

As the capacity and the bandwidth of the Internet increases, the protocols at work need to scale accordingly. There is a move in the networks community towards "Traffic Engineering" for improving resource management. A definition of this term is given in RFC 2702:

> "In general, [Traffic Engineering] encompasses the application of technology and scientific principles to the measurement, modeling, characterization, and control of Internet traffic, and the application of such knowledge and techniques to achieve specific performance objectives." [10]

Almost hidden among the sequence of adjectives in this definition, the word “control” signifies that the move towards Traffic Engineering is a direct challenge to the end-to-end nature of the Internet discussed in Section 1.2.2. As a result, it throws up a number of algorithmic problems not previously considered in the context of Internet routing.

The Multiprotocol Label Switching (MPLS) standard ([17, 71]) provides a powerful framework for the implementation of Traffic Engineering based schemes to tailor services based on quality of service (QOS) needs. With label switching allowing senders to determine what kinds of routes they want to take through the network, route computation problems have taken on a new importance. However, the robustness to link failure that traditional IP routing encapsulated is lost. In Chapter 4 we address the problem of building fault-tolerance into routing in a circuit-switched environment. The problems addressed in that chapter are just the tip of the iceberg of algorithmic issues brought to light by the move towards label switching networks and Traffic Engineering.

1.4.3 Growing Pains 2: Scaling the Naming Systems

A common naming system is essential for any communications system. In the case of the Internet, despite its decentralized nature, there is a broad consensus on this point (see e.g. [43]). The Domain Name System (DNS, introduced in [60]) performs the role of identifying each host on the Internet by a unique name.

Using the names provided by DNS hides the internal complexity of the Internet from the users. It also allows greater flexibility in changing around IP addresses (which are the unique name used by the network to identify individual hosts), making it possible for organizations to change Internet Service Providers (ISPs) without having to change their advertised “names.”

The DNS is hierarchically designed. At the top are the “root servers” containing the address of the top-level domains such as .com, .edu, .uk and so

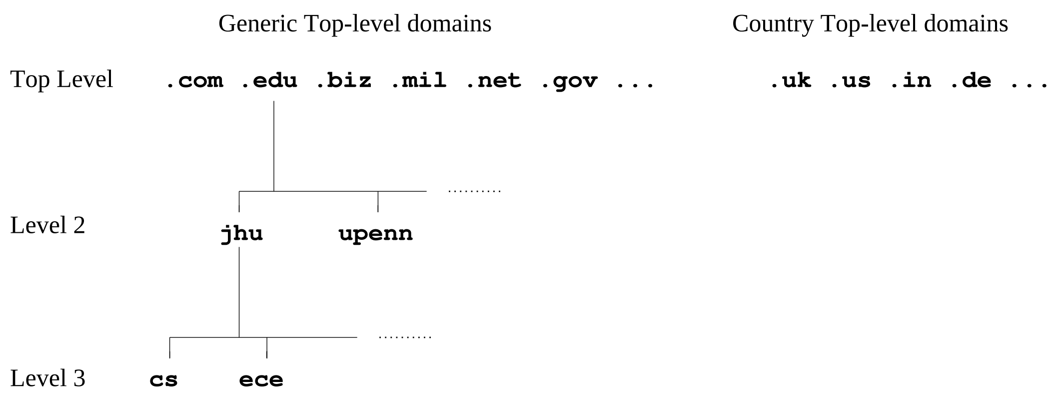


Figure 1.3: The DNS Hierarchy. Based on a figure from [26].

forth (see Figure 1.3). The top-level domain servers in turn record the names of the second level domain name servers, for example, the .edu domain server keeps a record of jhu.edu's DNS server. These second level servers are finally responsible for the mapping from DNS names to IP addresses.

The hierarchical system is in tune with scaling demands but it means that up to three DNS lookups are required for every name resolution operation. As the Internet expands to fill the name space (which can potentially accommodate billions of names), there is a danger that DNS querying will become slower and slower and hence form a bottleneck in communication. Additionally, in practice certain names are requested much more frequently than others and could potentially slow down things for the whole system if they are not handled extremely efficiently.

The way this is dealt with in practice is by using application-side DNS caching, in web browsers for example. The obvious disadvantage being that as the number of active names becomes larger, the application has to manage an increasingly unwieldy cache. Another approach is replication i.e. the distribution of the name database to several servers. The name query load can then be split across the various servers. The problem with this is that it adds another level of indirection to the name resolution process. The limitations of both the methods used brings us back to the fact that efficient querying algorithms are essential if name space expansion is to be managed without slowing the Internet to a crawl.

1.4.4 Easing the Pain 2: Biased Skip Lists for DNS Querying

In Chapter 3 we propose a new structure called Biased Skip Lists as an efficient and easy to implement solution for directory services in general and DNS querying in particular. We give a quick overview of this structure in Section 1.7 but we will leave a more detailed discussion of the advantages of biased skip lists in the specific domain of DNS querying for Chapter 3. We would like to note that the main advantage of skip lists over other search structures is that iterating over the entire data set is very easy, amounting to little more than a list traversal. This advantage was cited [79, 78] for the use of skip lists in the Spread Project [83] for group communications developed by the Center for Networking and Distributed Systems in the CS department at Johns Hopkins.

It is our belief that the Internet will continue to experience a very high rate of growth in the years to come, eventually meeting, and perhaps exceeding, the expectations that fueled the ill-conceived economic euphoria of the late 1990s. But that process of growth will be a difficult and contentious one. The bursting of the Internet bubble has also meant that the mad rush to devise standards and technologies has slowed down to some extent. The time is at hand for taking a deeper theoretical look at some of the relatively ad hoc decisions that were taken in the course of the last few years. It is also time to ensure that theoretically sound models are placed at the centre of future development.

1.5 Diversification

The Internet has come a long way from its beginnings as a network of research institutions sharing scientific data. The innovation of hypertext technology had been made at the European Center for Nuclear Research (CERN) by Tim Berners-Lee in 1980 (see [64], Chap 15) but it was the development of web browsers in the 1990s, the most prominent being Mosaic developed at the University of Illinois by Marc Andreessen, that signaled the beginning of the explosion in the popularity of the Internet, driven mainly by the rise of the World Wide Web.

With the World Wide Web as its public face, the Internet increasingly attracted the attention of various communities that had never used any telecommunication technology more sophisticated than a fax machine. The business community, for instance, was quick to recognize the tremendous potential of the Internet. This recognition was quickly translated into a buzzword: e-commerce.

E-commerce, however, is just a small part of a larger picture of economic activity the Internet has fostered. The most notable economic mechanism to gain success on the Internet was auctioning. In Section 1.5.3 we will discuss the implications of this success. But before we do, let us take a brief look at a few of the other ways in which the Internet is being used, and some of the issues that get raised in the process. Once again, this survey is not meant to be exhaustive. It is essentially meant to illustrate that there is tremendous scope for input by the theory of computing community in helping diversify the Internet's usage.

1.5.1 Case Study 1: Peer to Peer Networking

The term "Peer-to-Peer Networking" seems to be another name for the Internet at first glance. However, the name specifically refers to a class of file-sharing networks that gained popularity, and notoriety, through the music sharing system Napster [63]. The reason for the development of peer-to-peer networks was that file sharing on the Internet was difficult to do and largely restricted to Local Area Networks (LANs) [54]. The limited amount of file sharing that did go on required the users to know each other. Sharing files with new or unknown users was relegated to bulletin boards or Internet Relay Chat, both of which were not particularly easy to use.

Napster changed this scenario by providing a "killer app" for file sharing systems on the Internet. It provided a simple interface that would allow any user connected to the system to share his or her own files and download files from any other user currently logged on. The architecture had an element of centralization, a single server stored a database of the files currently available and their locations, making searches very efficient.

It is easy to see how even in this centralized database model the question of how to manage the load on different peer machines hosting a given file in view of large number of requests for that file is not a trivial one. If we add to this the fact that different machines can provide different upload speeds to the network, then the complexity of the problem increases. Deciding the criteria one needs to consider would give rise to a set of well-stated algorithmic problems.

The legal wrangles between Napster and Recording Industry Association of America have not diminished the popularity of peer-to-peer networking. If anything they have drawn attention to the fact that a centralized design is susceptible to regulation. A more decentralized design is the Gnutella structure [54] in which when a computer joins the network it informs its "neighbours" that it is alive. The neighbours then propagate this information to their neighbours and so on. Searches too propagate through the system in the same way, moving from host to host until the required file is found. Efficiently locating a particular file is an intriguing theoretical problem in this context and some attempts have been made to solve it [85]. In [69], this decentralized model is formulated as a "Content Addressable Network," a distributed infrastructure that provides "hash table-like functionality on Internet-like scales."

The obvious advantages of such a decentralized system is its robustness to failure, a harking back to the fundamentals of the Internet. However, this advantage is just a side effect of the political impulse to create a system that no one can shut down. A logical extension of this notion is to add anonymity into the system. This was proposed by Ian Clarke in [24] (see also [25]). The system is called Freenet. In this system, when an individual user decides to share a file, it is encrypted and then moved over the network to a location that the user does not control. Similarly, the host computer does not know what files reside on it, and hence bears no responsibility for them. The end result is that the file gets shared, but no one really knows where it is or who put it there. Other systems like Publius [89] place primary importance on resistance to censorship in web publishing, providing a "tamper-evident" kind of functionality.

Publius and Freenet provide fascinating examples of political ideology being translated into computer science terms, formulated as system design problems and then solved. While there are concerns about the ethical and political implications of these systems, it is undeniable that the algorithmic concerns that come to light form a rich space for investigation. The continued development of new peer-to-peer systems seems to be a given despite the alarm bells they seem to set off in some quarters. But those alarm bells might well be temporary. As the Urdu poet Ahmed Faraaz said[5]

> The things I say today
> have made me gallows bound
> What a shock! in tomorrow's textbooks
> if they were to be found

1.5.2 Case Study 2: Information Integration

The World Wide Web is home to several databases, most of them reasonably specific in terms of the data they contain. Often a user needs information

[5] Original Urdu version available at [34]. Reference courtesy Raj Kumar Pathria.

that cannot be obtained in its entirety from any one database. A web Information Integration system attempts to answer such queries, which may require extracting and combining data from multiple web sources. There are several important applications that need to deploy such systems, Geographical Information Systems for instance [90]. For a survey on Information Integration systems see [36].

Very little attention is paid in the development of such systems to optimizing querying times and latencies. At most attempts are made to ensure that delay is hidden [8]. In our view this area is ripe for intervention from experts in data structures, scheduling and online algorithms. Sophisticated Information Integration systems that incorporate the varying latencies of different databases, the transmission times of different sized query returns, and other such criteria, will definitely give better performance than some of the systems available today.

The advantages of a decentralized information structure on the Internet echo the advantages of the Internet itself: fault-tolerant, democratic, the property of the community as a whole. As we move forward making better use of this distributed wealth will become more and more important, and it will in turn be a greater load on network resources. It is important that in the relative infancy of the area of Information Integration, adequate attention be paid to making efficient systems that provide quick answers to involved questions while making judicious use of precious network resources.

1.5.3 A Different Case: Internet Auctions

In Sections 1.5.1 and 1.5.2 we discussed two examples of new kinds of systems overlaid on the Internet and emphasized the novelty of the algorithmic problems that designing such systems engendered. The issues essentially deal with different notions of network performance or resource management in new scenarios. However, the popularity of the Internet has also led to a revolution in traditional ways of doing a number of different things, like distributing movies and music for example, giving rise to new ways of thinking about the economics of these and other old[6] areas. The particularly striking case has been that of auctioning.

The democratic potential of the Internet has been a recurring theme of this chapter. The case of auctioning on the Internet reinforces this theme. Folklore has it that Pierre Omidyar, the founder of the first successful auctioning site on the World Wide Web, eBay, was inspired to start his company because his girlfriend (and subsequently, wife) wanted to collect Pez dispensers [21]. While this may or may not be the case, the fact of the matter is that according to a study conducted in 1998, 75% of all auctions on the Internet are for small collectible items with median prices well below $100 and almost no item above $1000 [56].

The prohibitive commissions charged by established auction houses (up to a net of 30% of the price divided between the buyer and the seller at Sotheby's and Christie's) as opposed to the reasonable commissions charged by online auction houses (in the range of 10% for most auction houses, all figures taken from [56]) have meant that it has become affordable to bid for goods and put items up on sale. But, most importantly, it is the ability to enter into an auction, as a buyer or seller, from the comfort of one's own home that has propelled Internet auctioning to the prominence it has today and made household names of companies like eBay, uBid and others. In the public imagination the word auction no longer connotes rich people in dinner jackets and evening gowns paying millions of dollars for a Van Gogh painting as they sip wine. It is more likely to conjure up an image of a baseball cap-wearing beer drinker making a few hundred thousand dollars on eBay for one of Barry Bonds' record breaking 73 home run balls.

Despite the proliferation of auction houses on the Internet, the number of ways of auctioning items has remained quite limited. The most popular kinds of auctions are ascending-bid auctions, which close at a certain time. Given

[6] We use this word to mean "predating the Internet."

the fact that there is no incentive to bid early in such an auction (it reveals information to other bidders), it was found that most people would wait till the end of the auction to enter their bids, making it, in effect, a sealed bid auction. Since then, most auction sites have instituted a practice of having automated bidding agents that each bidder can instruct to keep raising their bid till a certain value. However, this does not encourage bidders to bid their true valuation for an item like the so-called Vickrey auction might.[7]

The point is that there is a need to design sophisticated auctions that keep various considerations in mind. The closing time featured by most auctions is essential in the Internet scenario which is populated by "absentee bidders" who register a bid and then go about their work. They have to be given a time by when they will know whether they won the auction or not. This important requirement changes the rules of bidding for ascending bid auctions that were earlier conducted by a live auctioneer in room where all the bidders were present. This specific example illustrates the more general point that there is a need to reexamine the hoary discipline of auctioning in view of its having made the transition to the Internet.

In conclusion, let us highlight the fact that not just the way people bid or the way they sell has changed, the very nature of what they sell has changed. Consider for example the case of music being sold online in mp3 form: a digital item with potentially unlimited supply. The science of auctioning has to evolve to meet these challenges, and the tools and techniques of computer science can help. How they can help will be the subject of Chapter 2.

In the excitement of the last few years there has been a rush to develop new systems that can benefit from all the Internet has to offer. Much of the research done to this end has been is devising architectures of such systems,

[7]In the Vickrey auction, the highest bidder wins but pays the amount of the second highest bid.

an emphasis on how to do things without enough emphasis on how to do them well. The results of this resource thirsty expansion cannot but be adverse in the long term. On the other hand, the innovation signalled by this process is something worth nurturing and facilitating.

Innovation is also required in designing new processes for the economic interactions that the Internet fosters. Traditionally areas like auctioning have been the preserve of economists. This is the opportunity for the computer science algorithms community to enter the larger world of financial algorithms and make an impact on it.

1.6 Other Issues

It might well be felt that the area of Internet Security should be classified as one of the key areas that will facilitate the diversification of the Internet. And, indeed, without Public Key Cryptography (see e.g. [59]), a whole range of financial transactions could not take place on the Internet. However we feel that the techniques used in this area area are coherent and distinct enough to set it apart from the general run of Algorithmics. We will not undertake a discussion of this vast and very popular area here because the technical contribution of this thesis does not venture into it and there are several others better qualified to write about it than this author.

However, seeing that through the course of this chapter we have highlighted positively the democratic possibilities of the decentralized nature of the Internet, we feel it is only fitting we point out some of the negative consequences of this feature. In Section 1.6.1 we discuss a well-known and simple phenomenon whose solution is notoriously hard to formulate: Denial-of-Service attacks. Another problem with the lack of a single controlling entity is the diversity of essentially selfish objectives that various users hold. In Section 1.6.2 we talk about some of the attempts being made to understand the consequences of selfish individual behaviours on the Internet as a whole.

1.6.1 Combating Misuse: IP Traceback for Denial-of-Service Attacks

One of the problems with the End-to-End design of the Internet (See Section 1.2.2) coupled with the concept of packet switching, is that it is susceptible to various kinds of mischief. To understand what a Denial-of-Service (DoS) attack is, consider this analogy from an article in Linux Journal:

> "Our analogy begins on a college campus with a studious student (SS) who has the misfortune of being placed in a "party" dorm. On a typical evening, SS is studying at his desk trying to master some dry material on data link protocols for his computer networks class. Someone knocks at his door. Upon opening the door, he gets hit with a water balloon from his rowdy neighbors. . . .
>
> He decides on a "secret knock"– his friends announce themselves with a one to five knock code. SS hears the knock and goes to the door; however, he does not open it. Instead, he repeats the original knock and adds his own one to five knock code. Now the visitor knocks the next "sequence" of his code and repeats SS's knocks. . . . we can see that the knock code is able to defeat SS's rambunctious neighbors, but what if they decide to knock once an hour or once every five minutes? What is our studious student to do? The knocks distract him from his homework, but if he ignores the knocks he misses any friends who come by. In other words, frequent knocks deny service to SS's friends." [84]

A Denial-of-Service attack occurs when a large number of irrelevant packets are directed at a host. Informally, the amount of time spent in checking these packets to see if they are useful traffic or garbage means that useful traffic cannot get through. For a description of different kinds of DoS attacks see [23]. For a gripping tale of one man's courage in the face of a Distributed DoS attack (an attack where the malicious packets come from numerous sources) see [37].

The mechanism employed by DoS attacks is so simple and basic that there does not currently seem to be any way to combat it apart from relying on the long arm of the law to bring the perpetrators to book. Even this is not a simple task, given that attackers normally send packets pretending to come from other locations, a technique known as “IP Spoofing.” The first research effort in this area has been to find ways to trace the DoS packets back to their source. This is what the Internet community, known for its facility with nomenclature, refers to as IP Traceback.

For a survey of different techniques used for IP Traceback see [4]. The approaches range from those relying on system level pragmatism [15] to those involving sophisticated mathematics [29]. Probabilistic Packet Marking (PPM) seems to be a promising technique for IP Traceback. Simply stated, it involves routers along the way stamping a packet with their name some of the time i.e. with some low probability. The idea is that when a DoS occurs, a huge number of packets are received and it should be possible to reconstruct the path to the attackers using the markings on these packets. There has been some work on this technique [4, 66] already but the problem of DoS attacks remains.

DoS attacks dramatically highlight the weaknesses of the Internet. Paradoxically, these same weaknesses are also strengths. To eliminate the attacks by trying to change the open and transparent nature of the Internet would be like cutting off the nose to spite the face. DoS is a knotty problem, to which IP Traceback provides just one as yet incomplete solution. Research that helps solve this problem would, in effect, be helping save the freedom of the Internet.

1.6.2 Analysis: Game-Theoretic Views of Internet Behaviour

The Internet can be viewed as an interrelationship of a set of autonomous agents. Each entity needs to interact with others while optimizing a set of goals that might conflict with the goals of others. This is a direct consequence of the decentralized nature of the Internet. The kind of global optimizations possible in a system like the telephone network are simply infeasible on the

Internet. This leads to the question, first posed in algorithmic terms in a slightly different form in [51]: What is the price of freedom? A question that might be posed in slightly harsher terms as: What is the price of selfishness?

Game Theory provides interesting insights into the behaviour of selfish players working to optimize their benefit. The well-known Nash equilibrium and the lesser known Stackelberg equilibrium (in which one player has an altruistic motive, see [73] for details) provide a manageable handle on the exponential behavioural possibilities.

In an interesting sequence of papers, Tim Roughgarden has taken this line of questioning forward (See e.g. [74].) As the authors point out in [51], scenarios where algorithms cope with lack of information (online algorithms) or lack of unbounded resources (approximation algorithms) have been formalized. Analyzing the Internet in these game-theoretic terms gives rise to a field of problems where we have to cope with a lack of coordination.

We have attempted to demonstrate in this chapter that the Internet is at a stage where it requires input from the theory of computing community for its continued wellbeing and growth. It provides an unprecedented opportunity to bridge the long bemoaned gap between theory and practice in Computer Science [7]. The reward for the theory community will not just be funding and gratitude, but, most importantly, a rich set of interesting problems that will push forward our understanding of the fundamental natural phenomenon that is computing.

1.7 Organisation and Overview

We now present an overview of the technical contributions of this thesis, which occupy Chapters 2, 3 and 4.

In Chapter 2 we provide an algorithmic approach to the study of online auctioning. From the perspective of the seller, we formalize the auctioning problem as that of designing an algorithmic strategy that fairly maximizes the revenue earned by selling n identical items to bidders who submit bids online. We give a randomized online algorithm that is $O(\log B)$-competitive against an oblivious adversary, where the bid values vary between 1 and B per item. We show that this algorithm is optimal in the worst-case and that it performs significantly better than any worst-case bounds achievable via deterministic strategies. Additionally we present experimental evidence to show that our algorithm outperforms conventional heuristic methods in practice. And finally we explore ways of modifying the conventional model of online algorithms to improve competitiveness of other types of auctioning scenarios while still maintaining fairness. This work was done jointly with Amitabh Chaudhary and Michael T. Goodrich of Johns Hopkins University and Vijay Kumar and Rahul Garg of IBM's India Research Lab (IRL) in New Delhi. A significant portion of this work was done while the author was visiting the IRL in the summer of 1999.

Pugh [67] introduced skip lists as an alternative to such tree based search structures. Skip lists represent each item by a linked list of nodes; the lists are themselves linked together in a leveled fashion. In Chapter 3 we show how to provide biased access in skip lists. In our deterministic approach a node gets an initial height based on its weight; invariants analogous to those governing (a, b)-trees govern promotion and demotion of node heights to ensure the desired access time. We also describe a randomized variation of this idea, which gives the same, although expected, bounds, and is much simpler to implement. This is joint work with Michael T. Goodrich and Adam Buchsbaum of AT&T Research.

