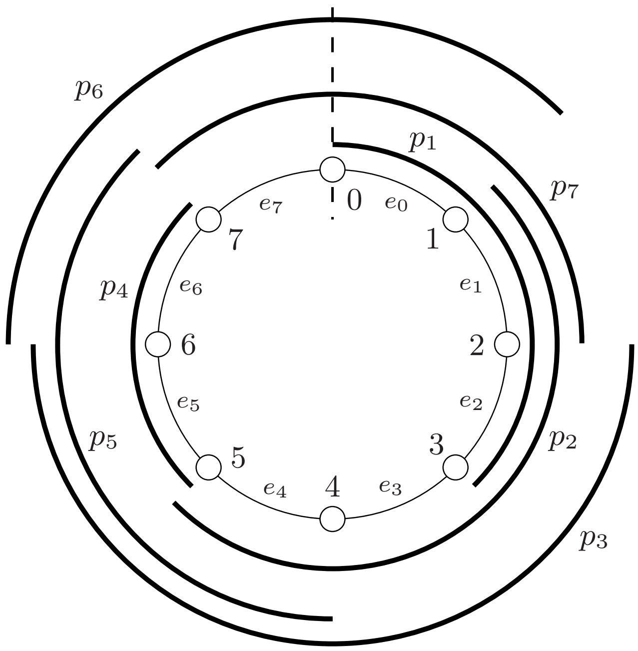
Diss. ETH No. 15370

Udo Adamy

Call Admission Control and On-line Interval Coloring



2003

DISS. ETH No. 15370, 2003

Call Admission Control and On-line Interval Coloring

A dissertation submitted to the
Swiss Federal Institute of Technology, ETH Zürich
for the degree of Doctor of Technical Sciences

presented by
Udo Adamy
Dipl. Inf. Universität Saarbrücken, Germany
born 22 November 1973 in Bernkastel–Kues, Germany

submitted to
Prof. Dr. Emo Welzl, ETH Zürich, examiner
Prof. Dr. Thomas Erlebach, ETH Zürich, co–examiner

For Franziska

Abstract

Driven by the wish of a globally connected society telecommunications became one of the leading growth sectors in the world economy. The prevalence of personal computers and cellular phones in everyday life makes data communication an important issue. Huge networks including wireless, fiber-optic and satellite networks were built to carry the increasing amount of data traffic.

This thesis is concerned with theoretical aspects of the optimization problems that have to be addressed in the operation of such communication networks. The main focus is put on call admission control problems in various network topologies. Call admission is the problem of deciding for a given set of connection requests (or calls) which of them to accept and which to reject such that the accepted requests do not exceed the capacity of any link and some objective function is maximized.

First we study call admission control variants, in which each call requires one unit of bandwidth on all links contained in its path, and the goal is to maximize the number of accepted calls. For chain networks we generalize an algorithm by Carlisle and Lloyd from equal to arbitrary capacities of the links while keeping its runtime linear. For ring networks with arbitrary capacities we present a polynomial-time algorithm that solves the problem optimally provided that the calls are given as pre-specified paths in the ring. This algorithm is based on a decision procedure that checks whether a solution with at least k paths exists by iteratively solving the problem for a chain network.

Further we investigate more general versions of call admission control problems in star networks, in which the calls have a starting time and a duration, and are associated with a positive bandwidth requirement and a profit value that is gained if the call is accepted. The goal is to maximize the sum of the profits of the accepted calls. We give an 18-approximation algorithm for this call admission control variant in stars resolving an open problem posed by Erlebach. Our method is randomized and uses the local ratio technique, which was developed by Bar-Yehuda and Even and first used in our context by Bar-Noy et al. We show how our algorithm can be derandomized by means of Hadamard matrices keeping the same approximation ratio. Moreover, only a slight modification makes the method capable of dealing with call alternatives, at most one of which may be accepted for each call. We achieve an approximation ratio of 24 in this case.

In addition, we examine the complexity of call admission control problems in star networks. If every link has unit capacity and each call needs one unit of bandwidth on its links, has unit profit and a fixed duration, the problem

can be solved optimally in polynomial time provided that the number of different starting times of the calls is at most two. Already for three different starting times, we prove this restricted variant to be $\mathcal{APX}$-hard by a reduction from the bounded maximum three-dimensional matching problem. A simple greedy algorithm approximates this problem with arbitrary starting times and durations within a factor of two from the optimal value. The parameterized version of that problem is fixed-parameter tractable, if the number of rejected calls is taken as the parameter. The variant with arbitrary capacities and profits can be approximated with an approximation ratio of three.

Another optimization problem that we address in this thesis is the on-line coloring problem of fixed paths with bandwidths in chain networks. This problem asks for an assignment of colors to the paths such that the sum of bandwidths of paths that are assigned the same color does not exceed one on any link. The goal is to find an assignment using as few colors as possible. We present an 195-competitive on-line algorithm.

Zusammenfassung

Die Telekommunikation nimmt auf dem Weltmarkt eine führende Stellung ein. Computer und Handy sind aus dem alltäglichen Leben kaum mehr wegzudenken. Im Mittelpunkt steht die weltweite Übertragung von Daten in Kommunikationsnetzwerken aller Art.

In dieser Arbeit untersuchen wir die theoretischen Aspekte einiger Optimierungsprobleme, die bei einer effizienten Nutzung solcher Kommunikationsnetzwerke auftreten. Der Schwerpunkt liegt dabei in der Analyse von Algorithmen zur Zugangskontrolle in verschiedenen Netzwerktopologien. Die Zugangskontrolle muss für eine Menge von Verbindungsanfragen entscheiden, welche Anfragen angenommen und welche abgelehnt werden, so dass die Kapazitäten des Netzwerks nicht überschritten werden und die akzeptierten Anfragen eine Zielfunktion maximieren.

Im ersten Teil betrachten wir Varianten der Zugangskontrolle, in denen jede Anfrage eine Einheit an Bandbreite benötigt und die Zielfunktion die Anzahl der akzeptierten Anfragen ist. Für Netzwerke, die aus einer Kette bestehen, verallgemeinern wir einen Algorithmus von Carlisle und Lloyd von gleichen zu beliebigen Kapazitäten. Die lineare Laufzeit bleibt dabei erhalten. Für Ringnetzwerke präsentieren wir einen Polynomialzeitalgorithmus, der das Problem optimal löst, wenn die Anfragen als Pfade im Ring vorgegeben sind. Das Herzstück dieses Algorithmus ist eine Entscheidungsprozedur, die prüft, ob eine Lösung mit mindestens k Pfaden existiert. Dies geschieht durch wiederholtes Lösen von Zugangskontrollproblemen in Ketten.

Desweiteren untersuchen wir allgemeinere Zugangskontrollprobleme in Sternnetzwerken, in denen die Anfragen durch eine Reihe von Parametern definiert sind. Sie haben sowohl eine Startzeit als auch eine Dauer, fordern eine bestimmte Bandbreite an und geben einen gewissen Profit, der nur dann erzielt wird, wenn die Anfrage angenommen wird. Diesmal maximieren wir die Summe der Profite der akzeptierten Anfragen. Wir entwickeln einen Approximationsalgorithmus mit Güte 18 für dieses Problem und lösen damit ein offenes Problem, das Erlebach gestellt hat. Unser Algorithmus ist randomisiert und basiert auf der "'local ratio technique"', die von Bar-Yehuda und Even entwickelt und von Bar-Noy et al. zum ersten Mal in unserem Zusammenhang verwendet wurde. Unsere Methode kann mittels Hadamard-Matrizen derandomisiert werden, ohne die Güte zu beeinträchtigen. Falls die Verbindungsanfragen verschiedene Alternativen zur Auswahl anbieten, approximiert ein leicht veränderter Algorithmus diesen Fall mit einer Güte von 24.