In Chapter 4 we consider the k *edge-disjoint paths problem* (k-EDP), which is a generalization of the well-known edge-disjoint paths problem: given a graph $G = (V, E)$ and a set of terminal pairs (or requests) T, the problem is to find a maximum subset of the pairs in T for which it is possible to select disjoint paths so that each pair is connected by k edge disjoint paths. To the best of our knowledge, nothing nontrivial is known for this problem so far for $k > 1$. In order to measure the performance of our algorithms we will use the recently introduced flow number F of a graph [50]. This parameter is known to fulfill $F = O(\Delta\alpha^{-1}\log n)$, where Δ is the maximum degree and α is the edge expansion of G. We show with the help of a simple, greedy online algorithm that it is possible to achieve a competitive ratio of $O(k^3 \cdot F)$, which naturally extends the best known bound of $O(F)$ for the case that $k = 1$ to higher k. In order to achieve this competitive ratio, we introduce a new method to convert a system of k disjoint paths into a system of k short disjoint paths. We also show that our all deterministic online algorithms have a competitive ratio of $\Omega(k \cdot F)$.

In addition, we study the k *disjoint flows problem* (k-DFP), which is a generalization of the well-known unsplittable flow problem (UFP): the k-DFP is similar to the k-EDP with the difference that now the graph has edge capacities and the requests can have arbitrary demands d_i. The aim is to find a subset of requests of maximum total demand for which it is possible to select flow paths so that all the capacity constraints are kept and each selected request with demand d_i is connected by k disjoint paths of flow value d_i/k. This work was done jointly with Christian Scheideler and Amitabh Chaudhary of Johns Hopkins and Petr Kolman of Charles University, Prague.

Chapter 2

Auctioning

Auctions are among the oldest forms of economic activity known to mankind. Of late there has been a renewed interest in auctioning as the Internet has provided a forum for economic interaction on an unprecedented scale. A number of web sites have been created for supporting various kinds of auctioning mechanisms. For example, at www.priceline.com, users present bids on commodity items without knowledge of prior bids, and the presented bids must be immediately accepted or rejected by the seller. Alternately, web sites such as www.ebay.com and www.ubid.com allow bidding on small lots of non-commodity items, with deadlines and exposure of existing bids. The rules for bidding vary considerably, in fact, even in how equal bids for multiple lots are resolved. Interestingly, it is a simple exercise to construct bidding sequences that result in suboptimal profits for the seller. For example, existing rules at www.ubid.com allow a $100 bid on 10 of 14 items to beat out two $70 bids on 7 items each. Thus, we feel there could be considerable interest in algorithmic strategies that allow sellers to maximize their profits without compromising fairness.

Since Internet auctioning requires a great deal of trust, notions of fairness and legitimacy are fundamental properties for an auction [82]. Indeed, there is considerable previous mathematical treatments of auctioning that classify various types of auctions, together with evolving criteria of fairness (e.g., see [92, 72, 30]). Still, the algorithmic aspects of auctioning have been largely neglected.

2.1 Preliminaries

Given the interactive nature of Internet auctioning today, we feel it most appropriate to study auctioning strategies from an online algorithms perspective. That is, algorithms must make immediate decisions based on existing, incomplete information, and are not allowed to delay responses to wait for future offers. This has the additional advantage of obviating the requirement for a closing time which leads to various inefficiencies (see Section 1.5.3.)

Moreover, given that existing auctioning web sites must implement what are essentially algorithmic rules for accepting or rejecting bids, in this paper we focus on algorithmic strategies for sellers. Even so, we restrict our study to strategies that are honest and fair to buyers. For example, we would consider as unacceptable a strategy that uses a fictitious external bidder that causes a real bidder to offer more than he would had there been less bidding competition. Besides being unethical, dishonest or unfair strategies are ultimately detrimental to any Internet auction house anyway, since their discovery drives away bidders.

2.1.1 Related Work

Offline scenarios for auctioning, where all bids are collected at one time, such as in sealed bid auctions, have been studied and understood in terms of knapsack problems, for which the algorithms community has produced considerable previous work [75, 42, 52]. We are not aware of much previous work on online auctioning strategies, however.

The general area of online algorithms [20] studies combinatorial optimization problems where the problem instance is presented interactively over time but decisions regarding the solution must be made immediately. Even though such algorithms can never know the full problem instance until the end of the sequence of updates, it might not even know when the sequence has ended, online algorithms are typically compared to optimal offline algorithms. We say that an online algorithm is c-competitive with respect to an optimal offline algorithm if the solution determined by the online algorithm differs from that of the offline algorithm by at most a factor of c in all cases.[1] The goal, therefore, in online algorithm design is to design algorithms that are c-competitive for small values of c. Often, as will be the case in this chapter, we can prove worst-case lower bounds on the competitive ratio, c, achievable by an online algorithm. Such proofs typically imply an adversary who constructs input sequences that lead online algorithms to make bad choices. In this paper, we restrict our attention to oblivious adversaries, who can have knowledge of the online algorithm we are using, but cannot have access to any random bits that it may use.

In work that is somewhat related to online auctioning, Awerbuch, Azar and Plotkin [12] study online bandwidth allocation for throughput competitive routing in networks. Their approach can be viewed as a kind of bidding strategy for bandwidth. Leonardi and Marchetti-Spaccamela [53] generalize the result of Awerbuch *et al.*.

Work for online call control [5, 13, 55] is also related to the problems we consider. In online call control, bandwidth demands made by phone calls must be immediately accepted or rejected based on their utility and on existing phone line usage. In fact, our work uses an adaptation of an algorithmic design pattern developed by Awerbuch *et al.* [13] and Lipton [55], which Awerbuch *et al.* call "classify-and-select." In applying this pattern to an online problem, one must find a way to partition the optimization space into q classes such that, for each class, one can construct a c-competitive algorithm (all using the same value of c). Combining all of these individual algorithms gives an online algorithm with a competitive ratio that is $O(cq)$. Ideally, the individual

[1] As a matter of convention we can never have a c-competitive algorithm for $c < 1$ (such algorithms would instead be called $1/c$-competitive)

c-competitive algorithms should be parameterized versions of the same algorithm, and the values c and q should be as small as possible. Indeed, the classify-and-select pattern is best applied to problems that can be shown to require competitive ratios that are $\Omega(cq)$ in the worst case against an oblivious adversary.

2.1.2 Our Results

We consider several algorithmic issues regarding online auctioning, from the viewpoint of the seller, in this paper. We begin, in Section 2.2, by defining the *multiple-item B-bounded online auctioning problem* in which bidders bid on multiple instances of a single item with each bidder allowed to bid on as many items as he or she wants to. We present an online algorithm for this problem that is $\theta(\log B)$-competitive with an oblivious adversary. The upper bound result presented in this sections is based on adaptations of the classify-and-select design pattern [13] to the specific problem of online auctioning.

In Section 2.3 we show that it is not possible for any deterministic algorithm to provide a satisfactory competitive ratio for this problem. Moreover, we show that the algorithm we give in Section 2.2 is "optimal" in the sense that no randomized algorithm can achieve a competitive ratio of $o(\log B)$. To do this we derive lower bounds, based on novel applications of Yao's "randomness shifting" technique [93] , that show the competitive ratios for our algorithm is worst-case optimal.

In order to show that our algorithm performs well in practice we undertook a number of experiments. The results, detailed in Section 2.4, demonstrate that our algorithm handles different types of input sequences with ease and is vastly superior to other online strategies in the difficult case where the bids vary greatly in size and benefit.

Finally, in Section 2.5 we discuss a possible modification of our model with an eye towards making it more flexible as regards its online nature so as to gain in terms of competitiveness. In particular, we allow the algorithm to "buffer" a certain number of bids before making a decision. We show that by buffering only $O(\log B)$ bids it is possible to be c-competitive with an oblivious adversary for the case in which we are selling a single item, for a constant c.

2.2 Multiple-Item B-Bounded Online Auctioning.

In this section we introduce the *multiple-item B-bounded* online auctioning problem. We have n instances of the item on sale and the bids which come in for them offer varying benefit per item. Each bid can request any number of items and offer a given benefit for them. The objective is to maximize the profit that can be earned from the sequence of bids with the additional requirement that the seller accept or reject any given bid before considering any future bids, if they exist.

The *price density* of a bid is defined as the ratio of the benefit offered by the bid to the number of instances of the item that the bid wants to buy. In other words the price density is the average price per item the bidder is willing to pay. The range of possible price densities that can be offered is between 1 and B, inclusively. This restriction is made without loss of generality in any scheme for single-item bidding that has bounded bid magnitude, as we can alternately think of B as the ratio of the highest and lowest bids that can possibly be made on this item. A sequence of bids need not contain the two extreme values, 1 and B, and any bid after the first need not be larger than or equal to the previous bid.

We assume that the algorithm knows the value of B. We discuss at the end of this section how this assumption can be dispensed with.

For this problem we propose an algorithm that uses an adaption of a random choice strategy of Awerbuch and Azar [11] together with the "classify and select" technique of [13], where we break the range of possible optimization values into groups of ranges and select sets of bids which are good in a probabilistic sense based on these ranges. Our algorithm is described in Figure 2.1.

Algorithm Price_And_Pack

- Select i uniformly at random from the integers 0 to $\log B - 1$.
- If i is 0 then set $pd_r = 1$ else set $pd_r = 2^{i-1}$.
- Define a bid as legitimate if it has a price density of at least pd_r
  - Toss a fair coin with two outcomes before any bid comes in.
  - If the coin has landed heads then wait for a legitimate bid on more than $n/2$ items to come in rejecting all smaller bids and all illegitimate bids.
  - Else keep accepting legitimate bids till there is capacity to satisfy them. Reject all illegitimate bids.

Figure 2.1: Auctioning multiple items with bids of varying benefit.

Theorem 2.2.1 Price_And_Pack *is an* $O(\log B)$ *competitive algorithm for the multiple item* B*-bounded online auctioning problem.*

Proof. Let the optimal offline algorithm OPT achieve profit density p on a given input sequence I. So if the optimal algorithm sells $n' \leq n$ items, its total profit is $n'p$. Let j be the largest integer such that $2^j \leq 4p/5$. Define $\alpha = \frac{2^j}{p}$. We say that *Price_And_Pack* chooses i *correctly*, if the chosen value of i equals j. It is easy to see that i is chosen correctly with probability $1/\log B$. In that event, bids of price density greater than $p\alpha$ are legitimate while the rest are not. Note that $\alpha \in (2/5, 4/5]$.

Let I_p be a subset of I, comprising all bids in I which have price density greater than $p\alpha$.

Lemma 2.2.2 *The sum of the revenues obtained by the optimal algorithm running on I_p is no less than $n'p(1-\alpha)$ where p is the profit density of OPT on I and n' is the number of items it sells.*

Proof: Suppose that OPT sells some $n_{lt} \leq n'$ instances to bids in $I - I_p$, and let rev_{ge} be the revenue earned by OPT from items which were sold to bids in I_p. Clearly,

$$rev_{ge} + n_{lt}.p\alpha \geq n'p$$

this gives us

$$rev_{ge} \geq n'p - n_{lt}.p\alpha$$

and since $n_{lt} \leq n'$ we get

$$rev_{ge} \geq n'p(1-\alpha)$$

Since rev_{ge} is the revenue obtained from a subset of the bids in I_p, the result follows. ■

We consider the following three cases, and show that in each case the expected revenue of *Price_And_Pack* is at least $\dfrac{np}{10 \log B}$.

Case 1: There is a bid of size greater than $n/2$ in I_p.

With probability at least $1/\log B$, *Price_And_Pack* chooses i correctly. With probability $1/2$ *Price_And_Pack* chooses to wait for a bid of size greater than size $n/2$. Thus, with probability at least $\frac{1}{2 \log B}$, *Price_And_Pack* will accept a bid of size at least $n/2$ and price density at least αp.

So in this case the expected revenue of *Price_And_Pack* is at least $\dfrac{np\alpha}{4 \log B}$. Since the revenue earned by OPT is np, and $\alpha > 2/5$, in this case *Price_And_Pack* is $10 \log B$ competitive with OPT.

Case 2: There is no bid of size greater than $n/2$ in I_p, and the total number of items demanded by the bids in I_p is more than $n/2$.

With probability 1/2 *Price_And_Pack* will choose to accept bids of any size. If it also chooses i correctly (the probability of which is $1/\log B$), it will sell at least $n/2$ instances [2], and earn a revenue of at least $p\alpha$ units for every item sold.

Thus, with probability $1/2\log B$, *Price_And_Pack* sells at least $n/2$ instances to bids whose price densities are no smaller than $p\alpha$. This means that, in this case, the expected revenue of *Price_And_Pack* is at least $\frac{np\alpha}{4\log B} > \frac{np}{10\log B}$, which makes it $10\log B$ competitive with OPT.

Case 3: There is no bid of size greater than $n/2$ in I_p, and taken together the bids in I_p demand no more than $n/2$ instances.

Again, with probability 1/2 *Price_And_Pack* decides to accept all bids, and with probability $1/\log B$, i is chosen correctly. Thus, with probability $1/2\log B$ our algorithm accepts all bids in I_p, and, by Lemma 2.2.2, earns a revenue no smaller than $n'p(1-\alpha)$ where n' is the number of items sold by OPT. So its expected revenue is at least $\frac{n'p(1-\alpha)}{2\log B} \geq \frac{n'p}{10\log B}$, which makes it $10\log B$ competitive with OPT in this case. ■

An important thing to note is that here the algorithm has to know the range of the input i.e. the algorithm has to be aware of the value of B. It is possible to dispense with this assumption to get a slightly weaker result following [13]. In other words, it is possible to give an $O((\log B)^{1+\epsilon})$ competitive algorithm, for any $\epsilon > 0$, which does not know the value of B beforehand. We do not detail it here because it does not provide any further insight into the problem of auctioning.

In the next section we give lower bounds which will show that *Price_And_Pack* gives the best possible competitive ratio for the this problem.

[2] If *Price_And_Pack* accepts all bids in I_p, it sells at least $n/2$ instances. If it rejects any bid in I_p, it must not have enough capacity left to satisfy it. But then at least $n/2$ instances must have been sold, since any bid in I_p — in particular the rejected bid — is of size no more than $n/2$.

2.3 Lower Bounds for the Online Auctioning Problem.

We consider the version of the online auctioning problem in which there is only one item to be auctioned and the range of possible prices that can be offered for this item is between 1 and B, inclusively. We call this the *single-item B-bounded* online auctioning problem. We give lower bounds for this problem. Upper bounds for this problem are given in [20]. It is easy to see that a lower bound on any algorithm for the single-item problem is a lower bound for the multiple-item problem as well.

In this section we first prove that no deterministic algorithm can in the worst case have a competitive ratio better than the maximum for the single-item problem. More precisely, we show that every deterministic algorithm must have a worst-case competitive ratio that is $\Omega(B)$. This lower bound is based on the fact that a seller does not know in advance how many bids will be offered. Even so, we also show that even if the seller knows in advance the number of bids in the input sequence, any deterministic algorithm is limited to a competitive ratio that is $\Omega(\sqrt{B})$ in the worst case.

Theorem 2.3.1 *Any deterministic algorithm for the single-item B-bounded auctioning problem has a competitive ratio that is $\Omega(B)$ in the worst case.*

Proof: For a given deterministic algorithm A we construct an adversarial input sequence I_A in the following way: Let the first bid in I_A be of benefit 1. If A accepts this bid, then I_A is the sequence $\{1, B\}$. In this case, on the sequence I_A, the deterministic algorithm A gets a benefit of 1 unit while the offline optimal algorithm would pick up the second bid thereby earning a benefit of B units.

If A does not accept this first bid, then I_A is simply the sequence $\{1\}$. In this case A earns 0 units of revenue while the optimal offline algorithm would accept the bid of benefit 1. ∎

Of course, B is the worst competitive ratio that is possible for this problem, so this theorem implies a rather harsh constraint on deterministic algorithms. Admittedly, the above proof used the fact, perhaps unfairly, that the seller does not know in advance the number of bids that will be received. Nevertheless, as we show in the following theorem, even if the number of bids is known in advance, one cannot perform much better.

Theorem 2.3.2 *Any deterministic algorithm for the single-item B-bounded online auctioning problem, where the number of bids is known in advance, has a competitive ratio that is $\Omega(\sqrt{B})$ in the worst case.*

Proof: Consider the input sequence $I_{base} = \{1, 2, 4, \ldots 2^i \ldots B/2, B\}$. For any deterministic algorithm A we construct our adversarial sequence I_A based on what A does with I_{base}.

We recall here that since we are considering the single-item problem, any deterministic algorithm essentially picks at most one of the bids in the input sequence.

Suppose A accepts some bid $2^i \leq \sqrt{B}$. Then we choose I_A to be the same as I_{base}. In this case A's benefit is less than $\sqrt{B}$, whereas an optimal offline algorithm would earn B units thereby making A an $\Omega(\sqrt{B})$ competitive algorithm.

If A accepts some bid $2^i > \sqrt{B}$, on the other hand, then we choose I_A to be $\{1,\ 2,\ 4,\ \ldots 2^{i-1},\ 1,\ 1,\ \ldots\}$, i.e., we stop increasing the sequence just before A accepts and then pad the rest of the sequence with bids of benefit 1. [3] This way A can get no more than 1 unit of benefit while the optimal offline algorithm gets 2^{i-1} which we know is at least $\sqrt{B}$.

If A accepts none of the bids in I_{base} then it is not a competitive algorithm at all (i.e. it earns 0 revenue while the optimal offline algorithm earns B units) and so we need not worry about it at all. ∎

[3] Bids are not always increasing. For a single item there is no issue at all if the bids are always increasing and the number of bids is known. Just wait for the last bid.

It is easy to see that the deterministic algorithm that either picks up a bid of benefit at least $\sqrt{B}$ or, if it does not find such a bid, picks up the last bid, whatever it may be, succeeds in achieving a competitive ratio of $O(\sqrt{B})$.

Theorem 2.3.1 tells us that no deterministic algorithm can effectively compete with an oblivious adversary in the worst case, if the number of bids is not known in advance. Indeed, although the proof used a sequence that consisted of either one bid or two, the proof can easily be extended to any sequence that is either of length n or $n+1$. This bleak outlook for deterministic algorithms is not improved much by knowing the number of bids to expect, however, as shown in Theorem 2.3.2.

Furthermore we show that even randomization does not help us too much. We can use Yao's principle [93] to show that no randomized algorithm can be more competitive against an oblivious adversary than *Price_and_Pack*.

Theorem 2.3.3 *Any randomized algorithm for the single-item B-bounded online auctioning problem is $\Omega(\log B)$-competitive in the worst case.*

Proof. We use Yao's Principle [93, 61] to show a lower bound for all randomized algorithms. To do this we give a probability distribution over the input space and determine the expected benefit of the best possible deterministic algorithm on this probabilistic input. The competitiveness of this expectation against the expected benefit of the optimal offline algorithm for this probabilistically distributed input will be, by Yao's Principle, a lower bound on the competitiveness of any randomized algorithm for this problem.

Consider the following input sequence which we will be calling the base sequence or I_{base}: $\{1, 2, 4, \ldots B/2, B\}$. Our set of input sequences will be derived from I_{base} by truncating it at a given point and substituting bids of revenue 1 for the tail of the input sequence. All other inputs occur with probability 0. The set of inputs, $I = \{I_1, I_2 \ldots I_{\log B}\} \cup \{I_f\}$ and associated probabilities are described as:

- $I_i = \{1, 2, 4, \ldots 2^i, 1 \ldots 1\}$ occurs with probability $P_i = \frac{1}{2^{i+1}}$. Each I_i

has $\log B + 1$ bids.

- $I_f = \{1\}$ occurs with probability $P_f = \frac{B+1}{2B}$.

The expected benefit that an optimal offline algorithm would earn on this probabilistically distributed input is:

$$E[OPT] = P_f.OPT(I_f) + \sum_{i=1}^{\log B} P_i.OPT(I_i)$$

$$= \frac{B+1}{2B}.1 + \sum_{i=1}^{\log B} \frac{1}{2^{i+1}}.2^i$$

$$= \frac{1}{2}\log B + \frac{B+1}{2B} \geq \log B$$

Now let us look at the expected benefit of the best possible deterministic algorithm on this set of inputs. Consider any deterministic algorithm's behaviour on I_{base}. In general it will reject the first j bids and accept the $j+1$st bid for some j. This algorithm will manage to earn a benefit of 2^j on all the input sequences $I_i, i > j$ but will earn at most 1 unit of benefit on all the $I_i, i \leq j$. Of course, it will earn no more than 1 unit on I_f. Therefore the expected benefit of this algorithm, call it A_j, will be:

$$E[A_j] \leq P_f.1 + \sum_{i=1}^{j} P_i.1 + \sum_{i=j+1}^{\log B} P_i.2^j$$

$$\leq \frac{B+1}{2B} + \sum_{i=1}^{j} \frac{1}{2^{i+1}} + 2^j.\sum_{i=j+1}^{\log B} \frac{1}{2^{i+1}}$$

$$\leq \frac{B+1}{2B} + 1 + 1 \leq 3$$

This means that for any deterministic algorithm the expected benefit is less than a constant, namely 3. Putting this together with the fact that $E[OPT]$ is always greater than $\log B$ we get that the competitive ratio of the best deterministic algorithm on our probabilistically distributed input, and hence the competitive ratio of every randomized algorithm on any input, is lower bounded by $\Omega(\log B)$. ■

2.4 Experimental Results.

In order to give an idea of the efficacy of *Price_And_Pack* we present the results of simulated auctions which use this algorithm.

The input sequences were generated by selecting each bid from a given probability distribution. The three distributions used were: Normal, Poisson and Uniform. Both the number of items being bid on and the price density offered by the bid were chosen from the same distribution.

We chose three different combinations of n and B and generated 100 input sequences for each combination. To get a good approximation to the average benefit of *Price_And_Pack* we ran the algorithm 1000 times on each instance and averaged the benefit over all these runs.