Betreffend der Komplexität von Zugangskontrollproblemen in Sternnetzwer-

ken erzielen wir folgende Ergebnisse. Wenn alle Verbindungen des Sterns die gleiche Kapazität aufweisen und alle Anfragen jeweils genau diese Kapazität anfordern, den gleichen Profit und eine feste Dauer haben, dann lässt sich dieses Problem in polynomieller Zeit optimal lösen, sofern die Anfragen nur zu zwei verschiedenen Zeiten starten. Lassen wir jedoch drei verschiedene Startzeiten zu, zeigen wir, dass das Problem $\mathcal{APX}$-schwer ist. Ein einfacher Algorithmus nähert die Variante mit beliebigen Startzeiten und beliebigen Dauern auf einen Faktor 2 des optimalen Wertes an. Für die parametrisierte Version mit der Anzahl abgelehnter Anfragen als Parameter beschreiben wir einen Festparameter-Algorithmus. Wenn wir beliebige Kantenkapazitäten und beliebige Profite zulassen, dann kann das Problem mit einer Approximationsgüte von 3 angenähert werden.

Im abschliessenden Teil behandeln wir die Online-Variante eines Intervallfärbungsproblems mit verschiedenen Bandbreiten auf Ketten. Dabei müssen den Intervallen Farben zugeordnet werden, so dass die Summe der Bandbreiten der Intervalle gleicher Farbe auf keiner Verbindung den Wert Eins überschreitet. Gesucht ist eine Zuweisung, die möglichst wenige Farben benötigt. Für dieses Problem entwickeln wir einen Algorithmus, der 195-kompetitiv ist.

Acknowledgments

I would like to thank my advisor Emo Welzl for giving me the great opportunity to work in his research group, for his support, and for providing an excellent working atmosphere.

I warmly thank my co-advisor Thomas Erlebach for being in charge of me. His constant advice in many encouraging meetings was most helpful.

Many thanks to all members of my research group, who made life on our floor happier: Bernd Gärtner, Jochen Giesen, Johannes Blömer, Lutz Kettner, Michael Hoffmann, Artur Andrzejak, Alex Below, Christoph Ambühl, Falk Tschirschnitz, Matthias John, Alexander May, Samuele Pedroni, Uli Wagner, József Solymosi, Sven Schönherr, Csaba Tóth, Tibor Szabó, Ingo Schurr, Bettina Speckmann, Kaspar Fischer, Yoshio Okamoto, Péter Csorba, Miloš Stojaković, Dieter Mitsche, Simon Spalinger, and Robert Berke. Tanja Krenn, Susanne Böhm, Franziska Hefti-Widmer, and Floris Tschurr helped me with the administrative stuff.

Special thanks to Yoshio Okamoto for proofreading most parts of my thesis. His comments and suggestions helped to improve this work.

I am grateful to my parents for supporting me in the best of all possible ways.

My friends Jochen and Nicole Comes, Stefan and Birgit Funke, Christian Sohler and Valentina Damerow, Birgit Ahrens, Georg Demme, Markus Denny, Ralph Osbild, Torsten Bücheler and Christina Kullen, Inma Jurado, Sagrario Zubieta, and Marta Garcia, I thank you for the time we spent together and for keeping in touch, even if I did not.

Many thanks go to all members of the ABV badminton club and the ASVZ cycling club. I really enjoyed the exciting badminton matches, and the exhausting but wonderful rides around Zurich and in the alps.

Eventually, I want to thank my girlfriend Franziska Stössel for always being there for me. Without her constant support this thesis would not have been possible. Finally, I would like to thank my unborn son Gian. I am very excited to see you from face to face and to hold you in my arms.

Udo Adamy, December 2003

Contents

Chapter 1

Introduction

Everyday the world seems to be moving faster and faster with an explosive growth of telecommunication and the World Wide Web. New services such as tele-conferencing and video on demand, as well as an ever-growing volume of traditional data, call for communication networks with ever-increasing capacities to enhance the quality of service.

Before the existence of computers linked together to communicate and share resources, telephone companies built an international network to carry telephone calls. These wide area networks (WANs) used primarily copper cables and were realized by circuit switching, which means that a connection is established by reserving a channel on all links of a path from the sender to the receiver as long as the connection is active. Soon the bandwidth limitations of copper cables became apparent, and many WANs were upgraded to optical fiber cables because of its potential for almost unlimited bandwidth.

When computers were becoming more prevalent in the office, linking them together was desirable. However, since the existing voice-optimized WANs did not send digital computer data very efficiently, separate WANs were build specifically to carry data traffic. Digital data were transmitted by packet switching. This term designates a procedure where messages are divided into small packets before they are sent. Each packet is then transmitted individually and can even follow different routes to its destination. Once all the packets arrive at the destination, they are recompiled into the original message. Packet switching is efficient and robust for data that can withstand some delays in transmission, such as e-mail messages and Web pages. However, modern applications like graphics animations, tele-conferencing, audio- and video streaming require connections with guaranteed available bandwidths and short delays. For the quality of such applications, it is essential that data is transmitted fast and arrives in successive order. Therefore, networks with guaranteed quality of service are becoming more and more important. In such networks, the connection requests specify their required bandwidths and the network allocates the requested bandwidth along a path from the sender to the receiver and reserves it as long as the connection is active. Because of obvious similarities with traditional circuit switching, this concept is called virtual circuit switching.

An efficient management of resources is of great importance in order to use such networks to their full extent. In this connection, we focus on two optimization problems that have to be addressed in the operation of networks with bandwidth reservations: call admission and wavelength routing. Call admission is the problem of deciding for a given set of connection requests which of them to accept and which to reject such that the accepted requests do not exceed the capacity of any link. The goal is to maximize an objective function. A natural one might be the number of accepted requests, for example. The wavelength routing problem asks for an assignment of paths and wavelengths to the connection requests such that any two paths that share an edge are assigned different wavelengths. The goal here is to minimize the number of wavelengths used. This is meaningful, for instance, in all-optical networks that support wavelength division multiplexing.

We consider several variants of these problems and are interested in their complexity. Which of them can be solved optimally in polynomial time, and which are $\mathcal{NP}$-hard [46] or even $\mathcal{APX}$-hard [11]? On the algorithmic side the results we obtain are either polynomial-time optimal algorithms or approximation algorithms (or on-line algorithms, respectively) with provable performance guarantees.

1.1 Results of this Thesis

As mentioned above the call admission control problem is an important optimization problem encountered in the operation of communication networks. In its general form the instances of the problem are specified by several parameters. Firstly, the underlying network is capacitated, i.e. every link is associated with a positive capacity. Secondly, each connection request (or call) is specified by two distinct nodes in the network that want to communicate, namely the sender and the receiver of the connection request. Each call has a certain bandwidth requirement and some time specification given by its starting time and its duration. If the network establishes a call, it first decides on a path from the sender to the receiver of that call, through which the call is being routed. Then it allocates the requested amount of bandwidth on all links along that path during the time period in which the call is active. Additionally, each call is associated with some profit, which is gained by the network provider only, if the desired connection is established. Call admission control is the problem of deciding which calls to accept and which to reject with the goal of maximizing the total profit accrued by the accepted calls. The decisions are made subject to the capacity constraints imposed by the network. A set of accepted calls is a feasible solution only, if the sum of bandwidth requirements of simultaneously active calls using the same link does not exceed the capacity of that link. In a feasible solution none of the capacity constraints is violated. In other words, call admission control is the problem of finding a feasible solution with maximum profit.

In general networks, the call admission control problem is not only $\mathcal{NP}$-hard, but also hard to approximate. It contains the so-called maximum edge-disjoint path problem as a special case, that cannot be approximated within $m^{1/2-\varepsilon}$ unless $\mathcal{P} = \mathcal{NP}$, where m denotes the number of links in the network [53]. Therefore the research on call admission control concentrated on special classes of network topologies. We investigate different variants of call admission control problems in chains, rings, and star networks.

A chain network consists of a set of nodes that are connected by links forming a simple path. For such networks we study the variant of call admission control without time specifications for the calls and unit bandwidth. That means that all calls are active at the same time and have a bandwidth requirement equal to 1. Further we assume that all calls have uniform profits, i.e. they are equally valuable. In this case maximizing the total profit of the accepted calls translates to maximizing the number of accepted calls.

It is well-known that a greedy algorithm solves the call admission control problem in the described scenario optimally. Suppose the chain is drawn horizontally in the plane and the nodes are numbered from left to right. The greedy algorithm processes the calls one by one in non-decreasing order of their right endnodes. The algorithm accepts the current call, if adding it to the solution computed so far does not violate any capacity constraints of the links. Otherwise, the current call is rejected.

The problem is closely related to the maximum k-colorable induced subgraph problem for interval graphs. The latter problem can be solved optimally in linear time by a clever implementation of the greedy algorithm provided that a sorted list of interval endpoints is given [22]. This immediately yields a linear-time algorithm for the call admission control problem in chains where all edges have the same capacity. While it is not difficult to adapt the approach to chains with arbitrary capacities incurring an increase in the running-time, we show how to generalize the algorithm of Carlisle and Lloyd to the case of arbitrary edge capacities by keeping the running-time linear in the number of calls.

In chain networks we ignored the issue of routing the requests because this issue does not exist, since there is a unique paths from the sender to the receiver of any call. However, in ring networks there exist exactly two possibilities to route a call from the sender node to the receiver node. A ring network is equivalent to a simple cycle. If it is drawn in the plane, the two possibilities to route a call are the clockwise and the counterclockwise direction. Since we are interested more in the call admission than in the routing issue, we investigate a variant of call admission control in rings, where the input specifies the calls as a set of paths in the ring. Hence, the route taken by a request is determined in the input and cannot be modified by the call admission control algorithm. Therefore, we denote the calls as paths.

The scenario we consider is the same as the one for chains that we described above. The calls have no time specifications and every call has unit profit and needs unit bandwidth on all the links contained in its path. For this call admission control variant we present a polynomial-time algorithm that solves the problem optimally. The algorithm is based on a decision procedure that checks whether a solution with at least k paths exists. Using binary search on the value of k we can find the maximum cardinality subset of paths that is feasible. The decision procedure is operating in rounds, where each round has to solve a call admission control problem in a chain network. We use the linear-time implementation for the call admission control problem in chains to obtain an efficient implementation for the call admission control variant in ring networks.

A star network consists of one center node and many outer (or leaf) nodes that are connected exclusively to the center node. We study the general variant of call admission control in star networks, where each call has a starting time and a duration, and is associated with a positive bandwidth requirement and a profit value that is gained if the call is accommodated. We present a constant-factor approximation algorithm for this call admission control variant in stars resolving an open problem posed by Erlebach [35]. Our method is randomized and achieves an approximation ratio of 18. We also show how our method can be derandomized (keeping the same approximation ratio). Further, we investigate a generalization of the problem by allowing call alternatives. That is, each call is allowed to specify several alternatives to increase its chances of being accepted. For example, a client wants a connection either from 8am to 11am, or in the afternoon from 2pm to 6pm. Such alternatives are allowed to differ in all call parameters. For this call admission control problem with alternatives we show that a modification of our algorithm always computes 24-approximations.

We also study the complexity of some very restricted variants. Suppose the star network has unit capacity on every link and the set of calls is such that each call has unit profit, needs one unit of bandwidth on its links, and has a fixed duration. If the number of possible starting times of the calls are at most two, the call admission control problem can be solved optimally in polynomial time by computing at most two maximum matchings in some dedicated graphs. Already for three different starting times, we prove that this simple looking problem is $\mathcal{APX}$-hard by a reduction from the bounded maximum 3-dimensional matching problem. We also give an alternative $\mathcal{NP}$-hardness proof of the kindred problem with arbitrary starting times and arbitrary durations by reducing the well-known 3-CNF satisfiability problem for Boolean formulas to it. A simple greedy algorithm is shown to approximate this problem within a factor of 2 from the optimal value. Further, we prove that the parameterized version of this problem is fixed-parameter tractable, if the number of rejected calls is taken as the parameter. For the variation with arbitrary capacities and arbitrary profits, a 3-approximation algorithm is given.

Another optimization problem encountered in the context of communication networks is that of wavelength routing and more specifically path coloring. Wavelength routing is the task of assigning paths and wavelengths to connection requests in all-optical networks with wavelength division multiplexing. If the requests are routed by some external mechanism or the route for each request is unique (as in tree networks), we call the problem of assigning wavelengths (or colors) to a set of fixed paths the *fixed path coloring* problem. The goal of this problem is to use as few colors as possible subject to the constraints that any two distinct paths must receive different colors if they share a link of the network. A natural generalization to the case of non-unit bandwidth in networks with bandwidth reservation is to interpret each wavelength as a channel of capacity 1. A set of paths can be assigned the same color (or wavelength) if for any link the sum of bandwidths of the paths containing that link is at most 1. The special case where every path has bandwidth 1 corresponds to the original path coloring problem. We consider the problem of online interval coloring in the case where the intervals have bandwidths in $(0, 1]$.

An interval can be viewed as a path in a chain network. In the on-line setting the intervals are presented to the on-line algorithm one by one, and the algorithm has to assign immediately each interval a color without any knowledge of the intervals arriving in the future. The coloring must be such that the total bandwidth of intersecting intervals with the same color must not exceed 1. We present an on-line algorithm for this problem that achieves a constant competitive ratio, where the constant is at most 195. Our algorithm is a combination of an optimal on-line algorithm for coloring interval graphs [65] and First-Fit coloring, for which we generalize the analysis of Kierstead [62] to the case of non-unit bandwidth.

1.2 Organization of the Thesis

In Chapter 2 we introduce the notation and basic concepts used throughout this thesis. We also give an overview of previous work that is related to the optimization problems studied in the subsequent chapters.

The Chapters 3 and 4 treat call admission control problems where the calls are given as paths that have no time specifications, require unit bandwidths, and have unit profit. In Chapter 3 we provide a linear-time implementation for this call admission control problem in chain networks with arbitrary capacities. We generalize an algorithm by Carlisle and Lloyd [22] for uniform capacities to the case of arbitrary capacities by keeping its running-time linear. In Chapter 4 we show that in ring networks this call admission control problem can be solved optimally in polynomial time. Our algorithm can be implemented efficiently using the result of Chapter 3 as a subroutine. A preliminary version of the results of Chapters 3 and 4 was presented at ICALP 2002 [1] as joint work with Christoph Ambühl, R. Sai Anand, and Thomas Erlebach.