We determined a lower bound on the amount of revenue obtained by our algorithm compared to the maximum possible revenue. To do this we implemented an offline algorithm which has been shown to be a 2 approximation [40]. By dividing the revenue obtained by *Price_And_Pack* by 2 times the revenue obtained by the offline algorithm we were able to provide a number which is effectively a lower bound on the actual ratio.

The numbers in Table 2.4 show that in practice *Price_And_Pack* performs quite well compared to the optimal offline algorithm and significantly better than the bound of $O(\log B)$ would suggest. We see that in the two distributions which tend to cluster sample points near the mean, i.e. Normal and Poisson,

(n, B)	Distribution	Expected ratio $(1/\log B)$	Ratio
(50, 1024)	Uniform Normal Poisson	.1	.31 .69 .61
(2000, 1024)	Uniform Normal Poisson	.1	.34 .62 .7
(2000, 2048)	Uniform Normal Poisson	.09	.34 .61 .69

Table 2.1: *Price_And_Pack* v/s the optimal offline algorithm.

the algorithm does especially well. However these distributions provide fairly regular input instances. The real power of *Price_And_Pack* is on view when the input instances have widely varying bids.

To demonstrate this we compared the performance of a simple *Greedy* heuristic with the performance of *Price_And_Pack*. *Greedy* simply accepts bids while it has the capacity to do so. In Table 2.4 we present the results in terms of the percentage extra revenue *Price_And_Pack* is able to earn over the *Greedy* heuristic.

Distribution	(n, B)	Average %
Uniform	(50, 1024) (2000, 1024) (2000, 2048)	25 28.5 27.1
Normal	(50, 1024) (2000, 1024) (2000, 2048)	0.5 0.7 0.5
Poisson	(50, 1024) (2000, 1024) (2000, 2048)	1.4 0.1 0.3

Table 2.2: *Price_And_Pack* over *Greedy*: % advantage.

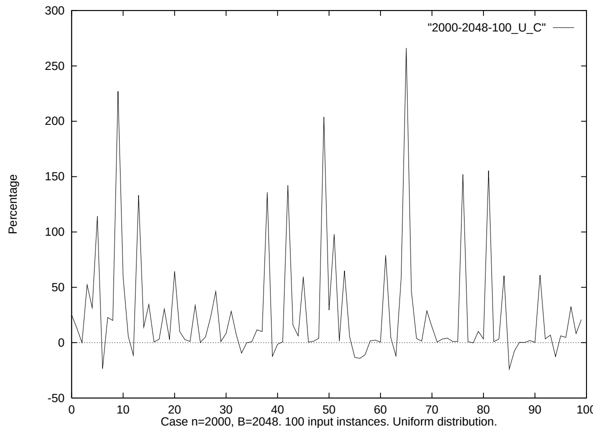


Figure 2.2: *Price_And_Pack* over *Greedy*: % advantage in 100 individual runs.

We see that when the bids are comparable to each other (i.e. when they are generated by the Normal or Poisson distribution) then *Price_And_Pack* does not do significantly better than *Greedy* but when the bids vary widely in size (i.e. when they are generated by the Poisson distribution) then *Price_And_Pack* definitely outperforms *Greedy*.

In Figure 2.4 we graph the percentage extra revenue earned by *Price_And_Pack* in 100 different input instances for a given choice of n and B. It is clear from the graph that *Price_And_Pack* consistently outperforms *Greedy*.

2.5 Improving Competitiveness by Giving Intermediate Information.

In this section we look at a way of modifying the auctioning model to improve competitiveness. Taxonomies of auctions (for eg. [92], [72]) have classified auctions along three broad categories: bid types, clearance mechanisms and intermediate information policies. We look at this lattermost classification to help us improve competitiveness.

In the preceding sections we saw that in the conventional online model, where every bid has to be accepted or rejected before the next bid comes in, we are limited to a competitive ratio of $\Omega(\log B)$. However it is possible to do better if we relax the online model slightly. In the model under consideration so far every bid has to be accepted or rejected immediately, or, more precisely, before the next bid comes in. However in real life auctions this is not always the case. Most auctions do release some intermediate information. For example in the outcry type of auction the current highest bid is announced. This amounts to informing those who bid at that level that their bid is still under consideration, although it might yet be beaten out by a better bid.

The problem with the outcry auction is that, in the case of a monotonically increasing sequence of bids, each bid is asked to hold on and then rejected i.e. $O(B)$ bids are made to wait till a new bid comes in before being rejected. This is clearly unacceptable since from the point of view of a bidder an intermediate notification that the bid is still under consideration is tantamount to saying that this bid has a reasonable chance of success. However if the bidder knows that $O(B)$ bids could be asked to hold then he might not consider this chance of success reasonable.

So, the model we propose is that only a certain small number of bids can be asked to wait without a definite notification of acceptance or rejection. We can think of these bids being buffered in a buffer which will need to contain only one item at a time and will not be allowed to hold more than a certain small number of items in the course of the auction. We call this structure a *k-limited access buffer* or a k-LAB. We denote the highest bid held in the LAB by $H(LAB)$.

That this relaxation is useful becomes immediately evident when we consider that a $\log B$-LAB allows us to become constant competitive deterministically with the optimal algorithm in the case where we want to sell one item and we know the number of bids. We give the algorithm for this in Figure 2.5. We have to view this in light of the fact that in Theorem 2.3.2 we showed that in a purely online setting it is not possible to do better than $\Omega(\sqrt{B})$ deterministically for this problem.

Algorithm LAB_Sell_One_N

- $LAB \leftarrow b_0$
- For each bid b_i for i going from 1 to N do
  - if $b_i > 2.H(LAB)$ then $LAB \leftarrow b_i$ else reject b_i
- Accept the bid in the LAB.

Figure 2.3: *LAB_Sell_One_N:* Auctioning a single item with a buffer when the number of bids is known.

Theorem 2.5.1 LAB_Sell_One_N *is* $\frac{1}{2}$ *competitive with the optimal.*

Proof: It is quite easy to see that the highest bid b_{off} which is the benefit of the offline algorithm would not be put in the buffer only if a bid which was at least half its benefit were already in there. Since a bid of at least half its benefit is already in the online algorithm's buffer therefore its benefit will be at least half of the online's. ■

2.6 Future directions

There are a number of possible directions to extend this work. One interesting project might be the investigation of buyer-focused online strategies for purchasing commodity items from many possible Internet auctioning sites or even the same auction site over time. Formulating strategies for selling goods in infinite supply, like mp3 files for example, is another area where new techniques need to be developed.

Chapter 3

Biased Skip Lists

The need to perform a “dictionary lookup” occurs at several crucial points in the process of end-to-end communication over the Internet. Routing is an important example; each router maintains a table which helps it route packets based on the final destinations. Another example is name resolution using the Domain Name System (DNS). In Section 1.4.3 we talked about DNS and the problems associated with the unfettered growth in the number of names being used. There is a definite need to ensure that DNS lookups are made as efficient as possible. One important aspect of this is to ensure that more frequently looked up names are resolved quicker than less frequently requested ones. For example, if the time taken to resolve www.branchdavidian.com is a little higher than average it might not affect the performance of the network, but if cnn.com is not resolved as fast as possible, it could bring the DNS server to its knees. The popularity of certain hosts on the network normally gets reflected in a skewed access pattern for any given DNS server. A structure which captures this property, and hence helps us optimise DNS performance, is the *Biased Dictionary*.

3.1 Biased Dictionaries

3.1.1 Definition

A *biased dictionary* is a data structure that maintains an ordered set X, each element i of which has a *weight*, w_i; without loss of generality, we assume $w_i \geq 1$. The operations are as follows.

Search(X, i)**.** Determine if i is in X.

Insert(X, i)**.** Add i to X.

Delete(X, i)**.** Delete i from X.

Join(X_L, X_R)**.** Assuming that $i < j$ for each $i \in X_L$ and $j \in X_R$, create a new set $X = X_L \cup X_R$. The operation destroys X_L and X_R.

Split(X, i)**.** Assuming without loss of generality that $i \notin X$, create $X_L = \{j \in X : j < i\}$ and $X_R = \{j \in X : j > i\}$. The operation destroys X.

FingerSearch(X, i, j)**.** Determine if j in in X, beginning the search with a handle in the data structure to element $i \in X$.

Reweight(X, i, w'_i)**.** Change the weight of i to w'_i.

Biased dictionaries can improve on the $\Theta(m \log n)$ time required to perform m accesses on n items in classical, unbiased dictionaries such as AVL trees[3], red-black trees [38], and (a, b)-trees [41]. Let w_i be the number of times item i is accessed. Define $W = \sum_{i=1}^{n} w_i$. A biased dictionary with $O\left(\log \frac{W}{w_i}\right)$ search time for the i'th item can perform m searches on n items in $O(m(1 - \sum_{i=1}^{n} p_i \log p_i))$ time, where $p_i = \frac{w_i}{m}$, which is optimal [2].

3.1.2 A New Structure for the Biased Dictionary Problem

Many biased search-tree data structures have been developed to implement biased dictionaries. All have optimal search time, but the times for the update operations vary slightly among them. Bent, Sleator, and Tarjan [16] and independently Feigenbaum and Tarjan [35] designed biased trees, which offer worst-case and amortized performance, albeit with complicated implementations. Sleator and Tarjan [81] showed how to achieve optimal amortized biased access with a very simple data structure: the splay tree. Seidel and Aragon [80] demonstrate randomized bounds with treaps.

All of the above are binary tree data structures, with updates based on rotations. Pugh [68] introduced skip lists as an alternative to such structures. Skip lists represent each item by a linked list of nodes; the lists are themselves linked together in a leveled fashion. We provide details in Section 3.2. Pugh's original structure was randomized and very simple to implement. A node's linked list was assigned an initial random height with a geometrically decreasing probability distribution; once assigned, the height never changed. Pugh showed that search and updates take $O(\log n)$ expected time in this structure, with no rotations or other rebalancing needed for updates. Munro, Papdakis, and Sedgewick [62] later showed how to determinize Pugh's structure.

We show how to provide biased access in skip lists. Our deterministic approach is simple: a node gets an initial height based on its weight; invariants analogous to those governing (a, b)-trees govern promotion and demotion of node heights to ensure the desired access time. We also describe a randomized variation of this idea, which gives the same, although expected, bounds, and is much simpler to implement. Mehlhorn and Näher [58] anticipated our approach but claimed only a partial result. Ergun et al. [32, 33] designed a skip list that provides access to item i in $O(\log r(i))$ time, where $r(i)$ is the number of accesses since the last time i was accessed.

In Section 3.2, we define skip lists and biased skip lists. Section 3.3 describes how to update biased skip lists, and Section 3.4 demonstrates a simple, randomized variation.

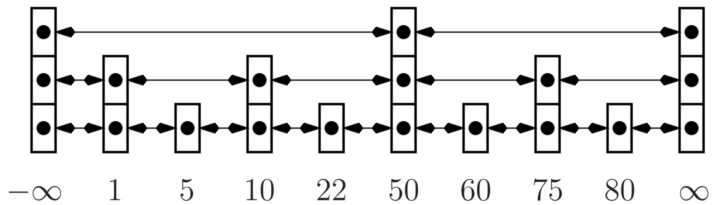


Figure 3.1: A skip list for the set $X = \{1, 5, 10, 22, 50, 60, 75, 80\}$.

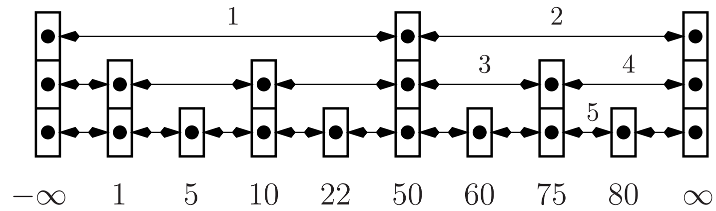
3.2 Biased Skip Lists

3.2.1 Definition

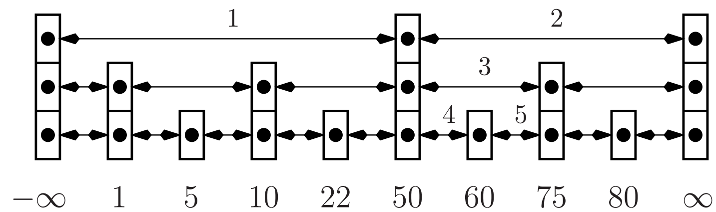
A *skip list* [68] S is a dictionary data structure, storing an ordered set X, the items of which we number 1 through $|X|$. Each item $i \in X$ has a key, x_i, and a corresponding *node* in the skip list of some integral *height*, $h_i \geq 0$. The *height* of S is $H(S) = \max_{i \in X} h_i$. The *depth*, d_i, of item i is $H(S) - h_i$. We use the terms item, node, and key interchangeably where convenient; the context clarifies any ambiguity. We assume without loss of generality that the keys in X are unique: $x_i < x_{i+1}$, $1 \leq i < |X|$.

Each node i is implemented by a linked list or array of length $h_i + 1$, which we refer to as the *tower* for that node. The *level-j successor* of a node i is the least node $\ell > i$ of height $h_\ell \geq j$; i.e., no node $i < k < \ell$ has height $h_k \geq j$. Symmetrically, the *level-j predecessor* of node i is the greatest node $\ell < i$ of height $h_\ell \geq j$. For node i and each $0 \leq j \leq h_i$, the j'th element of the node contains pointers to the j'th elements of the level-j successor and predecessor of i. Two distinct nodes $x < y$ are called *consecutive* if and only if $h_z < \min(h_x, h_y)$ for all $x < z < y$. A *plateau* is a maximal sequence of consecutive nodes of equal height.

For convenience we assume sentinel nodes of height $H(S)$ at the beginning (with key $-\infty$) and end (with key ∞) of S; in practice, this assumption is not necessary. We orient the pointers so that the skip list stores items in left-to-right order, and the node levels progress bottom to top. See Figure 3.1.



(a)



(b)

Figure 3.2: (a) Searching for key 80 in the skip list of Figure 3.1. Numbers over the pointers indicate the order in which they are traversed. (b) Similarly, searching for key 65.

To search for an item with key K, we start at level $H(S)$ of the left sentinel. When searching at level i from some node, we follow the level-i links to the right until we find a key matching K or a pair of nodes j, k such that k is the level-i successor of j and $x_j < K < x_k$. We then traverse one level down and continue the search at level $i - 1$ from node j. The search ends with success if we find a node with key K, or failure if we find nodes j and k as above on level 0. See Figure 3.2.

We describe a deterministic, biased version of skip lists. In addition to a key, x_i, each item $i \in X$ has a *weight*, w_i; without loss of generality, we assume $w_i \geq 1$. We define the *rank* of item i as $r_i = \lfloor \log_a w_i \rfloor$, where a is a constant to be defined shortly.

Definition 3.2.1 *For every a and b such that $1 < a \leq \left\lfloor \frac{b}{2} \right\rfloor$, an (a, b)-*biased skip list *is one in which each item has height $h_i \geq r_i$ and the following invariants hold.*

(I1) *For all $0 \leq i \leq H(S)$, there are never more than b consecutive items of*

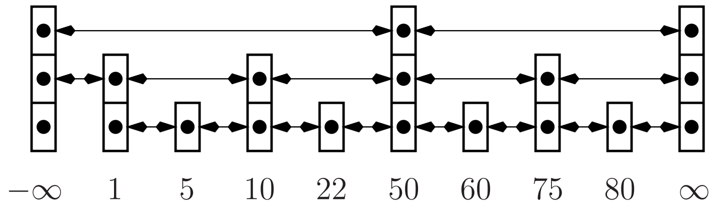


Figure 3.3: The skip list of Figure 3.1 with the unnecessary level-0 pointers between $-\infty$ and 1 set to nil.

height i.

(I2) *For each node x and all $r_x < i \leq h_x$, there are at least a nodes of height $i-1$ between x and any consecutive node of height at least i.*

To derive exact bounds for the case when an item does not exist in the skip list we modify the structure to eliminate redundant pointers. For every pair of adjacent items $i, i+1$, we set the pointers between them on levels 0 through $\min(h_i, h_{i+1}) - 1$ to nil. (See Figure 3.3.)

When searching for an item $i \notin X$, we assert failure immediately upon reaching a nil pointer. It suffices, in fact, to ensure only that the pointers between them on level $\min(h_i, h_{i+1}) - 1$ are nil; the pointers below that level remain undefined.

3.2.2 Analysis

Throughout the remainder of the paper, we define $W = \sum_{i \in X} w_i$ to be the weight of S before any operation. For any key i, we denote by i^- the item in X with largest key less than i, and by i^+ the item in X with smallest key greater than i. The main result of our definition of biased skip lists is summarized by the following lemma, which bounds the depth of any node.

Lemma 3.2.2 (Depth Lemma) *The depth of any node i in an (a,b)-biased skip list is $O\left(\log_a \frac{W}{w_i}\right)$.*

Before we prove the depth lemma, consider its implication on *access time* for key i: the time it takes to find i in S if $i \in X$ or to find the pair i^-, i^+ in S if $i \notin X$.

Corollary 3.2.3 (Access Lemma) *The access time for any key i in an (a, b)-biased skip list is $O\left(1 + \log_a \frac{W}{w_i}\right)$ if $i \in X$ and $O\left(1 + \log_a \frac{W}{\min\left(w_{i^-}, w_{i^+}\right)}\right)$ if $i \notin X$.*

Proof. By **(I1)**, at most $b + 1$ pointers are traversed at any level during a search. Because a search stops upon reaching the first nil pointer, the Depth Lemma thus implies the result for constants a and b. □

To prove the depth lemma, observe that the number of items of any given rank that can appear at higher levels decreases geometrically by level. Define $N_i = |\{x : r_x = i\}|$ and $N'_i = |\{x : r_x \leq i \wedge h_x \geq i\}|$.

Lemma 3.2.4 $N'_i \leq \sum_{j=0}^{i} \frac{1}{a^{i-j}} N_j$.

Proof. We prove the lemma by induction. The base case, $N'_0 = N_0$, is true by definition. For $i > 0$, **(I2)** implies that

$$\begin{aligned} N'_{i+1} &\leq N_{i+1} + \left\lfloor \frac{1}{a} N'_i \right\rfloor \\ &\leq N_{i+1} + \frac{1}{a} N'_i \end{aligned}$$

which, together with the induction hypothesis, proves the lemma. □

Intuitively, this implies that a node promoted to a higher level is supported by enough weight associated with items of equal rank at lower levels. Define $W_i = \sum_{r_x \leq i} w_x$.

Corollary 3.2.5 $W_i \geq a^i N'_i$.

Proof. By definition,

$$W_i \geq \sum_{j=0}^{i} a^j N_j$$
$$= a^i \sum_{j=0}^{i} \frac{1}{a^{i-j}} N_j.$$

Lemma 3.2.4 yields the result. □

Define $R = \max_{x \in X} r_x$. Any nodes with height exceeding R must have been promoted from lower levels to maintain the invariants. **(I2)** thus implies that $H(S) \leq R + \log_a N'_R$, and therefore the maximum possible depth of an item i is $d_i \leq H(S) - r_i \leq R + \log_a N'_R - r_i$.

By Corollary 3.2.5, $W = W_R \geq a^R N'_R$. Therefore $\log_a N'_R \leq \log_a W - R$. Hence, $d_i \leq \log_a W - r_i$. The Depth Lemma follows, because $\log_a w_i - 1 < r_i \leq \log_a w_i$. □

(I2) is stronger than necessary to prove the Depth Lemma. It would suffice for a node of height h exceeding its rank, r, to be supported by at least a items to each side only at level $h-1$, not at every level between the r and $h-1$. The stronger invariant is easier to maintain, however; the update procedures in the next section rely on the support occurring at every level.

3.3 Updating Biased Skiplists

We present and analyze deterministic procedures to update biased skip lists.

First, we define the profile of an item i. For $h_{i^-} \leq j \leq H(S)$, let L^i_j be the level-j predecessor of i; for $h_{i^+} \leq j \leq H(S)$, let R^i_j be the level-j successor of i. Define the ordered sets $PL(i) = \left(j : h_{L^i_j} = j,\ h_{i^-} \leq j \leq H(S)\right)$ and $PR(i) = \left(j : h_{R^i_j} = j,\ h_{i^+} \leq j \leq H(S)\right)$. $PL(i)$ (rsp., $PR(i)$) is the set of distinct heights of the nodes to the left (rsp., right) of i. We call the ordered set $\left(L^i_j : j \in PL(i)\right) \cup \left(R^i_j : j \in PR(i)\right)$ the *profile* of i. We call the subset of predecessors the *left profile* and the subset of successors the *right profile* of i. For example, in Figure 3.1, $PL(60) = (3)$; $PR(60) = (2, 3)$; the left profile of 60 is (50); and the right profile of 60 is $(75, \infty)$.

The profile definitions assume $i \in S$, but they are also precise when $i \notin S$, in which case they apply to the node that would contain key i. Given node i (if $i \in S$) or nodes i^- and i^+ (if $i \notin S$), we can trace i's profile back from lowest-to-highest nodes by starting at i^- (rsp., i^+) and, at any node x, iteratively finding its level-$(h_x + 1)$ predecessor (rsp., successor), until we reach the left (rsp., right) sentinel.

3.3.1 Inserting An Item

The following procedure inserts a new item with key i into an (a, b)-biased skip list S. We assume that i does not already exist in the skip list, or else we discover that in Step 1.

Procedure Insert(S, i)

1. Search S for i to discover the pair i^-, i^+.

2. Create a new node of height r_i to store i, and insert it between i^- and i^+ in S, tracing through i's profile to splice predecessors and successors as in a standard skip list [68].

3. Restore **(I2)**, if necessary. Any node x in the left (sym., right) profile of i might need to have its height demoted, because i might interrupt a sequence of consecutive nodes of height h_x, leaving fewer than a to its left (sym., right). In this case, x is demoted to the next lower height in the profile (or r_x, whichever is higher).

   More precisely, for j in turn from h_{i^-} up through r_i, if $j \in PL(i)$, consider node $u = L^i_j$. If **(I2)** is violated at node u, then demote u to height r_u if $u = i^-$ and otherwise to height $\max(j', r_u)$, where j' is the predecessor

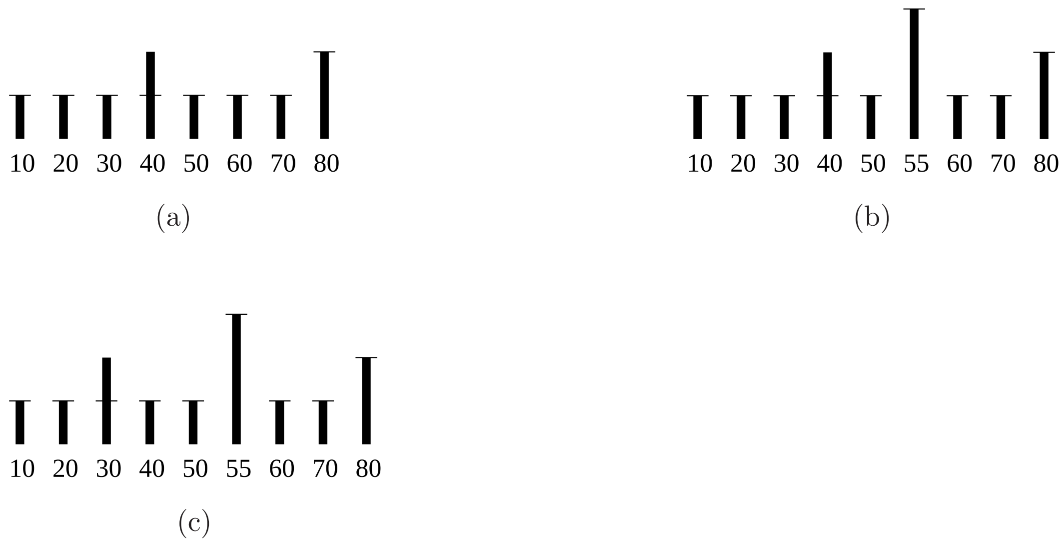


Figure 3.4: (a) A (2,4)-biased skip list. Nodes are drawn to reflect their heights; hatch marks indicate ranks. Pointers between nodes are omitted. (b) After the insertion of 55 with rank 3, node 40 violates **(I2)**. (c) After the demotion of node 40 and compensating promotion of node 30.

of j in $PL(i)$; let h'_u be the new height of u. If the demotion violates **(I1)** at level h'_u, then, among the $k \in (b, 2b]$ consecutive items of height h'_u, promote the $\lfloor \frac{k}{2} \rfloor$'th node (in order) to height $h'_u + 1$. (See Figure 3.4.) Iterate at the next j. Symmetrically process right profile of i.

4. Restore **(I1)**, if necessary. Starting at node i and level $j = r_i$, if node i violates **(I1)** at level j, then, among the $b+1$ consecutive items of height j, promote the $\lfloor \frac{b+1}{2} \rfloor$'th node (in order), u, to height $j+1$, and iterate at node u and level $j+1$. Continue until the violations stop. (See Figure 3.5.)

Lemma 3.3.1 *Invariants **(I1)** and **(I2)** are true after* **Insert**(S, i).

Proof. Assume the invariants were true before the insertion. To the left of node i, **(I2)** can be violated at most once at each level $j \in PL(i) \cap [h_{i^-}, r_i]$, where node i might split a sequence of consecutive nodes of height less than

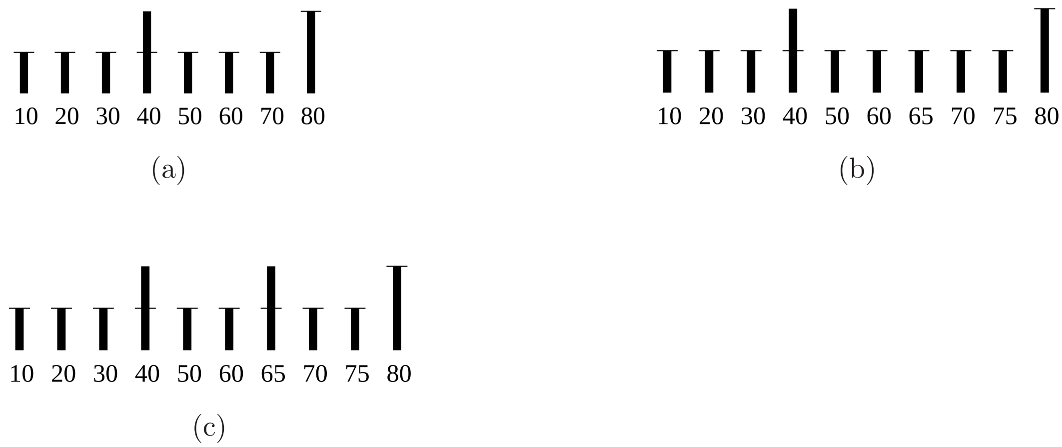


Figure 3.5: (a) The (2,4)-biased skip list of Figure 3.4(a). (b) **(I1)** is violated by the insertion of 65 and 75 with rank 1 each. (c) After promoting node 65.

j. Consider the least j and the associated node $u = L_j^i$ where such a violation occurs. Demoting u in Step 3 restores **(I2)** at level j, by the assumption that **(I2)** was true before the operation. The demotion might cause there to be more than b consecutive nodes of height h'_u around u, however, in which case the promotion of the node in the middle restores **(I1)**. By definition of profile, h'_u is the only height at which this demotion might incur a compensating promotion. Since $a \leq \lfloor \frac{b}{2} \rfloor$, this compensating promotion cannot violate **(I2)**. By induction, iterating Step 3 up to level r_i restores **(I2)** to the left of i. Symmetrically argue for the nodes to the right of i.

Step 4 restores **(I1)** at level r_i if it was violated by the insertion of node i. Promoting the node in the middle cannot violate **(I2)**, because $a \leq \lfloor \frac{b}{2} \rfloor$. The promotion might violate **(I1)** at the next level, however, in which case the iteration of Step 4 restores the invariant. □

Theorem 3.3.2 *Inserting an item i in an (a, b)-biased skip list takes time*

$$O\left(1 + \log_a \frac{W + w_i}{\min(w_{i^-}, w_i, w_{i^+})}\right).$$

Proof. Lemma 3.3.1 proves correctness of **Insert**(S, i).

By the Depth and Access Lemmas, Steps 1 and 2 take

$$O\left(1 + \log_a \frac{W + w_i}{\min(w_{i^-}, w_i, w_{i^+})}\right)$$

steps. For constants a and b, if $\min(h_{i^-}, h_{i^+}) \leq r_i$, Step 3 performs constant work at each level between $min(h_{i^-}, h_{i^+})$ and r_i; Step 4 performs at most constant work at each level from r_i through $H(S)$. Again applying the Depth Lemma yields the result. □

3.3.2 Deleting An Item

Deletion is the inverse of insertion.

Procedure Delete(S, i)

1. Search S to discover i.

2. Find the immediate neighbors i^- and i^+. Remove i, and splice predecessors and successors as required.

3. Restore **(I1)**, if necessary. (Removing i might unite sequences of consecutive nodes into sequences of length exceeding b.) For j in turn from $\min(h_{i^-}, h_{i^+})$ up through $h_i - 1$, if removing i violates **(I1)** at level j, consider the $k \in (b, 2b]$ consecutive nodes of height j, and promote the $\lfloor \frac{k}{2} \rfloor$'th among them to height $j + 1$. Iterate at the next j.

4. Restore **(I2)**, if necessary. (Removing i might decrease the length of the sequence of consecutive nodes of height h_i to $a - 1$, in which case one of the delineating towers might need to be demoted, and so on from there.) Starting at node i and the least $j \in PL(i)$ greater than h_i, if **(I2)** is violated at node $u = L^i_j$, then demote u to height $\max(h_i, r_u)$; let h'_u be the new height of u. For each j' in turn from h'_u up through h_u (the old height of u), if the demotion violates **(I1)** at level j', then, among the $k \in (b, 2b]$ consecutive items of height j', promote the $\lfloor \frac{k}{2} \rfloor$'th among them to height j'. Iterate, checking for an **(I2)** violation at the old height

h_u, continuing until the violations stop. Symmetrically process the right profile of i,

Lemma 3.3.3 *Invariants **(I1)** and **(I2)** are true after* **Delete(**S, i**)**.

Proof. Assume the invariants were true before the deletion. **(I1)** can be violated at most once at each level $\min(h_{i^-}, h_{i^+}) \leq j < h_i$, where the removal of node i might unite two previously separate sequences of consecutive nodes of height j. Step 3 restores **(I1)** at each such level j. Promoting the node in the middle cannot violate **(I2)**, because $a \leq \lfloor \frac{b}{2} \rfloor$, nor can the promotion propagate to higher levels, because the previous existence of node i at level j satisfied **(I1)**.

(I2) can be violated by the removal of i or the subsequent demotion of a predecessor (rsp., successor). By the assumption that **(I2)** held before the operation, any node so violating **(I2)** need be demoted no farther than the height of the preceding node in the left (rsp., right) profile of i (or i in the case of i^- (rsp., i^+)). As in Step 3 of **Insert**, the demotion might require compensating promotions, each of which cannot percolate higher. Step 4 thus restores **(I2)**. $\square$

Theorem 3.3.4 *Deleting an item i from an (a, b)-biased skip list takes time*

$$O\left(1 + \log_a \frac{W}{\min(w_{i^-}, w_i, w_{i^+})}\right).$$

Proof. Lemma 3.3.3 proves correctness of **Delete(**S, i**)**.

By the Depth and Access Lemmas, Steps 1 and 2 take

$$O\left(1 + \log_a \frac{W}{\min(w_{i^-}, w_i, w_{i^+})}\right)$$

steps. For constants a and b, if $\min(h_{i^-}, h_{i^+}) \leq h_i$, Step 3 performs constant work at each level between $min(h_{i^-}, h_{i^+})$ and h_i; Step 4 performs at most constant work at each level from h_i through $H(S)$. Again applying the Depth Lemma yields the result. $\square$

3.3.3 Joining Two Skiplists

Consider two biased skip lists, S_L and S_R, of total weights W_L and W_R respectively. The item with the largest key in S_L is denoted $L_{\max}$, and the item with the smallest key in S_R is denoted $R_{\min}$. If $L_{\max} < R_{\min}$, we can *join* S_L and S_R to form a single biased skip list.

Procedure Join(S_L, S_R)

1. Trace through the profiles of $L_{\max}$ and $R_{\min}$ to splice the pointers leaving S_L together with the pointers going into S_R.

2. Restore **(I1)**, if necessary. For each j in turn from $\max(h_{L_{\max}}, h_{R_{\min}})$ up through $\max(H(S_L), H(S_R))$, if **(I1)** is violated at level j, then among the $k \in (b, 2b+1]$ consecutive items of height j, promote the $\lfloor \frac{k}{2} \rfloor$'th node (in order) to height $j+1$.

Lemma 3.3.5 *Invariants **(I1)** and **(I2)** are true after* **Join(S_L, S_R)**.

Proof. Assuming the invariants were true before the join, splicing the pointers cannot violate **(I2)**, because nodes never lose predecessors or successors.

(I1) can be violated by joining two sequences of nodes at a given level; $\max(h_{L_{\max}}, h_{R_{\min}})$ is the lowest height at which such a violation can occur. Promoting the node in the middle cannot violate **(I2)**, because $a \leq \lfloor \frac{b}{2} \rfloor$. The promotion can add another node to the next higher level, but the splicing procedure left no more than $2b$ nodes there, by the assumption that **(I1)** was true before the join. Thus, no more than $2b+1$ nodes occur at any level prior to a promotion, and so the promotion strategy restores **(I1)**. $\square$

Theorem 3.3.6 *Joining (a, b)-biased skip lists S_L and S_R takes time*

$$O\left(1 + \log_a \frac{W_L}{w_{L_{\max}}} + \log_a \frac{W_R}{w_{R_{\min}}}\right).$$

Proof. Lemma 3.3.5 proves correctness of **Join(S_L, S_R)**. Step 1 performs $O(1)$ work at each level between $\min(h_{L_{\max}}, h_{R_{\min}})$ and $\min(H(S_L), H(S_R))$.

Step 2 potentially performs $O(1)$ work at each level between $\max(h_{L_{\max}}, h_{R_{\min}})$ and $\max(H(S_L), H(S_R))$. Putting this together and applying the Depth Lemma yields the result. $\square$

3.3.4 Splitting A Skiplist

Given a biased skip list S of total weight W and a key $i \notin S$, we can *split* S into two biased skip lists S_L, containing keys in S less than i, and S_R, containing keys in S greater than i. (We can formulate this equivalently when $i \in S$.)

Procedure Split(S, i)

1. Perform **Insert**(S, i), where the weight of i is $a^{H(S)+1}$.
2. Disconnect the pointers between i and its predecessors to form S_L; disconnect the pointers between i and its successors to form S_R.

Theorem 3.3.7 *Splitting an (a, b)-biased skip list on key i takes time*

$$O\left(1 + \log_a \frac{W}{\min(w_{i^-}, w_{i^+})}\right).$$

Proof. Lemma 3.3.1 proves that **(I1)** and **(I2)** are true after Step 1. Because i is taller than all of its predecessors and successors, disconnecting the pointers between them and i in Step 2 does not violate either invariant. Thus, Procedure **Split**(S, i) is correct.

Step 1 takes $O\left(1 + \log_a \frac{W}{\min(w_{i^-}, w_{i^+})}\right)$ time, by Theorem 3.3.2 together with the observation that $w_i = \Theta(W)$, and yields a biased skip list of height $H(S)+1$. Step 2 takes $O(2(H(S)+1) - h_{i^-} - h_{i^+})$ time. Applying the Depth Lemma finishes the proof. $\square$

3.3.5 Finger Searching

We can search for a key j in a biased skip list S starting at any node i (not just the left sentinel) to which we are given an initial pointer (or *finger*).

Assume without loss of generality that $j > i$. The following procedure is symmetric for the case $j < i$.

Procedure FingerSearch(S, i, j)

1. Initialize $u \leftarrow i$, $h \leftarrow r_i$.

2. (Up phase.) If $R_h^u \geq j$, then go to Step 3. Otherwise, if $h < h_u$, set $h \leftarrow h + 1$; else set $u \leftarrow R_h^u$; iterate at Step 2.

3. (Down phase.) Search from u, starting at height h, as in the normal skip-list search procedure outlined in Section 3.2.

The up phase moves up and to the right in the skip list until we detect a node $u < j$ with a level-h successor (for some h) $R_h^u > j$. That the procedure finds j if $j \in S$ or the pair j^-, j^+ if $j \notin S$ follows from the correctness of the vanilla search procedure and that we enter the down phase at the specified node u and height h.

Define $V(i, j) = \sum_{i \leq u \leq j} w_u$.

Lemma 3.3.8 *For any node u and $h \in [r_u, h_u]$, $V(L_h^u, u) \geq a^h$ and $V(u, R_h^u) \geq a^h$.*

Proof. We prove $V(L_h^u, u) \geq a^h$; the other direction is symmetric. If $h = r_u$, then $V(L_h^u, u) \geq w_u \geq a^h$ by definition. Otherwise, $h > r_u$, and by **(I2)**, there are at least a elements of height $h - 1$ between u and L_h^u. By induction, $V(L_h^u, u) \geq aa^{h-1} = a^h$. □

Theorem 3.3.9 *Accessing an item j in an (a, b)-biased skip list, given a pointer to an item i, takes*

$$O\left(1 + \log_a \frac{V(i, j)}{\min(w_i, w_j)}\right)$$

time if $j \in X$ and

$$O\left(1 + \log_a \frac{V(i, j^+)}{\min(w_i, w_{j^-}, w_{j^+})}\right)$$

time if $j \notin X$.

Proof. We can assume constant-time access to level r_i of any node i without affecting previous time bounds. Consider the node u and height h at which we enter the down phase. Intuitively, we show that sufficient weight supports either the link into which u is originally entered during the up phase or the link out of which u is exited during the down phase.

Define $j' = j$ if $j \in X$ and $j' = j^+$ if $j \notin X$. The total search time is 1 plus $O(1)$ per each of $h - \min(r_i, h_{j'}) \leq h - \min(r_i, r_{j'})$ levels. We need only show that $V(i, j') = a^{\Omega(h)}$, which, together with the definition of rank, proves the theorem.

In the case $u > i$, consider the first height $h' \leq h$ at which u is entered during the up phase. If $h' = h$, then $V(i, u) \geq V(L_h^u, u)$, which by Lemma 3.3.8 is at least a^h. Otherwise, $h' < h$, and hence $V(u, j') \geq V(u, R_{h-1}^u)$, which by Lemma 3.3.8 is at least a^{h-1}. Since $V(i, j') \geq V(i, u)$ and $V(i, j') \geq V(u, j')$, either subcase yields $V(i, j') = a^{\Omega(h)}$.

In the remaining case, $u = i$. If $h = r_u$, then $V(i, j') \geq w_i \geq a^h$. If $h > r_u$, then $V(i, j') \geq V(i, R_{h-1}^i)$, which by Lemma 3.3.8 is at least a^{h-1}. Again $V(i, j') = a^{\Omega(h)}$. □

Note that we start the finger search at height r_i, not h_i, which enables the proof to work in case the search starts the down phase immediately.

3.3.6 Changing the Weight of an Item

Finally, we can change the weight of an item i from w_i to w_i' without fully deleting and reinserting i. Denote the new rank of i by r_i'.

Procedure Reweight(S, i, w_i')

1. Search S to find node i.

2. If $r_i' = r_i$, then stop.

3. If $r_i' > r_i$, then do nothing if $h_i \geq r_i'$. Otherwise, promote i to height r_i'; apply Step 3 from Procedure **Insert**(S, i) but starting at height $h_i + 1$;

and apply Step 4 from Procedure **Insert**(S, i) starting at height r'_i.

4. If $r'_i < r_i$, then demote i to height r'_i; apply Step 3 from Procedure **Delete**(S, i) but starting at height r'_i; and apply Step 4 from Procedure **Delete**(S, i) starting at the least $j \in PL(i)$ greater than r'_i.

Lemma 3.3.10 *Invariants **(I1)** and **(I2)** are true after* **Reweight**(S, i, w'_i).

Proof. Assume the invariants were true before the reweight. If $r'_i = r_i$, then neither invariant can be violated. If $r'_i > r_i$, then if $h_i \geq r'_i$, again neither invariant can be violated. Otherwise, if $r'_i > h_i \geq r_i$, i must attain height at least r'_i, in which case **(I2)** can be violated between levels $h_i + 1$ and r'_i, and **(I1)** can be violated at level r'_i. Applying the demotion and promotion steps from the insert procedure fixes the violations as shown in the proof of Lemma 3.3.1.

Finally, if $r'_i < r_i$, **(I2)** can be violated between levels $r'_i + 1$ and r_i. Demoting i to height r'_i and then applying the promotion and deletion procedures from the delete procedure fixes the violations as shown in the proof of Lemma 3.3.3. $\square$

Theorem 3.3.11 *Changing the weight of any node i in an (a, b)-biased skip list from w_i to w'_i takes $O\left(1 + \log_a \frac{W + w'_i}{\min(w_i, w'_i)}\right)$ time.*

Proof. Lemma 3.3.10 proves correctness of **Reweight**(S, i, w'_i). Step 1 takes $O\left(1 + \log_a \frac{W}{w_i}\right)$ time by the Access Lemma. (We assume $i \in X$.) Height promotions and demotions in Steps 3 and 4 perform at most constant work per level and occur no lower than heights $h_i + 1$ and r'_i respectively. Apply the Depth Lemma completes the proof. $\square$

3.4 Randomized Updates

We can randomize the biased skip list structure presented in Section 3.3 to yield expected optimal access times without the need for promotions or demotions. Mehlhorn and Näher [58] suggested the following approach but claimed only that the expected maximal height of a node is $\log W + O(1)$. We will show that the expected depth of a node i is $E[d_i] = O\left(\log \frac{W}{w_i}\right)$.

We parameterize a randomized, biased skip list S by a positive constant $0 < p < 1$. We redefine the *rank* of an item i as $r_i = \lfloor \log_{\frac{1}{p}} w_i \rfloor$. When inserting i into S, we assign its height to be $h_i = r_i + e_i$ with probability $p^{e_i}(1-p)$ for e_i a non-negative integer, which we call the *excess height* of i. Algorithmically, we start node i at height r_i and continually increment the height by one as long as a biased coin flip returns heads (with probability p).

Reweight is the only operation that changes the height of a node. The new height is chosen as for insertion but based on the new weight, and the tower is adjusted appropriately. The remaining operations (delete, join, split, and (finger) search) perform no rebalancing.

Lemma 3.4.1 (Randomized Height Lemma) *The expected height of any item i in a randomized, biased skip list does not exceed* $\log_{\frac{1}{p}} w_i + O(1)$.

Proof.

$$
\begin{aligned}
E[h_i] &= r_i + E[e_i] \\
&= r_i + \sum_{j=0}^{\infty} j p^j (1-p) \\
&= r_i + \frac{p}{1-p} \\
&= \lfloor \log_{\frac{1}{p}} w_i \rfloor + O(1)
\end{aligned}
$$

□

The proof of the Depth Lemma for the randomized structure follows that for the deterministic structure.

Recall the definitions $N_i = |\{x : r_x = i\}|$; $N_i' = |\{x : r_x \leq i \wedge h_x \geq i\}|$; and $W_i = \sum_{r_x \leq i} w_x$.

Lemma 3.4.2 $E[N'_i] = \sum_{j=0}^{i} p^{i-j} N_j$.