In Chapter 5 we address various call admission control problems in star networks. It contains constant-factor approximation algorithms as well as hardness proofs. The approximation algorithms are based on the local ratio technique. Some of these results were obtained only recently in joint work with Thomas Erlebach, Dieter Mitsche, Ingo Schurr, Bettina Speckmann, and Emo Welzl [3].

Chapter 6 comprises a constant-competitive on-line algorithm for coloring intervals with bandwidth requirements in $(0, 1]$. It is to appear in the Proceedings of the 1st WAOA 2003 [2] as joint work with Thomas Erlebach.

Finally, Chapter 7 concludes this thesis with a summary of our results, some discussion and problems for future research.

Chapter 2

Preliminaries

2.1 Notation and Basic Concepts

In this section we present some basic definitions of mathematical notations and introduce the fundamental concepts used throughout this thesis. Readers already familiar with this material need only skim this section.

We start with the set of *natural numbers* $\mathbf{N} = \{0, 1, 2, \dots\}$. By $\mathbf{R}$ we denote the set of *real numbers*. A *set* is a collection of objects, called its *elements* or *members*. The *cardinality* or *size* of a set S is the number of elements in S, and is denoted by $|S|$.

By a (finite) *graph* we mean either a *simple graph* or a *multi-graph*. A graph is a pair $G = (V, E)$, where V is a finite set whose elements are called the *vertices* or *nodes* of G. The members of E are the *edges* of G. E is either a set of unordered pairs of distinct vertices, if G is a simple graph, or a family of unordered pairs of not necessarily distinct vertices, if G is a multi-graph. In the latter case an edge is allowed to appear several times in which case it is called a *multiple* edge. If $e = \{u, v\}$ is an edge in E, then u and v are said to be *adjacent* and they are called the *endpoints* of the edge e. We also say that the edge e is *incident* to its vertices u and v. A *weighted graph* is a graph $G = (V, E)$ together with a function $w : E \to \mathbf{R}$ which associates a *weight* with any edge. The *size* of a graph is defined to be $|V| + |E|$.

Given a graph G and two vertices v_0 and v_k, a *path* p from v_0 to v_k is a sequence $v_0 v_1 \dots v_k$ of distinct vertices (with the possible exception $v_0 = v_k$) such that $\{v_i, v_{i+1}\}$ belongs to E for all $i = 0, \dots, k-1$. Mostly, we represent a path by the set of edges it uses, i.e., $p = \{\{v_i, v_{i+1}\} \mid i = 0, \ldots, k-1\}$. The *length* of a path is the number of edges it uses. A path from v_0 to v_k with $v_0 = v_k$ is called a *closed path* or a *cycle*. A graph is *connected* if, for every two distinct vertices u and v, there is a path from u to v.

A graph is a *chain*, if it consists of a single path. It is a *ring*, if it consists of a single cycle. A *tree* is a connected graph with no cycles. A *star* is a graph with a center vertex and a set of outer (or leaf) vertices, each of which is adjacent exclusively to the center vertex. Note that chains and stars are special cases of trees.

Networks are commonly modeled by simple graphs. The vertices represent the nodes of the network, and there is an edge between two vertices if and only if there is a link between the corresponding nodes in the network. Connection requests (or calls) are represented by paths in the graph corresponding to the underlying network. In this thesis, we do not address the issue of routing the connections through the networks. The routes for the connection requests are either unique (like in tree networks) or assumed to be pre-specified (by some routing protocol or the input) in the case of ring networks.

For a graph $G = (V, E)$ a subset $M \subseteq E$ is called a *matching*, if no two edges in M share an endpoint. A *maximum matching* is a matching of maximum cardinality. Maximum matchings can be computed in $O(\sqrt{|V|} \cdot |E|)$ time in general graphs [33, 72]. A subset $I \subseteq V$ is called an *independent set* (in G), if no two vertices in I are connected by an edge in E. Further, a subset $C \subseteq V$ is called a vertex cover (of G), if each edge in E is incident to at least one vertex in C. For a given graph, the vertex cover problem is to find a vertex cover of minimum size. Its corresponding decision problem is $\mathcal{NP}$-complete [61], and its parameterized version is fixed-parameter tractable [31].

All problems considered in this thesis are *optimization problems*, which means that some value must be minimized or maximized. A problem P is called *polynomial-time solvable*, if there exists an algorithm $\mathcal{A}$ and a constant k such that $\mathcal{A}$ solves all instances of P in time $O(n^k)$, where n is the length of the (binary) encoded instance. We say that the problem P belongs to the complexity class $\mathcal{P}$ in that case. An optimization problem can typically be recast as decision problem by imposing a bound on the value to be optimized. This allows us to apply the theory of $\mathcal{NP}$-completeness to optimization problems. We call an optimization problem *$\mathcal{NP}$-hard*, if its corresponding decision problem is $\mathcal{NP}$-complete. A decision problem is called *$\mathcal{NP}$-complete*, if it is in $\mathcal{NP}$ and there is a polynomial-time reduction from every other problem in $\mathcal{NP}$ to the problem under consideration. For all optimization problems addressed in this thesis their corresponding decision problems are actually in $\mathcal{NP}$. Thus, proving $\mathcal{NP}$-hardness boils down to reducing a known $\mathcal{NP}$-complete problem to the decision problem of the optimization problem at hand. For a guide to $\mathcal{NP}$-completeness we refer the reader to the wonderful book by Garey and Johnson [46]. It discusses the theory of $\mathcal{NP}$-completeness and provides a catalog of many problems that were known to be $\mathcal{NP}$-complete at that time.

In parameterized complexity, people are interested in fixed-parameter tractable algorithms for $\mathcal{NP}$-hard problems. They identify a problem parameter k that captures a part of the input or output of the problem. An algorithm is called *fixed-parameter tractable* (FPT), if it runs in time $O(f(k) \cdot poly(n))$ where $f(k)$ is an arbitrary function of the parameter k and $poly(n)$ is a polynomial of the input size n. For a substantial reference to parameterized complexity we recommend the book of Downey and Fellows [31]. They also introduce the complexity class W[2] that is mentioned in the related work section of this thesis.

Since many of the optimization problems encountered in this thesis are $\mathcal{NP}$-hard, we are interested in approximation algorithms that compute near-optimal solutions. An algorithm is called a *ρ-approximation algorithm*, if it runs in polynomial time and always outputs a solution with a value that is at least a $1/\rho$ fraction of the optimal value for maximization problems and at most ρ times the optimal value for minimization problems. The value ρ is then called the *approximation ratio* of the algorithm. The smaller ρ is, the better is the approximation. An optimization problem P belongs to the complexity class $\mathcal{APX}$, if there exists a ρ-approximation algorithm for some constant $\rho > 1$. It is called $\mathcal{APX}$-hard, if there is an approximation preserving reduction (AP-reduction, for short) from any other problem in $\mathcal{APX}$ to the problem P. The detailed definition of an AP-reduction is given in Chapter 5 where we actually prove $\mathcal{APX}$-hardness for a call admission control problem. For an extensive introduction to optimization problems and their approximability properties we direct the reader to the excellent book by Ausiello et al. [11]. It also contains a compendium of $\mathcal{NP}$-hard optimization problems.