Proof. We prove the lemma by induction. The base case, $N'_0 = N_0$, is true by definition. Since the excess heights are i.i.d. random variables, we have, for $i > 0$, $E[N'_{i+1}] = N_{i+1} + pE[N'_i]$, which, together with the induction hypothesis, proves the lemma. $\square$

Corollary 3.4.3 $E[N'_i] \leq p^i W_i$.

Proof. By definition,

$$
\begin{aligned}
W_i &\geq \sum_{j=0}^{i} \frac{1}{p^j} N_j \\
&= \frac{1}{p^i} \sum_{j=0}^{i} p^{i-j} N_j .
\end{aligned}
$$

Lemma 3.4.2 yields the result. $\square$

Lemma 3.4.4 (Randomized Depth Lemma) *The expected depth of any node i in a randomized, biased skip list S is $O\left(\log_{\frac{1}{p}} \frac{W}{w_i}\right)$.*

Proof. The depth of i is $d_i = H(S) - h_i$. As before, define $R = \max_{x \in X} r_x$. By standard skip list analysis [68], we know that

$$
\begin{aligned}
E[H(S)] &= R + O(E[\log_{\frac{1}{p}} N'_R]) \\
&\leq R + cE[\log_{\frac{1}{p}} N'_R] \text{ for some constant } c \\
&\leq R + c \log_{\frac{1}{p}} E[N'_R] \text{ by Jensen's inequality} \\
&\leq R + c\left(\log_{\frac{1}{p}} W_R - R\right) \text{ by Corollary 3.4.3} \\
&= c \log_{\frac{1}{p}} W - (c-1)R
\end{aligned}
$$

By the Randomized Height Lemma, therefore, $E[d_i] \leq c \log_{\frac{1}{p}} W - (c-1)R - \log_{\frac{1}{p}} w_i$. The lemma follows by observing that $R \geq \lfloor \log_{\frac{1}{p}} w_i \rfloor$. $\square$

Corollary 3.4.5 (Randomized Access Lemma) *The expected access time for any key i in a randomized, biased skip list is $O\left(1+\log_{\frac{1}{p}}\frac{W}{w_i}\right)$ if $i \in X$ and $O\left(1+\log_{\frac{1}{p}}\frac{W}{\min(w_{i^-},w_{i^+})}\right)$ if $i \notin X$.*

Proof. As $n \to \infty$, the probability that a plateau starting at any given node is of size k is $p(1-p)^{k-1}$. The expected size of any plateau is thus $1/p$. Applying the Randomized Depth Lemma completes the proof. $\square$

The operations discussed in Section 3.3 become simple to implement.

Insert(S, i). Locate i^- and i^+ and create a new node between them, as described above, to hold i. The expected time is

$$O\left(1+\log_{\frac{1}{p}}\frac{W+w_i}{\min(w_{i^-},w_i,w_{i^+})}\right).$$

Delete(S, i). Locate and remove node i. The expected time is

$$O\left(1+\log_{\frac{1}{p}}\frac{W}{\min(w_{i^-},w_i,w_{i^+})}\right).$$

The Randomized Depth and Access Lemmas continue to hold, because S is as if i had never been inserted in the first place.

Join(S_L, S_R). Trace through the profiles of $L_{\max}$ and $R_{\min}$ to splice the pointers leaving S_L together with the pointers going into S_R. The expected time is

$$O\left(1+\log_{\frac{1}{p}}\frac{W_L}{w_{L_{\max}}}+\log_{\frac{1}{p}}\frac{W_R}{w_{R_{\min}}}\right).$$

Split(S, i). (Assuming $i \notin X$. An equivalent formulation holds when $i \in X$.) Disconnect the pointers that join the left profile of i^- to the right profile of i^+. The expected time is

$$O\left(1+\log_{\frac{1}{p}}\frac{W}{\min(w_{i^-},w_{i^+})}\right).$$

FingerSearch(S, i, j). Perform **FingerSearch(S, i, j)** as described in Section 3.3.5. It is straightforward to prove that Lemma 3.3.8 holds in the expected case. The expected time is thus

$$O\left(1 + \log_{\frac{1}{p}} \frac{V(i,j)}{\min(w_i, w_j)}\right)$$

if $j \in X$ and

$$O\left(1 + \log_{\frac{1}{p}} \frac{V(i,j^+)}{\min(w_i, w_{j^-}, w_{j^+})}\right)$$

if $j \notin X$.

Reweight(S, i, w'_i). Reconstruct the tower for node i as described above. The expected time is

$$O\left(1 + \log_{\frac{1}{p}} \frac{W + w'_i}{\min(w_i, w'_i)}\right).$$

3.5 An Open Problem

An open problem left is the problem of devising a deterministic biased skip list that has not only the worst-case times that we provide but also an amortized bound of $O(\log w_i)$ for *updating* node i; i.e., once the location of the update is discovered, inserting or deleting should take $O(\log w_i)$ amortized time.

The following counterexample demonstrates that our initial method of promotion and demotion does not yield this amortized bound. Consider a node i such that $h_i - r_i$ is large and, moreover, that separates two plateaus of size $b/2$ at each level j between $r_i + 1$ and h_i and two plateaus of size $b/2$ and $b/2 + 1$, rsp., at level r_i. Deleting i will cause a promotion starting at level r_i that will percolate to level h_i. Reinserting i with weight a^{r_i} will restore the structural condition before the deletion of i. This sequence of two operations can be repeated infinitely often; since $h_i - r_i$ is arbitrary, the cost of restoring the invariants cannot be amortized.

Chapter 4

Fault-Tolerant Routing in Circuit Switched Networks

In Section 1.4.2 we mentioned that there is a move towards Traffic Engineering to make communication across the Internet faster. New standards for packet delivery across the Internet emphasize making routing decisions for every end to end connection beforehand, keeping other connections in mind, in order to minimize congestion in the network. Under the IP routing protocol, two packets meant for the same destination would be forwarded along the same link, even if that link is being heavily used. However, with new technology (like the Multi-Protocol Label-switching System (MPLS), see e.g. [17] for details) it is possible to differentiate between two packets, maybe generated by two different applications, headed for the same machine, and send them along different links if the need arises.

In this context standard routing paradigms such as the edge-disjoint paths problem and the unsplittable flow problem [14, 49, 49] are insufficient for practical purposes, since they do not allow a rapid adaptation to edge faults or heavy load conditions. Instead of having just one path for each request, it would be much more desirable to determine a collection of alternative paths for each accepted request that can flexibly be used to ensure rapid adaptability. The paths, however, should be chosen so that not too much bandwidth is wasted under normal conditions. With this end in mind we introduce two new optimization problems: the k edge-disjoint paths problem (k-EDP) and the k disjoint flows problem (k-DFP).

4.1 Preliminaries

4.1.1 Problem definitions

In the k-EDP we are given an undirected graph $G = (V, E)$ and a set of terminal pairs (or requests) T. The problem is to find a maximum subset of the pairs in T such that each chosen pair can be connected by k disjoint paths and, moreover, the paths for different pairs are mutually disjoint.

Similarly, in the k-DFP we are given an undirected network $G = (V, E)$ with edge capacities and a set of terminal pairs T with demands d_i, $1 \leq i \leq |T|$. The problem is to find a subset of the pairs of maximum total demand such that each chosen pair can be connected by k disjoint paths, each path carrying d_i/k units of flow and no capacity constraint is violated.

In order to demonstrate that the k-DFP can be used to achieve fault tolerance together with a high utilization of the network resources and also rapid adaptability, consider a network G in which continuously new edge faults may occur but the total number of faulty edges at the same time is at most f. In this case, given a request with demand d, the strategy is to reserve $k + f$ disjoint flow paths for it, for some $k \geq 1$, with total demand $(1 + f/k)d$. As long as at most f edge faults appear at the same time, it will still be possible to ship a demand of d along the remaining paths. Furthermore, under fault-free conditions, only a fraction f/k of the reserved bandwidth is wasted, which can be made sufficiently small by setting k sufficiently large (which will, of course, be limited by the properties of the network).

4.1.2 Previous results

Since we are not aware of previous results for the k-EDP and the k-DFP for arbitrary k, we will just survey results for the heavily studied case of $k = 1$ i.e. the *edge-disjoint paths problem* (EDP) and the *unsplittable flow problem* (UFP).

Several results are known about the approximation ratio and competitive ratio achievable for the UFP under the assumption that the maximum demand of a commodity, $d_{\max}$, does not exceed the minimum edge capacity, $c_{\min}$, called the *no-bottleneck assumption* in the following [12, 47, 14, 22, 39, 49, 50]. If nothing other than the number m of edges is known, Baveja and Srinivasan [14] present a polynomial time algorithm with approximation ratio $O(\sqrt{m})$. On the lower bound side, it was shown by Guruswami et al. [39] that on directed networks the UFP is NP-hard to approximate within a factor of $m^{1/2-\epsilon}$ for any $\epsilon > 0$. The best result for the EDP and UFP was given by Kolman and Scheideler [50]. Using a new parameter called the *flow number* F of a network, they show that a simple online algorithm has a competitive ratio of $O(F)$ and prove that $F = O(\Delta\alpha^{-1} \log n)$, where for the EDP Δ is the maximal degree of the network, α is the expansion and n is the number of nodes (for the UFP the capacities are taken into account). Combining the approach of Kolman and Scheideler [50] with the AAP algorithm [12], Chakrabarti et al. [22] recently proved an approximation ratio of $O(\Delta^2\alpha^{-1} \log^2 n)$ for the more general UFP with profits.

We also consider two related problems, the *integral splittable flow problem* (ISF) [39] and the *k-splittable flow problem* (k-SFP). In both cases, the input and the objective (i.e., to maximize the sum of accepted demands) is the same as in the UFP. The difference is that in the ISF all demands are integral and a flow satisfying a demand can be split into several paths, each carrying an integral amount of flow, and, in the k-SFP a demand may be split into up to k flow paths (not necessarily integral). Under the no-bottleneck assumption Guruswami et al. [39] give an $O(\sqrt{m d_{\max}} \log^2 m)$ approximation for the ISF.

Recent results of Kolman and Scheideler [50] imply an $O(F)$ randomized competitive ratio and an $O(F)$ deterministic approximation ratio for both of these problems on uniform capacity networks. Although the ISF and the k-SFP on one side and the k-DFP on the other seem very similar at first sight there is a serious difference between the two. Whereas the ISF and the k-SFP are relaxations of the UFP (they allows the use of more paths for a single request and the paths are *not* required to be disjoint), the k-DFP is actually a more complex version of the UFP since it requires several *disjoint* paths for a single request.

4.1.3 New results

The main results of this chapter are

- a deterministic online algorithm for the k-EDP with competitive ratio $O(k^3 F)$.
- a deterministic offline algorithm for the k-DFP on unit-capacity networks with an approximation ratio of $O(k^3 F \log(kF))$.

Using known techniques, we also show how the online algorithm for the k-EDP can be transformed into an offline algorithm with approximation ratio $O(k^3 F)$ for the k-EDP with profits, and we describe how the offline algorithm for the k-DFP can be converted into a randomized online algorithm for the k-DFP with an expected competitive ratio of $O(k^3 F \log(kF))$.

In addition, we show that any deterministic online algorithm for the k-EDP (and consequently also for the k-DFP) has a competitive ratio of $\Omega(k \cdot F)$. Thus, for constant k, we have matching upper and lower bounds for the k-EDP. Furthermore, we demonstrate that the disjointness condition of the k paths for every single request seems to be the crucial condition that makes the problems above harder than the other related problems.

Our algorithms for the k-EDP and k-DFP are based on a simple concept, a natural extension of the *bounded greedy algorithm* (BGA) that has already been studied in several papers [47, 49, 50]: for every request for which there are still k disjoint flow paths of total length at most L available without violating the capacity constraints, select any such system of k paths for it. The core of the chapter is in the analysis of this simple algorithm. The problem is to show that this strategy works even if the optimal offline algorithm connects many requests via k disjoint paths of total length more than L. In order to solve this problem, we use a new technique, based on the Menger's theorem and the Lovász Local Lemma, that converts large systems of k disjoint paths into small systems of k disjoint paths. Previously, shortening strategies were only known for $k = 1$ [49, 50].

4.1.4 Basic notation

Many of the previous techniques for the EDP and related problems are hard put to prove strong upper bounds on approximation or competitive ratios due to the use of inappropriate parameters. If m is the only parameter used an upper bound of $O(\sqrt{m})$ is essentially the best possible for the case of directed networks [39]. Much better ratios can be shown if the expansion or the routing number [77] of a network are used. These measures give very good bounds for low-degree networks with uniform edge capacities, but are usually very poor when applied to networks of high degree or highly nonuniform degree or edge capacities. To get more precise bounds for the approximation and competitive ratios of algorithms, Kolman and Scheideler [50] recently introduced a new network measure, the *flow number* F. Not only does the flow number lead to more precise results, it also has the major advantage that, in contrast to the expansion or the routing number, it can be computed exactly in polynomial time. Hence we will use the flow number here as well.

Before we can introduce the flow number, we need some notation. In a *concurrent multicommodity flow problem* there are k commodities, each with two terminal nodes s_i and t_i and a demand d_i. A *feasible* solution is a set of flow paths for the commodities that obey capacity constraints but need not meet the specified demands. The important thing is that, in contrast to the unsplittable flow problem, the commodity between s_i and t_i can be routed along multiple paths. The *(relative) flow value of a feasible solution* is the maximum f such that at least $f \cdot d_i$ units of commodity i are simultaneously routed for each i. The *max-flow* for a concurrent multicommodity flow problem is defined as the maximum flow value over all feasible solutions. For a path p in a solution, the *flow value of* p is the amount of flow routed along it. A special class of concurrent multicommodity flow problems is the *product multicommodity flow problem* (PMFP). In a PMFP, a nonnegative weight $\pi(u)$ is associated with each node $u \in V$. There is a commodity for every pair of nodes and the demand for the pair (u, v) is equal to $\pi(u) \cdot \pi(v)$.

Suppose we have a network $G = (V, E)$ with arbitrary non-negative edge capacities. For every node v, let $c(v) = \sum_{w:\{v,w\}\in E} c(v, w)$ be the capacity of v and $\Gamma = \sum_v c(v)$ be the capacity of G. Given a concurrent multicommodity flow problem with feasible solution $\mathcal{S}$, let the *dilation* $D(\mathcal{S})$ of $\mathcal{S}$ be defined as the length of the longest flow path in $\mathcal{S}$ and the *congestion* $C(\mathcal{S})$ of $\mathcal{S}$ be defined as the inverse of its flow value (i.e., the congestion tells us how many times the edge capacities would have to be increased in order to satisfy all the demands along the paths of $\mathcal{S}$). Let I_0 be the PMFP in which $\pi(v) = c(v)/\sqrt{\Gamma}$ for every node v, that is, each pair of nodes (v, w) has a commodity with demand $c(v) \cdot c(w)/\Gamma$. The *flow number* $F(G)$ of a network G is the minimum of $\max\{C(\mathcal{S}), D(\mathcal{S})\}$ over all feasible solutions $\mathcal{S}$ of I_0. When there is no risk of confusion, we will simply write F instead of $F(G)$. Note that the flow number of a network is invariant to scaling of capacities.

The smaller the flow number, the better are the communication properties of the network. For example, $F(\text{line}) = \Theta(n)$, $F(\text{mesh}) = \Theta(\sqrt{n})$, $F(\text{hypercube}) = \Theta(\log n)$ and $F(\text{expander}) = \Theta(\log n)$.

Another useful class of concurrent multicommodity flow problems is the *balanced multicommodity flow problem* (BMFP). A BMFP is a multicommodity flow problem in which the sum of the demands of the commodities originating and the commodities terminating in a node v is at most $c(v)$ for every $v \in V$.

We will make use of the following property of the problem [50]:

Lemma 4.1.1 *For any network G with flow number F and any instance I of the BMFP for G, there is a feasible solution for I with congestion and dilation at most $2F$.*

Apart from the flow number we will also need Chernoff bounds and the general Lovász Local Lemma.

Lemma 4.1.2 (Chernoff-Hoeffding) *Consider any set of n independent random variables $X_1, \ldots, X_n$ that take values in the range $[0, k]$. Let $X = \sum_{i=1}^{n} X_i$ and μ be chosen so that $\mu \geq \mathrm{E}[X]$. Then it holds for all $\delta \geq 0$ that*

$$\Pr[X \geq (1+\delta)\mu] \leq \mathrm{e}^{-\min[\delta^2,\, \delta]\cdot\mu/(3k)} .$$

Lemma 4.1.3 (Lovász Local Lemma [31]) *Let $A_1, \ldots, A_n$ be "bad" events in an arbitrary probability space. Suppose that $G = (V, E)$ is a dependency graph of these events and suppose there are real numbers $x_i \in [0, 1)$, $1 \leq i \leq n$, with*

$$\Pr[A_i] \leq x_i \prod_{(i,j)\in E} (1 - x_j)$$

for all $1 \leq i \leq n$. Then,

$$\Pr[\bigcap_{i=1}^{n} \bar{A}_i] \geq \prod_{i=1}^{n} (1 - x_i) .$$

In particular, with positive probability no bad event A_i appears.

4.2 Algorithms for the k-EDP

Consider the folowing extension of the bounded greedy algorithm: Let L be a suitably chosen parameter. Given a request, reject it if it is not possible to find k edge-disjoint paths, $p_1, p_2, \ldots p_k$ between the terminal nodes of the pair such that $\sum_{i=1}^{k} |p_i| \leq L$, where $|p|$ is the length (i.e., the number of edges) of a path p. By disjoint we mean mutually disjoint and, moreover, disjoint with all other paths established so far. Let us call this algorithm k-BGA.

Note that the problem of finding k edge-disjoint paths $p_1, p_2, \ldots p_k$ of total length at most L can be reduced to the classical min-cost (integral) flow problem, which can be solved by standard methods in polynomial time [27, Chapter 4]. It is worth mentioning that if there were a bound L/k on length of every path the problem would not be tractable any more (cf. [18]).

4.2.1 The upper bound

Theorem 4.2.1 *Given a network G of flow number F, the competitive ratio of the k-BGA with parameter $L = 32k^3F$ is $O(k^3F)$.*

Proof. Since for $k = 1$ the result is known, we assume $k \geq 2$. In the following, we call the k edge-disjoint paths that were selected for a request a *k-system*. A k-system is called *small* if it has at most L edges.

Let $\mathcal{B}$ be the solution obtained by the k-BGA and $\mathcal{O}$ be the optimal solution. In the analysis of the algorithm we say that a k-system $q \in \mathcal{B}$ is a *witness* for a k-system p if p and q share an edge. Obviously, a request with a small k-system in the optimal solution that was rejected by the k-BGA must have a witness in $\mathcal{B}$.

Let $\mathcal{O}' \subseteq \mathcal{O}$ denote the set of all k-systems in $\mathcal{O}$ that are larger than L and that correspond to requests *not* accepted by the k-BGA and that do *not* have a witness in $\mathcal{B}$. Then each k-system in $\mathcal{O} - \mathcal{O}'$ either has a witness or was accepted by the k-BGA. Since the k-systems in $\mathcal{O} - \mathcal{O}'$ are edge-disjoint, each request accepted by the k-BGA can be a witness to at most L requests in $\mathcal{O} - \mathcal{O}'$. Hence, $|\mathcal{O} - \mathcal{O}'| \leq (1 + L)|\mathcal{B}|$.

It remains to prove an upper bound on $|\mathcal{O}'|$. To achieve this, we transform the k-systems in $\mathcal{O}'$ into a set $\mathcal{P}$ of possibly overlapping but *small* k-systems. Since these small k-systems are candidates for the k-BGA but were not picked, each of them has at least one witness. Then we show that the small k-systems in $\mathcal{P}$ do not overlap much and thus many k-systems from $\mathcal{B}$ are needed in order to provide a witness from $\mathcal{B}$ for every k-system in $\mathcal{P}$.

Lemma 4.2.2 *The k-systems in $\mathcal{O}'$ can be transformed into a set $\mathcal{R}$ of flow systems transporting the same amount of flow such that every flow path has a length of at most $8k \cdot F$. Furthermore, the congestion at every edge used by a k-system in $\mathcal{O}'$ is at most $1 + 1/(2k)$, and the congestion at every other edge is at most $1/(2k)$.*

Proof. Consider a pair (s_i, t_i) corresponding to a k-system in $\mathcal{O}'$, and let the k paths chosen for it be denoted by $p_1, p_2, \ldots, p_k$. At least one of these k paths must have a length at least $L/k = 32k^2F$, since otherwise the k-system would be in $\mathcal{O} - \mathcal{O}'$. A path with more than $8kF \leq 32k^2F$ edges is called *long*. Suppose that the number of long paths connecting the vertices s_i and t_i is n_i. Let us say that these long paths are $p_1, p_2, \ldots p_{n_i}$ and the others, the small paths, if any, are $p_{n_i+1}, p_{n_i+2}, \ldots, p_k$.

For each long path p_r, $1 \leq r \leq n_i$, we construct $4kF$ pairs of vertices in the following way. Let the first $4kF$ nodes of p_r be denoted by $a^{p_r}_{i,1} = s_i, a^{p_r}_{i,2} \ldots a^{p_r}_{i,4kF}$ and the last $4kF$ nodes be $b^{p_r}_{i,1}, b^{p_r}_{i,2} \ldots b^{p_r}_{i,4kF} = t_i$. Note that for each long path these two sets of nodes do not overlap. Denote by $\mathcal{L}$ the multiset

$$\mathcal{L} = \bigcup_{i:(s_i,t_i)\in\mathcal{O}'} \bigcup_{r=1}^{n_i} \bigcup_{j=1}^{4kF} \{(a^{p_r}_{i,j}, b^{p_r}_{i,j})\} \quad .$$

If we think about each pair $(a, b) \in \mathcal{L}$ as a request with demand one, then, since the k-systems in $\mathcal{O}'$ are edge-disjoint, $\mathcal{L}$ is (a subset of) an instance of a balanced multicommodity flow problem. From the properties of the flow number (Lemma 4.1.1) we know that there exists a (fractional) flow system $\mathcal{F}$ for $\mathcal{L}$ with congestion at most $2F$ and dilation at most $2F$.