So far, we dealt only with off-line algorithms that have complete knowledge of the entire input. However, this assumption is often unrealistic in practical applications. In many algorithmic problems the input is only partially available because it arrives dynamically over time. Algorithms for this scenario are called *on-line algorithms*. Whenever a part of the input arrives, they must generate an output immediately without any knowledge of the future input. The performance of on-line algorithms is typically analyzed by comparing their output to the output of an optimal off-line algorithm that knows the entire input in advance [80]. An on-line algorithm $\mathcal{A}$ is said to be *ρ-competitive*, if for a minimization problem $\mathcal{A}(I) \leq \rho \cdot \mathit{OPT}(I)$ (and, likewise, for a maximization problem $\mathcal{A}(I) \geq 1/\rho \cdot \mathit{OPT}(I)$) holds true for any input sequence I, where $\mathit{OPT}(I)$ measures the output of an optimal off-line algorithm. The factor ρ is called the *competitive ratio* of $\mathcal{A}$. A survey about on-line algorithms and competitive analysis can be found in [6, 7]. Further reading on the topic of on-line algorithms is provided in the book by Borodin and El-Yaniv [20].

2.2 Problem Definitions

In this section we formally define the two optimization problems call admission control and path coloring. For the latter one we will concentrate on the problem of coloring intervals in an on-line scenario where the intervals have bandwidth requirements in $(0, 1]$.

2.2.1 Call Admission Control Problems

There is a huge variety of call admission control problems, to which we refer to as CAC problems for short. To distinguish the numerous CAC variants we are now going to fix a notation for various parameters, in which they may differ.

The network is modeled by a capacitated graph $G = (V, E, c)$. The vertex set $V = \{v_1, \ldots, v_n\}$ represents the n nodes of the network, and the edge set $E = \{e_1, \ldots, e_m\}$ reflects the links of the network. There is an edge $\{v_i, v_j\}$ in the edge set E, if and only if the nodes corresponding to the vertices v_i and v_j are connected by a link in the network. The number of edges (or links) is denoted by $m = |E|$. The capacities of the links are given as a *capacity function* $c : E \to \mathbf{N}$ that maps each edge $e \in E$ to the *positive capacity* $c(e)$ of the link the edge e is representing. The values c_{min} and c_{max} constitute the minimum and maximum capacity, respectively, of the edges, i.e. $c_{min} = \min\{c(e) : e \in E\}$ and $c_{max} = \max\{c(e) : e \in E\}$.

The set of *connection requests* or *calls* is denoted by $\mathcal{R} = \{r_1, \ldots, r_\ell\}$. Each call $r_i = (s_i, u_i, b_i, t_i, d_i, w_i) \in \mathcal{R}$ is a tuple that specifies a connection request between the *source vertex* $s_i \in V$ and the *destination vertex* $u_i \in V$. The request r_i has a positive *bandwidth requirement* $b_i \in \mathbf{R}^+$ and is associated with a *starting time* $t_i \in \mathbf{N}$, a *duration* $d_i \in \mathbf{N}$, and a *profit* or *weight* $w_i \in \mathbf{R}$. We say that a call r_i is *active* from time t_i to time $t_i + d_i$. Let T denote the latest point in time at which some call is active, i.e. $T := \max\{t_i + d_i : r_i \in \mathcal{R}\}$. Further, let $b_{min} = \min\{b_i : i = 1, \ldots, \ell\}$ and

$b_{max} = \max\{b_i : i = 1, \ldots, \ell\}$ be the minimum and maximum bandwidth requirements of the calls in $\mathcal{R}$.

A *solution* consists of a set $Q \subseteq R$ of accepted calls and an assignment of paths $path(i)$ from s_i to u_i for every accepted call $r_i \in Q$. A solution Q is called *feasible*, if the sum of bandwidth requirements of simultaneously active calls in Q using the same edge does not exceed the capacity of that edge, i.e.

$$\sum_{r_i \in Q(e,t)} b_i \leq c(e) \qquad \forall e \in E \text{ and } \forall t = 0, 1, \ldots, T,$$

where $Q(e, t) = \{r_i \in Q : t_i \leq t < t_i + d_i \text{ and } e \in path(i)\}$ is the set of accepted calls that use the edge e at time t.

The *goal* of a CAC problem is to compute a feasible solution Q that maximizes the sum

$$\sum_{r_i \in Q} w_i$$

of the profits of the accepted calls.

In this thesis we are not concerned with the issue of finding a "good" routing for the calls, but rather with the decision which call to accept and which to reject. In the network topologies that we consider the routing of the calls is either unique like in tree networks (which include chain and star networks) or it is pre-determined by the input in the CAC problem in rings that we are going to investigate. Therefore we refer to the calls as *paths* in these cases.

A generalization of the CAC problem defined above is to allow *call alternatives*. In this framework each call is allowed to specify several alternatives to increase its chances of being accepted. For example, a client wants a connection either from 8am to 11am, or in the afternoon from 2pm to 6pm. Both alternatives belong to the same connection request and are termed *call alternatives*. Such alternatives are allowed to differ in all call parameters. The network can either accept one of the alternatives of a call, or reject the call completely. Thus, a solution to a CAC problem with call alternatives is given by the set of accepted call alternatives. A solution is feasible, if at most one alternative per call is accepted and the sum of bandwidths of simultaneously active call alternatives in the solution using the same edge does not exceed the capacity of that edge. The objective function stays the same as before, i.e. maximizing the sum of profits of the accepted call alternatives.

In this thesis we consider CAC problems in off-line scenarios. It should be noted that problems related to CAC are often encountered in an on-line setting, where the requests are presented to the algorithm one by one and the algorithm must accept or reject each request without knowledge of future requests.

However, we think that it is meaningful to study the off-line version as well for several reasons. First, an off-line CAC algorithm is needed in the *network design* phase, when a candidate topology with link capacities is considered and one wants to know how many of the forecasted traffic requirements can be satisfied by the network. Second, the on-line model ignores the possibility that resources might be requested ahead of the time when they are needed. A natural concept to incorporate this possibility is *advance reservation* [52, 71], which is attractive for both the network provider and the users. The network provider has the advantage of a higher flexibility and can use better algorithms to maximize the profit gained from the accepted calls. The users benefit from a guaranteed quality of service (QoS) once their calls are accepted. Quality of service is desirable for example in the context of video-conferencing. Finally, an optimal off-line CAC algorithm is helpful as a benchmark for the evaluation of other off-line or on-line call admission control strategies.

In order to have short descriptions for the huge amount of different CAC variants we introduce a two-field classification $<\alpha \,|\, \beta>$ that is similar to the $\alpha \mid \beta \mid \gamma$ notation for sequencing and scheduling problems that was introduced by Graham et al. [51] and is described in an article by Lawler et al. [69].

It works as follows. The first field $\alpha \subseteq \{\alpha_1, \alpha_2, \alpha_3\}$ specifies the network environment. Let $\circ$ denote the empty symbol. The parameter $\alpha_1 \in \{*$, Chain, Ring, Tree, Star, ToR, $\ldots\}$ specifies the network topology. The possible values are characterized as follows:

- $\alpha_1 = *$: the underlying network is general. That means, it is represented by an arbitrary simple graph;
- $\alpha_1 =$ Chain: the network topology is a chain consisting of a single path;
- $\alpha_1 =$ Ring: the network has ring topology, i.e. it is a single cycle;
- $\alpha_1 =$ Tree: the network is a tree modeled as a connected, acyclic graph;
- $\alpha_1 =$ Star: the network has the topology of a star. It has a unique center node and leaf nodes that are connected exclusively to that center node;
- $\alpha_1 =$ ToR: the network is a tree of rings, i.e. a collection of rings that are interconnected in a tree structure such that any two rings share at most one node.

Sometime we also use the notation $\alpha_1 = *$ for some special graphs (e.g. series parallel graphs). In these cases, the usage will be clear from the context.

If $\alpha_1 \in \{*, \text{Ring}, \text{ToR}\}$, the routes for the calls are not unique. In these instances, the second parameter α_2 deals with the routing policy:

- $\alpha_2 = \circ$: the CAC algorithm can decide on a route for each accepted call;
- $\alpha_2 = \text{det}$: the routes for the calls are pre-determined by the input or some external mechanism. The set of requests is given as a set of pre-determined paths.

If $\alpha_1 \in \{\text{Chain}, \text{Tree}, \text{Star}\}$, there is a unique path between every pair of nodes in the network. Hence, the path is uniquely determined by the source and destination node of a request, and a CAC algorithm has no choice for routing the connections. The parameter α_2 is superfluous in this case.

The third parameter α_3 defines the capacity function on the links of the network. We assume the capacities of the links to be positive integer values. The two possible distinctions are

- $\alpha_3 = \circ$: the edge capacities can assume arbitrary positive integer values;
- $\alpha_3 = c_i = k$: all edges e_i in the network have uniform edge capacities equal to the value of $k \in \mathbf{N}$.

The second field $\beta \subseteq \{\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6\}$ indicates a number of call characteristics. The value of β_1 stands for the bandwidth requirements of the calls:

- $\beta_1 = \circ$: the calls can have arbitrary bandwidth requirements;
- $\beta_1 = b_i = 1$: every call $r_i \in \mathcal{R}$ needs unit bandwidth on the edges along some path from the sender to the receiver node of that call.

The call parameters β_2 and β_3 specify the timing of calls.

- $\beta_2 = t_i$: the calls can have arbitrary starting times;
- $\beta_2 = \circ$: the starting times $t_i = 0$ for all calls $r_i \in \mathcal{R}$. This case models the CAC problems where the calls have no time specifications.

If $\beta_2 = t_i$, we have a further distinction according to the durations of the calls that is captured in a parameter β_3 with the following values:

- $\beta_3 = \circ$: the calls can have different durations;

- $\beta_3 = d_i = k$: every call $r_i \in \mathcal{R}$ has a uniform duration equal to the value of k.

The parameter β_4 is concerned with the profits of the calls. The two possibilities are:

- $\beta_4 = \circ$: the calls can have arbitrary profits;
- $\beta_4 = w_i = 1$: every call $r_i \in \mathcal{R}$ has unit profit.

The parameters β_5 and β_6 specify some additional characteristics of the CAC problems. β_5 is important for the on-line versions of the problems at hand:

- $\beta_5 =$ pmtn: preemption is allowed, i.e. the algorithm is allowed to preempt a call it has accepted at an earlier stage of its execution;
- $\beta_5 = \circ$: no preemption is allowed.

Finally, the parameter β_6 deals with call alternatives:

- $\beta_6 =$ alt: call alternatives are allowed;
- $\beta_6 = \circ$: no alternatives are allowed for the calls.

Occasionally, this field will contain additional characteristics such as $b_{max} \leq c_{min}$ or $t_i \in \{0, 1, 2\}$. The interpretation of these should be obvious.

For sequencing and scheduling problems Lawler et al. [69] use a third field γ in their notation that specifies the possible objective functions for this kind of problems. Since in the case of CAC problems the objective function is always the same, we omit the third field in the classification of the various CAC problems. In any case the objective function we consider is to maximize the sum of the profits of the accepted calls. If the call parameter $\beta_4 = w_i = 1$, all calls have unit profit, and the objective function to maximize the sum of profits of the accepted calls translates to maximizing the number of accepted requests.

For example <Ring, det $|\, b_i = 1, w_i = 1$> is the CAC problem of maximizing the number of accepted pre-determined paths of unit bandwidth in ring networks with arbitrary capacities. This variant can be solved optimally in polynomial time as shown in Chapter 4.

Another example is the CAC problem <Star $|\, t_i$> in star networks with arbitrary capacities. The calls have starting times and durations, a positive bandwidth requirement, and arbitrary profit values. The goal is to maximize the sum of profits of the accepted calls. This variant is $\mathcal{APX}$-hard, but admits an 18-approximation algorithm. These results are presented both in Chapter 5.

2.2.2 Path Coloring Problems

The second optimization problem that we consider is called the *path coloring* problem. It models the wavelength allocation problem in all-optical networks with wavelength division multiplexing.

In the path coloring problem, we are given a network modeled as a graph $G = (V, E)$ together with a set $\mathcal{R} = \{r_1, \ldots, r_\ell\}$ of *connection requests*. Each request $r_i = (s_i, u_i, b_i) \in \mathcal{R}$ is specified by the *source vertex* $s_i \in V$, the *destination vertex* $u_i \in V$, and the *bandwidth requirement* $b_i \in \mathbf{R}$, which in the conventional path coloring problem equals 1 and is therefore omitted. In the (conventional) path coloring problem, the task is to assign paths and colors to the requests such that paths that share an edge are assigned different colors. The *goal* is to use as few colors as possible.

The assignment of paths to the requests (also called *routing*) has to be computed by the path coloring algorithm. If the paths are pre-determined by the input or are unique as in tree networks, the problem is termed *fixed path coloring* problem.

The problem we are interested in, is a generalization of the on-line version of fixed path coloring in chain networks. It is called the *on-line interval coloring problem with bandwidths in* $(0, 1]$. As paths in chain networks can be viewed as intervals over the real numbers, the input of this problem is a collection $\mathcal{I} = \{i_1, \ldots, i_\ell\}$ of intervals over the real numbers $\mathbf{R}$. Each interval i_t is associated with a bandwidth requirement $b(i_t)$ with $0 < b(i_t) \leq 1$ for $t = 1, \ldots, \ell$. In an on-line scenario these intervals are presented one by one to an on-line algorithm in some externally determined sequence. Whenever an interval $i_t \in \mathcal{I}$ arrives, the algorithm knows the previously processed intervals and the colors assigned to them, but it knows nothing about the unprocessed intervals arriving in the future. Based on this knowledge the algorithm must irrevocably assign a color $f(i_t)$ to the interval i_t, in such a way that for every color x and every point $r \in \mathbf{R}$ the sum of the bandwidths of the intervals containing r that are assigned color x is at most 1.

The special case where every interval has bandwidth 1 corresponds to the original on-line interval coloring problem. Our problem is thus a simultaneous generalization of on-line interval coloring and on-line bin-packing.

2.3 Related Work

In this section we give an overview of results that are related to our work. We group the extensive literature into sections according to the different network optimization problems addressed in these articles. We start with CAC problems.

2.3.1 Call Admission Control

Many authors have investigated CAC problems for various network topologies in the off-line and on-line setting. For survey articles on on-line algorithms for CAC problems we refer the reader to [20, Chapter 13] and [70, 76] and mention only some of the known results in this section.