For every $(a^{p_r}_{i,j}, b^{p_r}_{i,j}) \in \mathcal{L}$ let $f^r_{i,j}$ denote the corresponding (possibly fractional) flow subsystem as $\mathcal{F}$, and let $g^{p_r}_{i,j}$ denote the flow system between s_i and t_i of value $1/(4kF)$ that first moves from s_i to $a^{p_r}_{i,j}$ along the path p_r, then from $a_{i,j}^{p_r}$ to $b_{i,j}^{p_r}$ along the flow system $f_{i,j}^r$, and finally, from $b_{i,j}^{p_r}$ to t_i along the path p_r again, each with a flow value $1/(4kF)$. If this is done for every pair $(a, b) \in \mathcal{L}$, and, moreover, if all the short flow paths $\bigcup_{i:(s_i,t_i)\in\mathcal{O}'} \bigcup_{r=n_i+1}^{k} p_r$ are added, then in the resulting flow system, denoted by $\mathcal{R}$, k units of flow are transferred between all pairs in $\mathcal{O}'$. Thus, we maintain in $\mathcal{R}$ the original flow for every $(s_i, t_i) \in \mathcal{O}'$. Furthermore, all flow paths are small (i.e., consist of at most $4kF + 2F \leq 8kF$ edges), and the additional edge congestion caused by the flows is at most $(2F)/(4kF) = 1/(2k)$. Since the congestion of the k-systems in $\mathcal{O}'$ is at most 1, the lemma follows. □

Lemma 4.2.2 does not immediately provide us with short k-systems for the requests in $\mathcal{O}'$, since the flow systems in $\mathcal{R}$ may not consist of disjoint paths. Hence, we still have to show how to extract short k-systems out of these.

Lemma 4.2.3 *For every request in $\mathcal{O}'$, a set of small k-systems can be extracted out of its flow system in $\mathcal{R}$ with a total flow value of at least* $1/4$.

Proof. Let (s_i, t_i) be a fixed request from $\mathcal{O}'$ and let E_i be the set of all edges that are traversed by the flow system for (s_i, t_i) in $\mathcal{R}$. Consider any set of $k-1$ edges in E_i. Since the edge congestion caused by $\mathcal{R}$ is at most $1 + 1/(2k)$, the total amount of flow in the flow system for (s_i, t_i) in $\mathcal{R}$ that traverses E_i is at most $(k-1)(1+1/(2k)) \leq k - 1/2$. Thus, the minimal $s_i - t_i$-cut in the graph (V, E_i) consists of at least k edges. Hence, Menger's theorem [19] implies that there are k edge-disjoint paths between s_i and t_i in E_i. We take any such k paths and denote them as the k-system σ_1. We associate a weight of $k \cdot \epsilon_1$ with σ_1, where ϵ_1 is the minimum flow from s_i to t_i through an edge in E_i belonging to the k-system σ_1.

Assume now that we have already found ℓ k-systems $\sigma_1, \cdots, \sigma_\ell$, for some $\ell \geq 1$. If $\sum_{j=1}^{\ell} k \cdot \epsilon_j \geq \frac{1}{2}$ we stop the process of defining σ_j. Otherwise, the minimal $s_i - t_i$-cut in (V, E_i) must still be at least k, because the total flow along any $k-1$ edges in E_i is still less than the total remaining flow from s_i to t_i. Thus, we can apply Menger's theorem again. This allows us to find another k-system $\sigma_{\ell+1}$ between s_i and t_i and in the same way as above we associate with it a weight ϵ_{l+1}. Let $\hat{\ell}$ be the number of k-systems at the end of the process.

So far there is no guarantee that any of the k-systems defined above will be small, neither that they will transport enough flow between the terminal pair s_i and t_i. However, after a simple procedure they will satisfy our needs.

According to Lemma 4.2.2, all flow paths in $\mathcal{R}$ have a length of at most $8kF$. Hence, the total amount of edge capacity consumed by a flow system in $\mathcal{R}$ representing a request in $\mathcal{O}'$ is at most $8k^2F$. If there were k-systems in $\sigma_1, \cdots, \sigma_{\hat{\ell}}$ of total flow value at least $1/4$ that use more than $32k^3F$ edges each, then they would not fit into the available edge capacity, because $32k^3F \cdot 1/(4k) \geq 8k^2F$. Thus, there exists a subset of the k-systems $\sigma_1, \cdots, \sigma_{\hat{\ell}}$ with total flow value at least $1/4$ such that each of them is small, that is, each of them uses at most $32k^3F$ edges. □

Let $\mathcal{S}_i$ denote the set of small k-systems for request (s_i, t_i) given by this lemma, i.e. Lemma 4.2.3, and let $\mathcal{S}$ be the set of all $\mathcal{S}_i$. A random experiment will finally help us to bound $|\mathcal{O}'|$ in terms of $|\mathcal{B}|$. Independently for each $(s_i, t_i) \in \mathcal{O}'$, choose exactly one of its k-systems in $\mathcal{S}$, where a k-system σ_j is picked with a probability proportional to its flow value. After the selection, each of the chosen k-systems is used to carry k units of flow, one unit along each of its paths. Let $\mathcal{P}$ denote the chosen k-systems with the k units of flow.

For every $\sigma_j \in \mathcal{S}_i$, let the binary random variable $X_i^{\sigma_j}$ be one if and only if σ_j was chosen in the random experiment. Since each k-system in $\mathcal{P}$ is small, and therefore a candidate for the k-BGA, but was rejected by the k-BGA, it must have a witness in $\mathcal{B}$. This witnessing must be at an edge that is not used by any k-system in $\mathcal{O}'$, because otherwise the definition of $\mathcal{O}'$ would be violated. Hence, only edges outside of the edges used by $\mathcal{O}'$ can be potential witness edges. From Lemma 4.2.2 we know that each of these edges can have a congestion of at most $1/(2k)$. Hence, after selecting a random, small k-system for each request and shipping a demand of 1 along each of its paths, the expected congestion at every potential witness edge is at most 2. Thus, in expectation, every k-system from $\mathcal{B}$ can serve as a witness to at most $2 \cdot L$ k-systems from $\mathcal{P}$. We conclude that there exists such a random choice for which the k-systems from $\mathcal{B}$ serve as witnesses to at most $2 \cdot L \cdot |\mathcal{B}|$ k-systems from $\mathcal{P}$ (cf. [49]). Since $|\mathcal{P}| = |\mathcal{O}'|$, the proof is completed. $\square$

The upper bound on the competitive ratio for the k-BGA in Theorem 4.2.1 is the best possible, since a k-system of size $\Theta(k^3 F)$ may prevent $\Theta(k^3 F)$ other k-systems from being selected. An open question is whether it is possible to achieve a better competitive ratio with a stronger restriction on the size of the k-systems that are used by the k-BGA.

4.2.2 General lower bound

Next we show there is a lower bound that holds for the competitive ratio of any deterministic online algorithm for the k-EDP problem which is not far away from the performance of the k-BGA.

Theorem 4.2.4 *For any n, k, and $F \geq \log_k n$ with $n \geq k^2 \cdot F$ there is a graph G of size $\Theta(n)$ with maximum degree $O(k)$ and flow number F so that the competitive ratio of any deterministic online algorithm on G is $\Omega(k \cdot F)$.*

Proof. Consider the graph in Figure 4.1. The core of this graph consists of $m = n/(k \cdot F) \geq k$ line graphs and a so-called k-ary multibutterfly network. In addition, a node s is connected to the first k line graphs on the left, and a node t is connected to the first k line graphs on the right. For every i, line graph i consists of a sequence of $F + \log_k n$ diamond structures, and each diamond structure represents a smallest possible k-system between two nodes: structure j has two nodes $s_{i,j}(= t_{i,j-1})$ and $t_{i,j}(= s_{i,j+1})$ that are connected via k intermediate nodes. The endpoints of the diamond structures in the shaded area represent the nodes of the underlying multibutterfly. We will use the fact that a k-ary multibutterfly with n' inputs and outputs (which is a network of degree $O(k)$) can route any r-relation from the inputs to the outputs with congestion and dilation at most $O(\max[r/k, \log_k n'])$ [77].

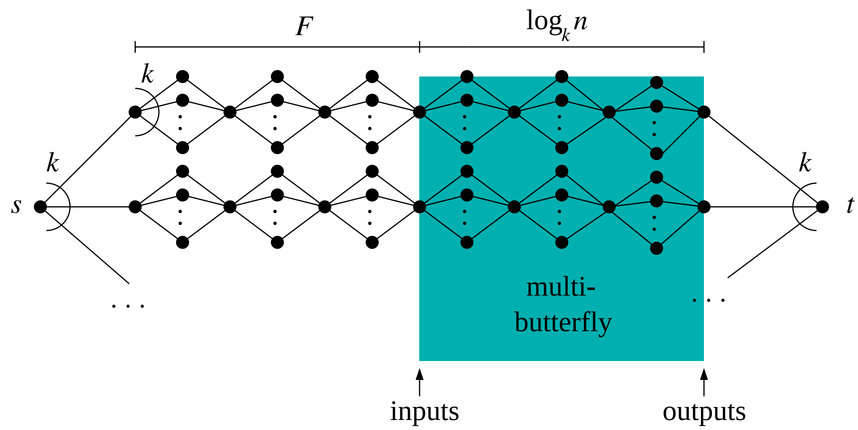


Figure 4.1: The graph for the lower bound.

First, we show that this graph has a flow number of $\Theta(F)$. For this we have to prove that the PMFP for the given graph can be solved with congestion and dilation $O(F)$. Consider each node v of degree δ_v to consist of δ_v copies of nodes and let V' be the set of all of these copies, $V' = \{0, \ldots, N-1\}$. Then the PMFP reduces to the problem of sending a packet of size $1/N$ for any pair of nodes in V'. Such a routing problem can be split into N permutations π_i with $\pi_i(v) = (v + i) \bmod N$ for all $i \in \{0, \ldots, N-1\}$ and $v \in V'$. Each such permutation represents a routing problem ρ in the original network where each node is the starting point and endpoint of a number of packets that is equal to its degree. We want to bound the congestion and dilation for routing such a problem.

In order to route ρ, we first move all packets to the inputs of the k-ary multibutterfly by using the line graphs. This can clearly be done with edge congestion $O(F)$ and dilation $O(F)$. Next, we use the multibutterfly to send the packets to the rows of their destinations. Since every input has $O(k \cdot F)$ packets, this can also be done with congestion and dilation $O(F)$. Finally, all packets are sent to their correct destinations. This also causes a congestion and dilation of at most $O(F)$. Hence, routing ρ only requires a total congestion and dilation of $O(F)$.

Combining the fact that all packets are of size $1/N$ with the fact that we have N permutations π_i, it follows that the congestion and dilation of routing the PMFP in the given graph is $O(F)$. Hence, its flow number is $O(F)$.

Now consider the following two sequences of requests:

(1) (s, t)

(2) $(s, t), (s_{1,1}, t_{1,1}), (s_{1,2}, t_{1,2}), \ldots, (s_{1,F}, t_{1,F}), (s_{2,1}, t_{2,1}), \ldots, (s_{k,F}, t_{k,F})$

Obviously, every deterministic online algorithm has to accept (s, t) to ensure a finite competitive ratio for (1). However, in this case none of the other requests in (2) can be satisfied. But the optimal solution for (2) is to reject (s, t) and to accept all other requests. Hence, the competitive ratio is $\Omega(k \cdot F)$. □

4.2.3 Managing requests with profits

We define the *k edge-disjoint paths with profits problem (k-EDPP)* as follows: we are given an undirected graph $G = (V, E)$ and a set of terminal pairs (or requests) T. Each request $r_i = (s_i, t_i)$ offers a profit (or benefit) $b(r_i)$ if it is accepted. The problem is to find a subset S of the pairs in T for which it is possible to select disjoint paths so that each pair is connected by k disjoint paths such that the sum of the profits of the requests in S is maximum.

It turns out that a simple offline version of the k-BGA gives the same approximation for the k-EDPP as we have for the k-EDP, that is an $O(k^3 \cdot F)$ approximation. The algorithm involves sorting the requests in decreasing order of their profits and running the BGA on this sorted sequence. We call this algorithm the *sorted* k-BGA.

Theorem 4.2.5 *Given a network G of flow number F, the competitive ratio of the sorted k-BGA with parameter $L = 32k^3F$ is $O(k^3F)$ for the k-EDPP.*

Proof. The proof is almost identical to the proof of Theorem 4.2.1. We will just point out the differences here.

We carry forward all the notation from the proof of Theorem 4.2.1. For any set of requests $\mathcal{S}$, denote the sum of the profits of the requests in $\mathcal{S}$ by $P(\mathcal{S})$. We retain the definition of a witness from the proof of Theorem 4.2.1 and we add the notion of *profit witnessed*. If an edge e witnesses requests whose profits total up to some l, we say that the profit witnessed by e, denoted $pw(e)$, is l. The profit witnessed by a path $p \in \mathcal{B}$, denoted $pw(p)$ is the sum of profits witnessed by the edges along that path.

As before, let $\mathcal{O}' \subseteq \mathcal{O}$ denote the set of all k-systems in $\mathcal{O}$ that are larger than L and that correspond to requests *not* accepted by the k-BGA and that do *not* have a witness in $\mathcal{B}$.

We claim the following:

Claim 4.2.6 *There exists a rearrangement of the k-systems of $\mathcal{O}'$ into a set $\mathcal{S}$ such that*

1. *Every request in $\mathcal{O} \setminus \mathcal{O}' \cup \mathcal{S}$ is either accepted by the sorted k-BGA or has a witness in $\mathcal{B}$.*

2. *For every edge e in a path $p \in \mathcal{B}$*

$$pw(e) \leq 3 \cdot b(p)$$

Proof. We know from the proof of Theorem 4.2.1, that each edge in $\mathcal{B}$ witnesses at most one request in $\mathcal{O} \setminus \mathcal{O}'$. And also, from the second case, that there is a rearrangement of $\mathcal{O}'$ into a set $\mathcal{S}$ such that each k-system in $\mathcal{S}$ is small, i.e. it has a witness in $\mathcal{B}$, and the congestion among the k-systems is no more than 2.

The claim follows from the simple observation that if a request p in $\mathcal{B}$ witnesses a request q then, since we sorted the requests by decreasing profits, $b(q) < b(p)$. □

We denote E' to be the set of witness edges. From Claim 4.2.6 it follows that:

$$\begin{aligned} P(\mathcal{O}) &= P(\mathcal{O} \setminus \mathcal{O}') + P(\mathcal{S}) \\ &\stackrel{(1)}{\leq} P(\mathcal{B}) + \sum_{e \in E'} pw(e) \\ &\stackrel{(2)}{\leq} P(\mathcal{B}) + \sum_{p \in \mathcal{B}} 3 \cdot b(p) \\ &\leq (1 + 3 \cdot L) \cdot P(\mathcal{B}) \end{aligned}$$

Where (1) follows from the first part of Claim 4.2.6 and (2) follows from the second part. □

4.2.4 The Multiway EDP

A variant of the k-EDP to which our techniques can be applied is the *Multiway Edge Disjoint Paths Problem* (MWDP) which can be defined as follows: given a graph G and a set of terminal pairs with integral demands d_i, find a maximum subset of the pairs for which it is possible to select disjoint paths so that every selected pair i has d_i disjoint paths.

The k-BGA used for the k-EDP can be used here as well, with k set to $d_{\max}$, where $d_{\max}$ us the maximum demand of all the requests.

Theorem 4.2.7 *Given a network G of flow number F, the competitive ratio of the BGA with parameter $L = 32d_{\max}^3 F$ is $O(d_{\max}^3 F)$, where $d_{\max}$ is the maximum demand of an accepted request.*

The theorem can be shown along the same lines as Theorem 4.2.1.

4.3 Algorithms for the k-DFP

Throughout this section we will assume that the maximal demand is at most k times larger than the minimal edge capacity, which is analogous to assumptions made in almost all papers about the UFP. We call this the *weak bottleneck assumption*. Moreover, we assume that all edge capacities are the same, equal to one. The minimal demand of a request will be denoted by $d_{\min}$. We first show how to solve the offline k-DFP, and then mention how to extend this solution to the online case.

To solve the offline k-DFP, we first sort the requests in decreasing order of their demands. On this sorted sequence of requests we use an algorithm that is very similar to the k-BGA: let L be a suitably chosen parameter. Given a request with a demand of d, reject it if it is not possible to find k edge-disjoint paths $p_1, p_2, \ldots p_k$ of flow value d/k between the terminal nodes of the request that fit into the network without violating the capacity constraints and whose total length $\sum_{i=1}^{k} |p_i|$ is at most L. This extension of the k-BGA will be called *k-flow BGA*.

The next theorem demonstrates that the performance of the k-flow BGA for the k-DFP is comparable to the performance of the k-BGA for the k-EDP. It is slightly worse due to technical reasons; it is much harder to use our technique for extracting short k-systems for the k-DFP than for the k-EDP.

Theorem 4.3.1 *Given a unit-capacity network G with flow number F, there is a constant γ such that the approximation ratio of the k-flow BGA for the k-DFP with parameter $L = \gamma \cdot k^3 F \log(kF))$, when run on requests sorted in non-increasing order, is $O(k^3 F \log(kF))$.*

Proof. As usual, let $\mathcal{B}$ denote the set of k-systems for the requests accepted by the BGA and $\mathcal{O}$ be the set of k-systems in the optimal solution. Each k-system consists of k disjoint flow paths which we also call *streams*.

For each stream $q \in \mathcal{B}$ or $q \in \mathcal{O}$, let $f(q)$ denote the flow along that stream. If q belongs to the request (s_i, t_i) with demand d_i, then $f(q) = d_i/k$. For a set $\mathcal{Q}$ of streams let $||\mathcal{Q}|| = \sum_{q \in \mathcal{Q}} f(q)$. Also, for an edge $e \in E$ and a stream q, let $F(e, q)$ denote the sum of the flow values of all streams in $\mathcal{B}$ passing through e whose flow is at least as large as the flow along q, i.e., $F(e, q) = ||\{p \mid p \in \mathcal{B},\ e \in p,\ f(p) \geq f(q)\}||$. A stream $p \in \mathcal{B}$ is a *witness* for a stream q if $f(p) \geq f(q)$ and p and q intersect in an edge e with $F(e,q) + f(q) > 1$. For each edge e let $\mathcal{W}(e, \mathcal{B})$ denote the set of streams in $\mathcal{B}$ that serve as witnesses on e. Similarly, for each edge e let $\mathcal{V}(e, \mathcal{Q})$ denote the set of streams in $\mathcal{Q}$ that have witnesses on e. We also say that a k-system has a witness on an edge e if any of its k streams has a witness on e. We start with a simple observation.

Lemma 4.3.2 *For any stream q and edge e, if q has a witness on e then $\|\mathcal{W}(e, \mathcal{B})\| \geq 1/2$.*

Proof. Let p be a witness of q on e. Assume, by contradiction, that $F(e,q) < 1/2$. It easily follows that $f(p) < 1/2$. Since $f(q) \leq f(p)$ and $F(e,q)+f(q) > 1$ by the definition of a witness, we have a contradiction. □

Let $\mathcal{O}' \subset \mathcal{O}$ be the set of k-systems that are larger than L and that correspond to requests *not* accepted by the BGA and that do *not* have a witness in $\mathcal{B}$. The next two bounds on $||\mathcal{O} \setminus \mathcal{O}'||$ and $||\mathcal{O}'||$ complete the proof.

Lemma 4.3.3 $||\mathcal{O} \setminus \mathcal{O}'|| \leq (1 + 2L) \cdot ||\mathcal{B}||$.

Proof. We partition $\mathcal{O} \setminus \mathcal{O}'$ into two sets. Let $\mathcal{O}_1 \subseteq \mathcal{O} \setminus \mathcal{O}'$ consist of all the k-systems corresponding to requests accepted by the BGA and let $\mathcal{O}_2 = (\mathcal{O} \setminus \mathcal{O}') \setminus \mathcal{O}_1$. Note that each k-system in $\mathcal{O}_2$ must have a witness in $\mathcal{B}$. Let $E' \subseteq E$ denote the set of all edges on which some k-system from $\mathcal{O}_2$ has a witness. We then have

$$\|\mathcal{O}_2\| \leq \sum_{e \in E'} k\|\mathcal{V}(e, \mathcal{O}_2)\| \leq \sum_{e \in E'} k \leq \sum_{e \in E'} k \cdot 2\|\mathcal{W}(e, \mathcal{B})\|$$

For the first inequality note that a k-system of demand d_i in $\mathcal{O}_2$ may only have a witness at a single edge, and this edge can only be traversed by a flow of d_i/k belonging to that k-system.

Since all k-systems in $\mathcal{B}$ are of length at most L, we have

$$\sum_{e \in E'} \|\mathcal{W}(e, \mathcal{B})\| \leq \sum_{\text{streams } p \in \mathcal{B}} |p| \cdot f(p) \leq \sum_{k-\text{systems } s \in \mathcal{B}} L \cdot d(s)/k \leq L \cdot \|\mathcal{B}\|/k \; .$$

This completes the proof. □

In the next lemma we bound $\|\mathcal{O}'\|$ by first transforming the large k-systems in $\mathcal{O}'$ into a set $\mathcal{S}$ of small k-systems and then bounding $\|\mathcal{S}\|$ in terms of $\|\mathcal{B}\|$.

Lemma 4.3.4 *$\|\mathcal{O}'\| = O(L \cdot \|\mathcal{B}\|)$.*

Proof. In order to prove the lemma, we will transform the k-systems in $\mathcal{O}'$ into a set of k-systems $\mathcal{S}$ in which each k-system has a length of at most L and therefore must have a witness in $\mathcal{B}$. To achieve this, we perform a sequence of transformations:

1. First, we scale the demands and edge capacities so that each edge in G has a capacity of $C = \lceil 3k/d_{min} \rceil$ and all requests have demands that are integral multiples of k. More precisely, the demand of each request of original demand d is set to $d' = k \cdot \lceil C \cdot d/k \rceil$. Since $d'/C \in [d, (1 + 1/3)d]$, this slightly increases the demands and therefore also the flows along the streams so that the total flow along an edge is now at most $(1 + 1/3)C$. Note that slightly increasing the demands only increases $\|\mathcal{O}'\|$ and therefore only makes the bound on the relationship between $\|\mathcal{O}'\|$ and $\|\mathcal{B}\|$ more pesimistic.

2. Next, we replace each request (s_i, t_i) in $\mathcal{O}'$ by d'_i/k *elementary* requests of demand k each, shipped along the same k-system as for (s_i, t_i). For every k-system of such a request, we only keep the first $8c \cdot kF$ and the last $8c \cdot kF$ nodes along each of its k streams for some $c = O(\log(kF))$. The remaining pieces will be called a *k-core*. As shown in Claim 4.3.5, we can then distribute the elementary requests among C/c sets $S_1, \ldots, S_{C/c}$ so that the congestion caused by the k-cores within each set is at most $2c$ at each edge and at most $2\delta c$ at each node of degree δ.

3. Afterwards, we consider each S_i separately. We will reconnect disconnected streams in each k-core in S_i with flow systems given by the flow

number. Then, we will show in Claim 4.3.6 how to extract k-systems of length at most L from each reconnected k-core.

4. Once we have found the short k-systems, we will be able to compare $\|\mathcal{O}'\|$ with $\|\mathcal{B}\|$ with the help of witnesses.

We next present two vital claims.

Claim 4.3.5 *The elementary requests can be distributed into C/c sets named $S_1, \ldots S_{C/c}$ for some $c = O(\log(kF))$ so that for each set S_i the congestion caused by its k-cores is at most $2c$ at each edge and at most $2\delta c$ at each node of degree δ.*

Proof. We first prove the claim for $c = O(\log(kF + C))$ and afterwards mention how to improve it to $c = O(\log(kF))$.

Our proof uses the Chernoff-Hoeffding bounds and the general Lovász Local Lemma (LLL). Consider the random experiment of assigning to each elementary request a number in $\{1, \ldots, C/c\}$ uniformly and independently at random, and let S_i be the set of all requests that choose number i. For every node v and edge e let the random variable $X_{v,i}$ be the number of streams assigned to S_i that traverse v and the random variable $Y_{e,i}$ be the number of streams assigned to S_i that traverse e. Obviously,

$$\mathrm{E}[X_{v,i}] \leq \delta_v \cdot 4c/3 \quad \text{and} \quad \mathrm{E}[Y_{e,i}] \leq 4c/3$$

for all v and e, where δ_v denotes the degree of v. To be able to apply the Chernoff-Hoeffding bounds, we have to specify how much a k-core can contribute to a random variable $X_{v,i}$ and $Y_{e,i}$. For every node v of degree δ_v, any k-core can have at most $\min[\delta_v, k]$ steams crossing it. Hence, a k-core can contribute a value of at most $\min[\delta_v, k]$ to $X_{v,i}$. On the other hand, every edge e can be used by at most one stream of any k-core. Hence, a k-core can contribute at most 1 to $X_{e,i}$. Thus,

$$\begin{aligned}\Pr[X_{v,i} \geq (1 + 1/3)\delta_v \cdot 4c/3] &\leq \mathrm{e}^{-(1/3)^2\delta_v \cdot (4c/3)/(3\min[\delta_v,k])} \\ &\leq \mathrm{e}^{-\max[1,\delta_v/k]4c/3^4}\end{aligned}$$

and

$$\Pr[Y_{e,i} \geq (1+1/3) \cdot 4c/3] \leq \mathrm{e}^{-(1/3)^2 \cdot (4c/3)/3} = \mathrm{e}^{-4c/3^4} \ .$$

Let $A_{v,i}$ be the event that $X_{v,i} > 2\delta_v c$ and $B_{e,i}$ be the event that $Y_{e,i} > 2c$. Since $(4/3)^2 \leq 2$, we can use the above probability estimates to bound the probability that the events $A_{v,i}$ and $B_{e,i}$ appear. Our aim is to show with the help of the LLL that it is possible in the random experiment to assign numbers to the requests so that none of these events emerges. To apply the LLL, we have to bound the dependencies among the events $A_{v,i}$ and $B_{e,i}$.

Each node v can be used by at most $2\delta_v C$ k-cores and each of these k-cores consists of at most $2k(8c \cdot kF)$ nodes and edges. Hence, the set N_v of nodes w that depend on v has a size of at most $(2\delta_v C) \cdot (16ck^2F) = 32c\delta_v k^2 CF$. The same holds for the set E_v of edges that depend on v. On the other hand, each edge e can be used by at most $2C$ k-cores, each of these k-cores consists of at most $2k(8c \cdot kF)$ nodes and edges. Hence, the set N_e of nodes w that depend on e has a size of at most $32ck^2CF$. The same holds for the set E_e of edges that depend on e.

According to the general LLL we have to find values $x_{v,i}$ and $y_{e,i}$ so that

$$\Pr[A_{v,i}] \leq x_{v,i} \prod_{\text{dep. } A_{w,j}} (1 - x_{w,j}) \cdot \prod_{\text{dep. } B_{e,j'}} (1 - y_{w,j'})$$

and

$$\Pr[B_{e,i}] \leq y_{e,i} \prod_{\text{dep. } A_{w,j}} (1 - x_{w,j}) \cdot \prod_{\text{dep. } B_{e,j'}} (1 - y_{w,j'})$$

to guarantee a possibility that none of the events $A_{v,i}$ and $B_{e,i}$ appears. Let us choose

$$x_{v,i} = \mathrm{e}^{-\max[1,\delta_v/k]2c/3^4} \quad \text{and} \quad y_{e,i} = \mathrm{e}^{-2c/3^4} \ .$$

In this case, $\Pr[A_{v,i}] \leq x_{v,i}^2$ and $\Pr[B_{e,i}] \leq y_{e,i}^2$. When choosing $c = 4 \cdot 3^4 \ln(2kF + C)$ we obtain

$$x_{v,i} = (2kF + C)^{-8 \cdot \max[1,\delta_v/k]} \quad \text{and} \quad y_{e,i} = (2kF + C)^{-8} \ .$$

Recall that $C \geq 3k$. Together with the fact that $1 - x \geq \mathrm{e}^{-2x}$ for all $0 \leq x \leq 1/2$ we get in the case of $A_{v,i}$ that

$$\begin{aligned}
\prod_{\text{dep. } A_{w,j}} (1 - x_{w,j}) &\geq \prod_{\text{dep. } A_{w,j}} \mathrm{e}^{-2x_{w,j}} \geq \prod_{\text{dep. } A_{w,j}} \exp\left(-2(2kF + C)^{-8\cdot\max[1,\delta_v/k]}\right) \\
&\geq \prod_{w \in N_v} \exp\left(-2 \cdot (2\delta_w C/c) \cdot (2kF + C)^{-8\cdot\max[1,\delta_w/k]}\right) \\
&\geq \exp\left(-\sum_{w \in N_v} 4\delta_w C/c \cdot (2kF + C)^{-8\cdot\max[1,\delta_w/k]}\right) \\
&\geq \exp\left(-\sum_{w \in N_v} \frac{4C}{c} \cdot \frac{\delta_w}{2kF + C} \cdot (2kF + C)^{-\delta_w/k}\right) \cdot \\
&\quad \exp\left((2kF + C)^{-6\cdot\max[1,\delta_w/k]}\right) \\
&\geq \exp\left(-(32c\delta_v k^2 CF) \cdot \frac{C}{c} \cdot (2kF + C)^{-6}\right) \\
&\geq \exp\left(-32\delta_v \cdot \frac{k^2 C^2 F}{(2kF + C)^6}\right) \\
&\geq \mathrm{e}^{-2\delta_v/k}
\end{aligned}$$

and

$$\begin{aligned}
\prod_{\text{dep. } B_{e,j'}} (1 - y_{w,j'}) &\geq \prod_{\text{dep. } B_{e,j'}} \mathrm{e}^{-2y_{e,j'}} \geq \prod_{\text{dep. } B_{e,j'}} \exp\left(-2(2kF + C)^{-8}\right) \\
&\geq \prod_{e \in E_v} \exp\left(-2 \cdot (2C/c) \cdot (2kF + C)^{-8}\right) \\
&\geq \exp\left(-(32c\delta_v k^2 CF) \cdot (4C/c) \cdot (2kF + C)^{-8}\right) \\
&\geq \exp\left(-128\delta_v \cdot \frac{k^2 C^2 F}{(2kF + C)^{-8}}\right) \\
&\geq \mathrm{e}^{-2\delta_v/k} .
\end{aligned}$$

Hence,

$$\begin{aligned}
&x_{v,i} \prod_{\text{dep. } A_{w,j}} (1 - x_{w,j}) \cdot \prod_{\text{dep. } B_{e,j'}} (1 - y_{w,j'}) \\
&\qquad \geq (2kF + C)^{-8\cdot\max[1,\delta_v/k]} \left(\mathrm{e}^{-2\delta_v/k}\right)^2 \\
&\qquad \geq ((F + C)^{-8\cdot\max[1,\delta_v/k]})^2 \geq \Pr[A_{v,i}] .
\end{aligned}$$

With a similar calculation it can be shown that the LLL inequality also holds for $B_{e,i}$. Thus, it is possible to find an outcome so that none of the bad events is true, which completes the proof of the claim for $c = O(\log(kF + C))$.

To improve this result to $c = O(\log(kF))$, a sequence of refinements has to be done. In a first refinement, $c_1 = O(\log^2 C)$ can be used to show that the requests can be split into sets $S_1, \ldots, S_{C/c_1}$ so that the congestion at each edge is at most $(1 + o(1))c_1$ and the congestion at each node of degree δ is at most $(1 + o(1))\delta c_1$ for each S_i. Afterwards, each S_i is refined separately. $c_2 = O(\log^2 c_1)$ can be used to show that the requests in S_i can be split into sets $S'_1, \ldots, S'_{c_1/c_2}$ so that the congestion at each edge is at most $(1 + o(1))c_2$ and the congestion at each node of degree δ is at most $(1 + o(1))\delta c_2$ for each S'_i. These refinement continue with $c_{j+1} = O(\log^2 c_j)$ until finally a c_j is reached with $c_j = O(\log(kF))$. At that point, the method described above for $c = O(\log(kF + C))$ is used to perform a last refinement with c_j, which yields the claim. The details of the proof are tedious but not new and therefore we skip them here and refer the interested reader to similar techniques used in [77]. $\square$

Claim 4.3.6 *For every set S_i, every elementary request in S_i can be given k-systems of total weight at least $1/4$ such that each of them consists of at most L edges. Furthermore, the congestion at every edge used by an original k-system in S_i is at most $2c + 1/(2k)$, and the congestion at every other edge is at most $1/(2k)$.*

Proof. The proof of the claim is similar to the proof of Theorem 4.2.1. Let us consider some fixed elementary request r and let $p^r_1, \ldots, p^r_{\ell_r}$ be all the disconnected streams in its k-core, $1 \le \ell_r \le k$. Let the first nodes in p^r_i be denoted by $a^r_{i,1}, \ldots, a^r_{i,8c\cdot kF}$ and the last nodes in p^r_i be denoted by $b^r_{i,1}, \ldots, b^r_{i,8c\cdot kF}$. Consider the set of pairs

$$\mathcal{L} = \bigcup_{r\in S_1} \bigcup_{i=1}^{\ell_r} \bigcup_{j=1}^{8c\cdot kF} \{(a^r_{i,j}, b^r_{i,j})\} \ .$$

Due to the node congestion bound in Claim 4.3.5 it is possible to connect each of these pairs $(a^r_{i,j}, b^r_{i,j})$ by flow systems of length at most $2F$ and flow value $f(p^r_i)$ so that the edge congestion is at most $2c \cdot 2F$. Let the flow system from $a^r_{i,j}$ to $b^r_{i,j}$ be denoted by $f^r_{i,j}$. For each request $r = (s,t)$ and each $1 \leq i \leq \ell_r$ and $1 \leq j \leq 8c \cdot kF$, we define a flow system $g^r_{i,j}$ that first moves from s to $a^r_{i,j}$ along p^r_i, then from $a^r_{i,j}$ to $b^r_{i,j}$ along $f^r_{i,j}$, and finally from $b^r_{i,j}$ to t along p^r_i and assign to it a flow value of $f(p^r_i)/(8c \cdot kF)$. This ensures that a total flow of $f(p^r_i)$ is still being shipped for each p^r_i. Furthermore, this allows us to reduce the flow along $f^r_{i,j}$ by a factor of $1/(8c \cdot kF)$. Hence, the edge congestion caused by the $f^r_{i,j}$ for all r, i, j reduces to at most $4c \cdot F/(8c \cdot kF) = 1/(2k)$. Therefore, the additional congestion at any edge is at most $1/(2k)$, which proves the congestion bounds in the claim.

Now consider any given elementary request $r = (s,t)$. For any set of $k-1$ edges, the congestion caused by the flow systems for r is at most $(k-1)(1 + 1/2k) \leq k - 1/2$. Hence, according to Menger's theorem there are k edge-disjoint flows in the system from s to t. Continuing with the same arguments as in Theorem 4.2.1, we obtain a set of k-systems for r with properties as stated by the claim. $\square$

Now that we have short k-systems for every elementary request, we combine them back into the original requests. For a request with demand d this results in a set of k-systems of size at most L each and total flow value at least $d/(4k)$. Let the set of all these k-systems for all requests be denoted by $\mathcal{S}$. Since every k-system has a size of at most L, it could have been a candidate for the BGA. Thus, each of these k-systems must have a witness. Crucially, every edge that has witnesses for these k-systems must be an edge that is not used by *any* of the original k-systems in $\mathcal{O}'$. (This follows directly from the definition of $\mathcal{O}'$.) According to the proof of Claim 4.3.6, the amount of flow from $\mathcal{P}$ traversing any of these edges is at most $1/(2k)$. Let E' be the set of all witness edges.

We choose now one of the k-systems for each request independently at random according to its weight. This will result in a set of k-systems $\mathcal{P}$ in which each request has exactly one k-system and in which the expected amount of flow traversing any edge in E' is at most $1/(2k)$. Next, we assign the original demand of the request to each of these k-systems. This causes the expected amount of flow that traverses any edge in E' to increase from at most $1/(2k)$ to at most $4k \cdot 1/(2k) = 2$.

Now we are ready to bound $\|\mathcal{P}\|$ in terms of $\|\mathcal{B}\|$. For every k-system $h \in \mathcal{S}$, let the indicator variable X_h be one if and only if h is chosen to be in $\mathcal{P}$. We shall look upon $\|\mathcal{P}\|$ as a random variable (though it always has the same value) and bound its value by bounding its expected value $E[\|\mathcal{P}\|]$. In the following we assume that $d(h)$ is the demand associated with the k-system h.

$$
\begin{aligned}
\mathrm{E}[\|\mathcal{P}\|] &\leq \mathrm{E}\left[\sum_{e\in E'} k \cdot \|\mathcal{V}(e,P)\|\right] \leq \sum_{e\in E'} k \cdot \mathrm{E}\left[\sum_{p\in\mathcal{S}:\, e\in p} X_p \cdot \frac{d_i}{k}\right] \\
&\leq \sum_{e\in E'} k \cdot \sum_{p\in\mathcal{S}:\, e\in p} \frac{f(p)\cdot k}{d_i/(4k)} \cdot \frac{d_i}{k} \leq \sum_{e\in E'} k \cdot 4k \sum_{p\in\mathcal{S}:\, e\in p} f(p) \\
&\leq \sum_{e\in E'} 4k^2 \cdot \frac{1}{2k} \leq \sum_{e\in E'} 2k \\
&\leq 4k \sum_{e\in E'} \|\mathcal{W}(e,\mathcal{B})\| \leq \ldots \leq 4L \cdot \|\mathcal{B}\|
\end{aligned}
$$

where the last calculations are done in the same way as in the proof of Lemma 4.3.3. □

Combining the two lemmas proves the theorem. □

We note that if the minimum demand of a request, $d_{\min}$, fulfills $d_{\min} \geq k/\log(kF)$, then one would not need Claim 4.3.5. In particular, if $d_{\min}$ were known in advance, then the k-flow BGA could choose $L = O(k^3F/(d_{\min}/k))$ to achieve an approximation ratio of $O(k^3F/(d_{\min}/k))$. This would allow a smooth transition from the bounds for the k-EDP (where $d_{\min} = k$) to the k-DFP.

4.3.1 Online algorithms for the k-DFP

Next we present a randomized online algorithm for the k-DFP. This algorithm, which we will call the *randomized* k-flow BGA, is an extension of the k-flow BGA algorithm for the offline k-DFP.

First consider the set $\mathcal{O}$ of k-systems for requests accepted by the optimal algorithm. Let $\mathcal{O}_1 \subseteq \mathcal{O}$ consist of k-systems each with demand at least $k/2$. And, let $\mathcal{O}_2 = \mathcal{O} \setminus \mathcal{O}_1$. Exactly one of the following events is true: (1) $||\mathcal{O}_1|| \geq 1/2 \cdot ||\mathcal{O}||$ or (2) $||\mathcal{O}_2|| > 1/2 \cdot ||\mathcal{O}||$.

The randomized k-flow BGA begins by *guessing* which of the two events above shall be true. If it guesses the former, it ignores all requests with demand less than $k/2$ and runs the regular k-flow BGA on the rest of the requests. If it guesses the latter, it correspondingly ignores all requests with demand at least $k/2$ and runs the k-flow BGA on the rest. We now prove the following theorem.

Theorem 4.3.7 *Given any unit-capacity network G with flow number F, the expected competitive ratio of the randomized k-flow BGA for the online k-DFP without deletions is $O(k^3 F \log(kF))$, when run with parameter $L = O(k^3 F \log(kF))$.*

Proof. The proof runs on exactly the same lines as the proof for Theorem 4.3.1, except that we have to prove Lemma 4.3.2 in our changed situation. Note that the original proof for Lemma 4.3.2 relied on the fact that requests were sorted in a non-decreasing order before being considered. That need not be true here. $\mathcal{B}$ denotes, as usual, the k-systems for requests accepted by the randomized k-flow BGA.

Consider the case when the algorithm guesses that $||\mathcal{O}_1|| \geq 1/2 \cdot ||\mathcal{O}||$. We claim that for any stream $q \in \mathcal{O}_1$ and edge e, if q has a witness on e then $\|\mathcal{W}(e, \mathcal{B})\| \geq 1/2$. Let q be witnessed by p, a stream in $\mathcal{B}$. Now, since the algorithm only considers requests with demand at least $k/2$, $f(p) \geq 1/2$. The claim follows since $\|\mathcal{W}(e, \mathcal{B})\| \geq f(p)$. Following the rest of the proof for

Theorem 4.3.1, substituting $\mathcal{O}_1$ for $\mathcal{O}$, shows that in this case the randomized k-flow BGA will have a competitive ratio $O(k^3 F \log(kF))$.

Consider now the case when the algorithm guesses $||\mathcal{O}_2|| \geq 1/2 \cdot ||\mathcal{O}||$. We again claim that even in this case for any stream $q \in \mathcal{O}_2$ and edge e, if q has a witness on e then $\|\mathcal{W}(e, \mathcal{B})\| \geq 1/2$. ¿From the definition of witnessing, we have $F(e, q) + f(q) > 1$. Next, from the definition of $\mathcal{O}_2$, $f(q) < 1/2$. The claim follows as $\|\mathcal{W}(e, \mathcal{B})\| \geq F(e, q)$. As in the previous case, the rest of the proof for Theorem 4.3.1 applies here too; substitute $\mathcal{O}_2$ for $\mathcal{O}$.

The competitive ratio in both cases is $O(k^3 F \log(kF))$. Note that the algorithm may guess incorrectly which event shall be true. But that just reduces the expected competitive ratio by a factor of 2. $\square$

4.3.2 Consequences for the k-splittable flow problem

Our proof for the k-DFP has interesting consequences for the *k-splittable flow problem* (k-SFP) be defined as follows: given a graph with edge capacities and a set of requests i with demand d_i, find a subset of requests of maximum total demand for which it is possible to select up to k flow paths for each accepted request i whose flow values sum up to d_i so that all the capacity constraints are kept.

Certainly, the k-SFP is a relaxation of the UFP. However, this does not necessarily make the problem easier, since not only the online algorithm but also the optimal offline algorithm has more freedom in choosing flow paths for its requests.

Theorem 4.3.8 *Let G be a unit-capacity network with flow number F. The approximation ratio of the k-BGA for the k-SFP, when run on requests ordered according to non-increasing demands with parameter $L = 3kF$ is $O(k^2 F)$.*

Proof Sketch. We use arguments similar to those in the proofs of the approximation ratios for the k-DFP. For each stream $q \in \mathcal{B}$ or $q \in \mathcal{O}$, let $d(q)$ denote the demand of the request belonging to that stream. For a set

$\mathcal{Q}$ of streams let $||\mathcal{Q}|| = \sum_{q\in\mathcal{Q}} d(q)$. Also, for an edge $e \in E$ and a stream q, let $D(e,q)$ denote the sum of the demands of all streams in $\mathcal{B}$ passing through e whose demand is at least as large as the demand associated with q, i.e., $D(e,q) = ||\{p \mid p \in \mathcal{B},\ e \in p,\ d(p) \geq d(q)\}||$. A stream $p \in \mathcal{B}$ is a *witness* for a stream q if $d(p) \geq d(q)$ and p and q intersect in an edge e with $D(e,q) + d(q) > 1$. For each edge e let $\mathcal{W}(e,\mathcal{B})$ denote the set of streams in $\mathcal{B}$ that serve as witnesses on e. Similarly, for each edge e let $\mathcal{V}(e,\mathcal{Q})$ denote the set of streams in $\mathcal{Q}$ that have witnesses on e. We also say that a k-system has a witness on an edge e if any of its k streams has a witness on e.