In the off-line case, some CAC variants were studied intensively. One prominent example is the problem $<\ast, c_i = 1 \mid b_i = 1, w_i = 1>$, which is called the *maximum edge-disjoint paths* problem (MEDP). A survey on this problem can be found in [67]. The MEDP problem is polynomial for chains, rings, and undirected trees. Constant-factor approximation algorithms have been presented for bidirected trees [39], for trees of rings [34], and for complete graphs [41, 23]. For general directed graphs with m edges, the MEDP problem cannot be approximated within $m^{1/2-\varepsilon}$ for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$ [53]. An approximation algorithm with approximation ratio $O(\sqrt{m})$ is known even for the more general *unsplittable flow* problem $<\ast \mid b_{max} \leq c_{min}>$ provided that the maximum bandwidth requirement b_{max} is at most the minimum edge capacity c_{min} [12].

Equivalently to maximizing the number of accepted calls, some authors studied CAC problems with the goal of minimizing the number of rejected requests. This is meaningful to consider since the number of rejected calls is expected to be small in practice. Blum et al. [18] studied preemptive on-line CAC variants with this objective function. They obtain competitive ratios of 2 for $<\text{Chain} \mid b_i = 1, w_i = 1, \text{pmtn}>$ and $<\ast, c_i = 1 \mid b_i = 1, w_i = 1, \text{pmtn}>$, and of $O(\sqrt{m})$ for $<\ast \mid b_i = 1, w_i = 1, \text{pmtn}>$ where m is the number of edges in the network. For the off-line version of $<\ast \mid b_i = 1, w_i = 1>$, they give an $O(\log m)$-approximation algorithm.

Anand et al. investigated the parameterized complexity of CAC problems with the number of rejected calls as parameter. For an introduction to parameterized complexity we refer the reader to [31]. Anand et al. [10] showed that $<\ast, \text{det}, c_i \leq C \mid b_i = 1, w_i = 1>$ is fixed-parameter tractable (FPT), if C is a fixed constant. Allowing arbitrary edge capacities, $<\ast, \text{det} \mid b_i = 1, w_i = 1>$

is W[2]-hard even for series-parallel graphs, which are graphs with treewidth at most two [19]. For series-parallel graphs no FPT algorithm can exist for $<\! * \mid b_i = 1, w_i = 1 \!>$ unless $\mathcal{P} = \mathcal{NP}$. They also present FPT algorithms for both $<\!\text{Tree} \mid b_i = 1, w_i = 1\!>$ and $<\!\text{ToR}, c_i = 1 \mid b_i = 1, w_i = 1\!>$. Their FPT algorithms apply the method of bounded search trees [31].

Chain Networks

For chain networks, the off-line version of CAC is closely related to the problem of finding a maximum k-colorable induced subgraph for interval graphs. Here the calls are interpreted as intervals on the real line, and interval graphs are graphs that capture the intersection relation for some set of intervals on the real line. The maximum k-colorable induced subgraph problem for interval graphs can be solved optimally in polynomial time [85]. Carlisle and Lloyd presented an algorithm running in linear time as a result of a clever implementation of a greedy algorithm provided that a sorted list of interval endpoints is given [22]. This immediately gives a linear-time algorithm for $<\!\text{Chain}, c_i = k \mid b_i = 1, w_i = 1\!>$. It is not difficult to adapt their approach to chain networks with arbitrary capacities incurring an increase in running-time. They also propose an efficient algorithm for the problem of computing a maximum-weight k-colorable subset of a given set of weighted intervals on the real line. This algorithm is based on solving a min-cost flow problem and immediately gives a polynomial-time optimal algorithm for the CAC problem $<\!\text{Chain}, c_i = k \mid b_i = 1\!>$.

The on-line version of $<\!\text{Chain}, c_i = 1 \mid b_i = 1, w_i = 1, \text{pmtn}\!>$ was studied in [45]. They give a deterministic algorithm that achieves a competitive ratio of $O(\log n)$ for a chain with n nodes, and show that this bound is optimal for deterministic on-line algorithms. For $<\!\text{Chain}, c_i = k \mid w_i = 1, \text{pmtn}\!>$, Adler and Azar gave a randomized on-line algorithm with constant competitive ratio [4]. Their approach can be adapted to ring networks with equal capacities.

For arbitrary profits and arbitrary bandwidth requirements, Bar-Noy et al. [13] presented a 5-approximation algorithm for $<\!\text{Chain}, c_i = 1 \mid \text{alt}\!>$ and a 3-approximation algorithm for $<\!\text{Chain}, c_i = 1 \mid \,\!>$. The factor 3 for the latter problem was improved to $2 + \varepsilon$ by Calinescu et al. [21].

However, for the variant $<\!\text{Chain} \mid \,\!>$ without any restrictions on the bandwidth requirements of the calls, no constant-factor approximation algorithm is known so far. Constant-factor approximations for this problem are known only under the "no-bottleneck" assumption that the maximum bandwidth requirement is not larger than the minimum edge capacity, i.e. for the problem

<Chain | $b_{max} \leq c_{min}$> [24, 25].

If the calls have time specifications, the problem <Chain, $c_i = 1 \mid b_i = 1, t_i$> is equivalent to the *maximum weight independent set* problem (MWIS) in the intersection graph of axis-parallel rectangles. Each rectangle corresponds to a call in the following way. Its projection on the x-axis represents the path from the source node to the destination node, and its projection on the y-axis represents the time interval of the call. Motivated by map labeling problems, Agarwal, van Kreveld and Suri [5] gave an approximation algorithm for the MWIS problem for axis-parallel rectangles that achieves an approximation ratio of $O(\log n)$. This factor was improved to $\log n/\alpha$ for any constant α by Berman et al. [17]. Lewin-Eytan, Naor and Orda [71] gave a $4c$-approximation algorithm for this problem, where c denotes the maximum number of rectangles that simultaneously cover a point in the plane.

Ring Networks

For network topologies containing cycles, an important distinction for CAC problems is whether the calls are given as paths specified as part of the input or can be determined by the CAC algorithm. In the latter case, only the endpoints of each call are specified in the input.

For the CAC problem <Ring | $b_i = 1$, $w_i = 1$>, Anand and Erlebach presented a polynomial time algorithm that always computes a feasible solution with at least *OPT* $- 3$ accepted calls, where *OPT* denotes the number of calls in the optimal solution [9]. To the best of our knowledge, it is not known whether this problem in rings is $\mathcal{NP}$-hard.

If the paths are pre-determined in ring networks, they can be viewed as arcs on a circle. Therefore path problems in rings are closely related to *circular arc graphs*. A graph is a circular-arc graph if its vertices can be represented by arcs on a circle such that two vertices are joined by an edge if and only if the corresponding arcs intersect [49]. Circular-arc graphs can be recognized efficiently [42]. For a given circular-arc graph, a maximum clique or a maximum independent set can be computed in polynomial time [49], whereas coloring a circular-arc graph with the minimum number of colors is $\mathcal{NP}$-hard [47]. A coloring with at most $1.5\,\omega$ colors always exists and can be computed efficiently [60], where ω is the size of a maximum clique in the graph. The problem <Rings, det, $c_i = 1 \mid b_i = 1$> is equivalent to the maximum weight independent set problem in circular-arc graphs.

For many call admission control problems in ring networks it is possible to obtain 2-approximation algorithms using a general approach called the edge-removal technique. It works as follows. Suppose one has an optimal algorithm *ALG* for some CAC problem in chains. The following procedure yields 2-approximations for the same CAC problem in rings. It constructs two feasible solutions for the ring problem and outputs the one with larger profit. Let e denote the edge in the ring with the smallest capacity, i.e. $c(e) = c_{min}$. If the paths are pre-determined, let $\mathcal{R}_e$ be the subset of paths that contain the edge e, and let $\mathcal{R}_{\bar{e}}$ be the subset of paths avoiding the edge e. The first solution is then the output of the optimal algorithm *ALG* on input $\mathcal{R}_{\bar{e}}$, which lives in a chain because the edge e can be omitted. The second solution is obtained by taking the $c(e)$ many paths from $\mathcal{R}_e$ with the largest profit. This solution does not violate any edge capacity because $c(e)$ was chosen to be the minimum edge capacity among all edges in the ring. A simple calculation shows that the larger profit of these two feasible solutions is indeed a 2-approximation.