Lemma 4.3.9 *For any stream q and edge e, if q has a witness on e then $\|\mathcal{W}(e,\mathcal{B})\| \geq 1/2$.*

Proof. Let p be a witness of q on e. Assume, by contradiction, that $D(e,q) < 1/2$. Then it easily follows that $d(p) < 1/2$. Since $d(q) \leq d(p)$ and $D(e,q) + d(q) > 1$ by the definition of a witness, we have a contradiction. □

Let $\mathcal{O}' \subset \mathcal{O}$ be the set of flow systems that are larger than L and that correspond to requests *not* accepted by the BGA and that do *not* have a witness in $\mathcal{B}$. The next two bounds on $||\mathcal{O} \setminus \mathcal{O}'||$ and $||\mathcal{O}'||$ complete the proof.

Lemma 4.3.10 (Case 1) $||\mathcal{O} \setminus \mathcal{O}'|| \leq (1 + 2kL) \cdot ||\mathcal{B}||$.

Proof. We partition $\mathcal{O} \setminus \mathcal{O}'$ into two sets. Let $\mathcal{O}_1 \subseteq \mathcal{O} \setminus \mathcal{O}'$ consist of all the flow systems corresponding to requests accepted by the BGA and let $\mathcal{O}_2 = (\mathcal{O} \setminus \mathcal{O}') \setminus \mathcal{O}_1$. Note that each flow system in $\mathcal{O}_2$ must have a witness in $\mathcal{B}$. Let $E' \subseteq E$ denote the set of all edges on which some flow system from $\mathcal{O}_2$ has a witness. We then have

$$\|\mathcal{O}_2\| \leq \sum_{e\in E'} k\|\mathcal{V}(e,\mathcal{O}_2)\| \leq \sum_{e\in E'} k \leq \sum_{e\in E'} k \cdot 2\|\mathcal{W}(e,\mathcal{B})\|$$

The first inequality holds since every demand is counted k-fold in $\|\mathcal{O}_2\|$. Since all flow systems in $\mathcal{B}$ are of length at most L, we have

$$\sum_{e\in E'} \|\mathcal{W}(e,\mathcal{B})\| \leq \sum_{\text{streams } p\in\mathcal{B}} |p| \cdot d(p) \leq \sum_{k-\text{systems } s\in\mathcal{B}} L \cdot d(s) \leq L \cdot \|\mathcal{B}\| \, .$$

The last inequality is tight since the flow systems in $\mathcal{B}$ may just consist of single paths. This completes the proof. $\square$

In the next lemma we bound $\|\mathcal{O}'\|$ by first transforming the large flow systems in $\mathcal{O}'$ into a set $\mathcal{S}$ of small flow systems and then bounding $\|\mathcal{S}\|$ in terms of $\|\mathcal{B}\|$.

Lemma 4.3.11 (Case 2) *$\|\mathcal{O}'\| = O(kL \cdot \|\mathcal{B}\|)$.*

Proof Sketch. In this case we are left with the large flow systems. Using Lemma 4.1.1, we can replace all streams which are longer than $L/k = 3F$ by a system of streams of length at most $3F$ by increasing the edge congestion to at most 3 (1 due to the original streams and 2 due to the shortcut streams). Using a random experiment to round the short streams down to at most k for each request, we arrive at flow systems that could have been used by the BGA while having an expected congestion of at most 3. Using Lemma 4.3.9 it is then possible to show in the same flavour as in the proof of Theorem 4.3.1 that $\|\mathcal{O}'\| = O(kL \cdot \|\mathcal{B}\|)$. $\square$

Combining Lemma 4.3.10 and 4.3.11 results in the theorem. $\square$

4.3.3 Consequences for the integral splittable flow problem

The *Integral Splittable Flow Problem* was defined in [39] as follows: given a graph G and a set of terminal pairs with integral demands d_i, find a maximum subset of the pairs for which it is possible to fulfill the demand of each pair by routing at least as much flow as it demands, while maintaining the capacity constraints at each edge and ensuring that the flow through any edge is integral.

The best previously known upper bound for this problem is $O(\sqrt{m d_{\max}} \log^2 m)$ [39]. We will demonstrate that we can improve this upper bound for many relevant cases.

Without the "no-bottleneck" condition: In case we do not have the condition that any edge can support all the demand for a given request i.e. we *do not* have the condition that $d_{\max} \leq 1$, we simply use Theorem 4.3.1 with the parameter k set to $d_{\max}$ to get the following corollary:

Corollary 4.3.12 *Given any unit-capacity network G with flow number F, for the online ISF in which no request has demand more than $d_{\max}$, the expected competitive ratio of the randomized $d_{\max}$-flow BGA is $O(d_{\max}^3 F \log(kF))$, when run with parameter $L = O(d_{\max}^3 F \log(kF))$.*

With the "no-bottleneck" condition: We now show that for a uniform capacity network of capacity C, if $d_{\max} \leq C$, we can prove a much better bound for the ISF. This follows with some additions from the proof of Theorem 4.1 from [50].

The algorithm we propose is the one used in [50]: the elementary BGA, with parameter $4 \cdot F$. This involves finding a *single*, i.e. unsplit, path of length at most $4 \cdot F$ for a request (s_i, t_i) which can satisfy the entire demand d_i for that request, in the residual graph left after satisfying previously accepted requests. If such a path cannot be found, this request is rejected.

Theorem 4.3.13 *Given any uniform-capacity network G of capacity C with flow number F, for the online ISF in which no request has demand more than $d_{\max} \leq C$, the expected competitive ratio of the elementary BGA when run with parameter $L = 4 \cdot F$ is $O(F)$.*

Proof Sketch.

As before, let the optimal solution be denoted as $\mathcal{O}$ and the BGA's solution be denoted as $\mathcal{B}$. Let $\mathcal{O}' \subseteq \mathcal{O} \setminus \mathcal{B}$ be those requests which the optimal satisfies by splitting their flow. In other word, for all the requests in $\mathcal{O} \setminus (\mathcal{O}' \cup \mathcal{B})$, the optimal algorithm routes the entire demand of each request along a single path.

From the proof of Theorem 4.1 of [50] it follows that:

$$||\mathcal{O} \setminus \mathcal{O}'|| \leq (16F + 1)||\mathcal{B}||$$

To bound $||\mathcal{O}'||$, we transform the optimal solution's split flows into a set of unsplit flows, $\mathcal{S}$, which do not violate capacity constraints by more than twice on the average. To do this, consider the flow for $(s_i, t_i) \in \mathcal{O}'$. This can be separated into d_i possibly overlapping streams of unit flow. First we use the Shortening Lemma (Lemma 3.1 from [50]) to transform all the streams in $\mathcal{O}'$ into streams of length at most $4F$ at the cost of increasing the congestion on each edge to at most twice its capacity. We now perform a random experiment on these d_i streams: we choose one of them, uniformly at random, and decide to send all the d_i flow along its path. The set $\mathcal{S}$ consists of the set of unsplit flows determined by this random experiment.

Claim 4.3.14 *The average flow through any edge in G due to the flows in $\mathcal{S}$ is at most $2C$.*

Proof. Consider any edge. It has at most $2C$ shortened streams flowing through it, if we consider the separate streams from a given request individually. Let the demands of the requests they are associated with be $d_1, d_2, \ldots d_{2C}$. The average congestion on any given edge is

$$\sum_{i=1}^{2C} \frac{1}{d_i} \cdot d_i = 2C$$

□

The flows in $\mathcal{S}$ were candidates for the BGA and since they were not taken, it means that they were witnessed by $\mathcal{B}$. Given the claim we can now say, using arguments similar to the ones in Theorem 4.1 of [50], that $E[||\mathcal{S}||] \leq 32Fcdot||\mathcal{B}||$. This implies that there is an outcome $\mathcal{S}'$ of the random experiment for which $||\mathcal{S}'|| = ||\mathcal{O}'|| \leq 32F \cdot ||\mathcal{B}||$. □

Essentially what Theorem 4.3.13 tells us is that if the no-bottleneck condition is satisfied then splitting the flows does not buy us anything in terms of an approximation ratio for the problem.

Additionally we note without going into detail that it is possible to give a randomized online algorithm similar to the one described in Section 4.3.1 for the ISF with the no-bottleneck conditions which achieves a competitive ratio of $O(F)$.

4.4 Conclusions

Several open problems remain. For example, what is the best competitive ratio a deterministic algorithm can achieve for the k-EDP? We suspect that it is $O(k \cdot F)$, but it seems to be very hard to prove. It might be easier to show this for the k-splittable flow problem first. Another open problem that might be easier to solve is to improve the $O(k^3 F \log(kF))$ bound for the k-DFP to $O(k^3 F)$.

Chapter 5

Conclusion

There is a significant amount of work already being done by algorithms researchers which could be classified as Internet Algorithmics. Apart from the areas we have mentioned in Chapter 1, and the new areas we have covered in Chapters 2, 3 and 4, there are several other areas in which some progress has been made. An important example is that of web searching where algebraic techniques have been used to organize the welter of information floating around the World Wide Web (see e.g. [48].) We feel that there is already a move towards Internet Algorithmics in the research community. There is a need to recognize the special characteristics of this area and the interrelationship between the various problems which it throws up.

The network of connections, so to speak, between the these problems and the layered nature of the issues they arise from make for a fascinatingly rich problem space. But this also means that the process of formulating clean theoretical problems to attack becomes that much harder. While abstracting away from details is essential to the business of algorithmic research, it is only when we keep the larger picture in mind that we can ensure that we do not make assumptions which make our solutions irrelevant. This is one of the important considerations that has led to our thesis that the vast and diverse area of Internet Algorithmics has to be viewed in a unified way.

In this dissertation we have tried to make a case for Internet Algorithmics as a coherent and important new area of study. It is our contention that investigating the theoretical underpinnings of the processes of the Internet is a fruitful endeavour which can potentially lead not only to a vast increase in the usefulness and quality of the Internet as a medium for communication and commerce, but also to a furthering in our understanding of the phenomenon of computing.

Bibliography

[1] J. Abbate. *Inventing the Internet*. The MIT Press, 1999.

[2] N. Abramson. *Information Theory and Coding*. McGraw-Hill, New York, 1963.

[3] G. M. Adel'son-Vel'skii and Y. M. Landis. An algorithm for the organisation of information. *Doklady Akademii Nauk SSSR*, 146:263–6, 1962. English translation in *Soviet Math. Dokl.* **3**:1259-62, 1962.

[4] M. Adler. Tradeoffs in probabilistic packet marking for ip traceback. In *Proceedings of the 34th ACM Symposium on the Theory of Computing*, pages 407–418, 2002.

[5] R. Adler and Y. Azar. Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 1999.

[6] E. Aginam. Experts list challenges facing Internet growth in Nigeria. *Vanguard*, 21st November, 2001. Available online at http://www.vanguardngr.com/.

[7] A. V. Aho, D. S. Johnson, R. M. Karp, S. R. Kosaraju, C. C. McGeoch, C. H. Papdimitriou, and P. Pevzner. Emerging opportunities for theoretical computer science. Technical Report UW-CSE-96-03-03, University of Washington, 1996.

[8] L. Amsaleg, M. Franklin, and A. Tomasic. Dynamic query operator scheduling for wide-area remote access. *Distributed and Parallel Databases*, 6(3):217–246, July 1998.

[9] Anamorph: Internet statistics. http://www.anamorph.com/docs/stats/stats.html.

[10] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over mpls. Request for Comments (RFC2702), September 1999. Available online at http://www.ietf.org/rfc/rfc2702.txt.

[11] B. Awerbuch and Y. Azar. Blindly competitive algorithms for pricing and bidding. Manuscript.

[12] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 32–40, Palo Alto, California, 3–5 Nov. 1993.

[13] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosen. Competitive non-preemptive call-control. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, 23-25 Jan 1994.

[14] A. Baveja and A. Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *MOR: Mathematics of Operations Research*, 25, 2000.

[15] S. M. Bellovin. ICMP traceback messages: draft-bellovin-itrace-00.txt. Internet Draft. Available online at http://www.research.att.com/ smb/papers/draft-bellovin-itrace-00.txt.

[16] S. W. Bent, D. D. Sleator, and R. E. Tarjan. Biased search trees. *SIAM Journal on Computing*, 14(3):545–68, 1985.

[17] U. Black. *MPLS and Label Switching Networks*. Prentice Hall Series in Advanced Communications Technologies. Prentice Hall PTR, 2001.

[18] A. Bley. On the complexity of vertex-disjoint length-restricted path problems. Technical Report SC-98-20, Konrad-Zuse-Institut Berlin, 1998.

[19] B. Bollobás. *Modern Graph Theory*. New York: Springer, 1998.

[20] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[21] D. Bunnell. *The eBay Phenomenon: Business secrets behind the world's hottest Innternet company*. John Wiley and Sons, New York, 2000. with Richard A. Luecke.

[22] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. In *Proceedings 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, 2002.

[23] Cisco Systems. Characterizing and tracing packet floods using Cisco routers.
http://www.cicso.com/warp/public/707/22.html.

[24] I. Clarke. A distributed decentralized information storage and retrieval system. Unpublished, 1999.

[25] I. Clarke, B. Wiley, T. Hong, and O. Sandberg. Freenet: A distributed anonymous information storage and retrieval system. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, volume LNCS 2009, New York, 2001. Springer.

[26] Committee on the Internet in the Evolving Information Infrastructure. *The Internet's Coming of Age*. National Academy Press, Washington, D.C., 2001. A Committee formed by the Computer Science and Telecommunications Board, Commission on Physical Sciences, Mathematics, and Applications, National Research Council.

[27] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley, New York, 1997.

[28] V. Crishna, N. Baqai, B. R. Pandey, and F. Rahman. Telecommunications infrastructure: A long way to go. http://www.sasianet.org/telecominfrastr.html.

[29] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP Traceback. In *Proceedings of Network and Distributed System Security Symposium*, San Diego, California, February 2001.

[30] R. Engelbrecht-Wiggans, M. Shubik, and R. M. Stark, editors. *Auctions, Bidding, and Contracting: Uses and Theory*. New York University Press, 1983.

[31] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, R. Rado, and V. Sós, editors, *Infinite and Finite Sets (to Paul Erdős on his 60th birthday)*, pages 609–627. North-Holland, Amsterdam, 1975.

[32] F. Ergun, S. C. Sahinalp, J. Sharp, and R. K. Sinha. Biased dictionaries with fast inserts/deletes. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 483–91, 2001.

[33] F. Ergun, S. C. Sahinalp, J. Sharp, and R. K. Sinha. Biased skip lists for highly skewed access patterns. In *Proceedings of the 3rd Workshop on Algorithm Engineering and Experiments*, volume 2153 of *Lecture Notes in Computer Science*, pages 216–29. Springer, 2001.

[34] A. Faraaz. Ab ke ham bichhde. In *Dard-Aashob*, pages 215–216. Yousuf Publishers, Bank Road, Rawalpindi, Pakistan, c. 1965.

[35] J. Feigenbaum and R. E. Tarjan. Two new kinds of biased search trees. *Bell System Technical Journal*, 62(10):3139–58, 1983.

[36] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world wide web: A survey. *ACM SIGMOD Record*, 27(3):59–74, 1998.

[37] S. Gibson. The strange tale of the denial of service attacks against grc.com.
http://grc.com/dos/grcdos.htm.

[38] L. J. Guibas and R. Sedgewick. A dichromatic framework for balanced trees. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 8–21, 1978.

[39] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 19–28, 1999.

[40] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, MA., 1997.

[41] S. Huddleston and K. Mehlhorn. A new data structure for representing sorted lists. *Acta Informatica*, 17:157–84, 1982.

[42] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and subset sum problems. *Journal of the ACM*, 22:463–468, 1975.

[43] Internet Architecture Board. IAB technical comment on the unique DNS root. Request for Comments (RFC2826, May 2000. Available online at http://www.ietf.org/rfc/rfc2826.txt.

[44] Internet Architecture Board, Network Working Group, B. Carpenter, ed. Architectural principles of the Internet. Request for Comments (RFC1958), June 1996. Available online at http://www.ietf.org/rfc/rfc1958.txt.

[45] Internet Software Consortium. Internet domain survey. http://www.isc.org/ds/WWW-200201/index.html, Jan 2002.

[46] D. Karger, E. Lehman, D. Lewin, M. L. T. Leighton, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the 29th annual ACM Symposium on Theory of Computing*, pages 654–663, May 1997.

[47] J. Kleinberg. *Approximation Algorithms for Disjoint Paths Problems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[48] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[49] P. Kolman and C. Scheideler. Simple, routing-based on-line algorithms for maximum disjoint paths problem. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 38–47, 2001.

[50] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. In *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[51] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, pages 404–413, Trier, Germany, 4–6 Mar. 1999.

[52] E. L. Lawler. Fast approximation algorithms for knapsack problems. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, pages 206–213, 1977.

[53] S. Leonardi and A. Marchetti-Spaccamela. On-line resource management with applications to routing and scheduling. In *Proceedings of the 22nd In-*

ternational Colloquium on Automata, Languages and Programming, pages 303–314, 1995.

[54] Limewire. Modern peer-to-peer file-sharing over the Internet. http://www.limewire.com/index.jsp/p2p.

[55] R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 302–11, 23-25 Jan 1994.

[56] D. Lucking-Reilly. Auctions on the Internet. *Journal of Industrial Economics*, 48(3):227–253, September 2000.

[57] Matrix Science. http://www.mids.org/.

[58] K. Mehlhorn and S. Näher. Algorithm design and software libraries: Recent developments in the LEDA project. In *Proc. IFIP 12th World Computer Congress*, volume 1, pages 493–505. Elsevier, 1992.

[59] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[60] P. Mockapetris. Domain names - concepts and facilities. Request for Comments (RFC0882), November 1983. Available online at http://www.ietf.org/rfc/rfc0882.txt.

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

[62] J. I. Munro, T. Papadakis, and R. Sedgewick. Deterministic skip lists. In *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, pages 367–75, 1992.

[63] Napster. http://www.napster.com/.

[64] J. Naughton. *A Brief History of the Future: The origins of the Internet*. Weidenfield and Nicolson, London, 1999.

[65] A. M. Odlyzko. Internet growth: Myth and reality, use and abuse. *J. Computer Resource Management*, 102:23–27, spring 2001.

[66] K. Park and H. Lee. On the effectiveness of probabilistic packet marking for IP traceback under Denial of Service attack. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2001.

[67] W. Pugh. Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–76, June 1990.

[68] W. Pugh. Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–76, June 1990.

[69] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of SIGCOMM '01*, pages 161–172, San Diego, California, August 2001.

[70] L. G. Roberts. Beyond Moore's law: Internet growth trends. *Computer*, 33(1):117–119, 2000.

[71] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Request for Comments (RFC3031), January 2001. Available online at
http://www.ietf.org/rfc/rfc3031.txt.

[72] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

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

[74] T. Roughgarden. The price of anarchy is independent of the network topology. In *Proceedings of the 34th ACM Symposium on the Theory of Computing*, 2002. 428-437.

[75] S. Sahni. Approximation algorithms for the 0/1 knapsack problem. *Journal of the ACM*, 22:115–124, 1975.

[76] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, 1984.

[77] C. Scheideler. *Universal Routing Strategies for Interconnection Networks*. Lecture Notes in Computer Science 1390, Springer Verlag, 1998.

[78] T. Schlossnagle. Skip lists: Implementation effects on the Spread Group Communication System and other applications. Manuscript.

[79] T. Schlossnagle. Personal Communication, February 2002.

[80] R. Seidel and C. R. Aragon. Randomized search trees. *Algorithmica*, 16(4/5):464–97, 1996.

[81] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–86, 1985.

[82] C. W. Smith. *Auctions: The Social Construction of Value*. The Free Press, 1989.

[83] Spread Concepts LLC. The Spread toolkit. http://www.spread.org.

[84] D. L. Stewart, P. T. Maginnis, and T. Simp. Who is at the door: The SYN denial of service. *Linux Journal*, 38, June 1997. Available online at http://www.linuxjournal.com/article.php?sid=2056.

[85] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of SIGCOMM '01*, pages 149–160, San Diego, California, August 2001.

[86] A. S. Tanenbaum. *Computer Networks, Third Edition*. Prentice Hall, 1996.

[87] Telcordia Technologies. Evaluating the size of the Internet. http://www.netsizer.com/.

[88] United States Census Bureau. http://www.census.gov/.

[89] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, August 2000.

[90] F. Wang. Distributed Geographic Informations Systems on the Web. In *Handbook of Internet Computing*, chapter 17, pages 393–414. CRC Press, Boca Raton, Florida, 2000.

[91] D. B. West. *Introduction to Graph Theory*. Prentice-Hall, 1996.

[92] P. R. Wurman, M. P. Wellman, and W. E. Walsh. A parametrization of the auction design space. *Games and Economic Behaviour*, 35(1/2):304–338, April 2001.

[93] A. C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

Vita

Amitabha Bagchi was born in New Delhi in 1974. He completed his schooling at Delhi Public School, R. K. Puram in 1992. He studied Computer Science and Engineering at the Indian Institute of Technology, Delhi for his Bachelors in Technology which he obtained in 1996. From 1996 to 2002 he was at Johns Hopkins University, Baltimore, pursuing his PhD in the Computer Science department. Along the way he earned his Masters in Computer Science in 1999. He is currently working as a postdoctoral associate in the Information and Computer Science department at the University of California, Irvine.