If the calls are specified only by their endnodes and the CAC algorithm can decide on their routes, the method works likewise. The only difference is in the definition of the sets $\mathcal{R}_e$ and $\mathcal{R}_{\bar{e}}$. This time, every call has two possible routes in the ring. One contains the edge e, the other does not. Hence, $\mathcal{R}_e$ denotes this time the set of paths obtained by routing all the calls in $\mathcal{R}$ in such direction that they contain the edge e. Similarly, $\mathcal{R}_{\bar{e}}$ consists of the paths of all calls routed in the way that they avoid the edge e. Again, solving the problem in chains on input $\mathcal{R}_{\bar{e}}$ and taking the $c(e)$ most valuable paths in $\mathcal{R}_e$ yields two feasible solutions. The solution with larger profit among these two is a 2-approximation for the CAC problem in rings.

Tree and Star Networks

In this section, we first report on previous work concerning CAC problems in general tree networks, and then give an overview of some results for the special case of star networks.

The CAC problems in trees with unit edge capacities received considerable attention. Lewin-Eytan, Naor and Orda [71] gave a 5-approximation algorithm for <Tree, $c_i = 1 \mid$ >. For the more general <Tree, $c_i = 1 \mid t_i$> problem they achieve an approximation ratio of $5 \log n$, where n is the number of nodes in the tree.

Erlebach [35] considered CAC problems with call alternatives in different network topologies. For <Tree, $c_i = 1 \mid$ alt> he presented an approximation algorithm that achieves an approximation ratio of 10, which can be improved to 8 by a careful analysis. For the same problem <ToR, $c_i = 1 \mid$ alt> in trees of rings, he gives a 18-approximation algorithm. Further, he presents approximation algorithms for several CAC variants with call alternatives in star networks. He gives a 5-approximation for <Star, $c_i = 1 \mid$ alt>, and 10-approximation algorithms for <Star | alt> and <Star, $c_i = 1 \mid t_i$, alt>. The most general variant of CAC in stars, namely <Star | t_i, alt> is shown to be approximable within $O(\log C)$, where C is the ratio of the maximum edge capacity c_{max} to the minimum edge capacity c_{min}. All these algorithms are obtained by first solving a linear programming relaxation and then decomposing the optimal fractional solution into a convex combination of integral solutions. The output of the algorithm is the best of the integral solutions obtained this way.

Overview

The Table 2.1 gives an extensive overview of known approximation ratios for off-line call admission control problems. To avoid many repetitions, in the column for the parameter α_1 the network topology is stated only once between two horizontal lines, but is assumed to be the same in all consecutive rows up to the next entry. In the column for the parameter α_2 the entry "(det)" expresses the fact that the paths for the connection requests are pre-determined by the nature of the respective network topology, i.e., the paths are unique in trees and their special cases. This entry is also not repeated. In all other columns an empty entry has the meaning of the $\circ$ symbol as defined in Section 2.2.1.

2.3.2 Path Coloring

Another optimization problem in communication networks that has been studied intensively is called the *path coloring* problem. It models the wavelength allocation problem in all-optical networks with wavelength division multiplexing. In path coloring problems, the bandwidth requirement of each path is typically assumed to be 1, implying that no two intersecting paths can get the same color. For fixed paths, Chlamtac et al. [28] proved that the path coloring problem in arbitrary networks is $\mathcal{NP}$-hard, but can be solved in polynomial time for in-trees and out-trees, i.e. for trees, in which all edges are directed towards the root or towards the leaves, respectively. They showed that a set of paths with maximum load L in such graphs can be colored using L colors.

A considerable amount of work on path coloring studies scenarios where the set of connection requests is of a special form. These include *one-to-all* path coloring (also called broadcast), *all-to-all* path coloring (also known as gossiping), path coloring for *k-relations* [77], and *permutation* path coloring.

α_1	α_2	α_3	β_1	β_2	β_3	β_4	β_5	β_6	Ratio	Comment
Chain	(det)	$c_i = 1$	$b_i = 1$			$w_i = 1$			1	MEDP
		$c_i = k$	$b_i = 1$			$w_i = 1$			1	Greedy algorithm in linear time [22]
			$b_i = 1$			$w_i = 1$			1	Greedy algorithm in linear time (see Chapter 3)
		$c_i = k$	$b_i = 1$						1	Min-cost network flow problem [22]
			$b_i = 1$						1	Also by min-cost network flow
		$c_i = 1$							$2 + \varepsilon$	For any $\varepsilon > 0$ [21]
		$c_i = 1$						alt	5	Based on the local ratio technique [13]
			$b_{max} \leq c_{min}$						$2 + \varepsilon$	For $\varepsilon > 0$ under "no-bottleneck" assumption [25]
		$c_i = 1$	$b_i = 1$	t_i					$O(\log \ell)$	MWIS of ℓ axis-parallel rectangles [5]
									$4 \cdot c$	$c =$ max. clique number of intersection graph [71]
Star	(det)	$c_i = 1$	$b_i = 1$			$w_i = 1$			1	By computing maximum matching(s)
		$c_i = 1$	$b_i = 1$	$t_i \in \{0, 1\}$	$d_i = 2$	$w_i = 1$			1	
		$c_i = 1$	$b_i = 1$	$t_i \in \{0, 1, 2\}$	$d_i = 2$	$w_i = 1$			2	$\mathcal{APX}$-complete (see Section 5.4)
		$c_i = 1$	$b_i = 1$	t_i		$w_i = 1$			2	Greedy algorithm (see Section 5.5.1)
			$b_i = 1$	t_i					3	With local ratio technique (see Section 5.5.2)
		$c_i = 1$						alt	5	LP + rounding and partitioning [35],
		$c_i = 1$		t_i				alt	8	same ratios possible with local ratio technique
								alt	10	
				t_i					18	See Section 5.3
				t_i				alt	24	
Tree (undir.)	(det)	$c_i = 1$	$b_i = 1$						1	MEDP [82]
		$c_i = 1$							5	Based on the local ratio technique [71]
		$c_i = 1$						alt	8	LP + rounding and partitioning [35]
		$c_i = 1$		t_i					$5 \log \ell$	Binary partition of ℓ requests [71]
			$b_{max} \leq c_{min}$						48	Via integrality gap of natural LP formulation [25]
Tree (bidir.)	(det)	$c_i = 1$	$b_i = 1$			$w_i = 1$			$5/3 + \varepsilon$	For every fixed $\varepsilon > 0$. MEDP [39]
Ring	det		$b_i = 1$			$w_i = 1$			1	See Chapter 4 or [1]
	det		$b_i = 1$						2	Edge-removal technique
	$\circ$	$c_i = 1$	$b_i = 1$			$w_i = 1$			1	MEDP
	$\circ$		$b_i = 1$			$w_i = 1$			$OPT - 3$	$\rightarrow$ PTAS [9]
	$\circ$		$b_i = 1$						2	Edge-removal technique
Tree of Rings	$\circ$	$c_i = 1$	$b_i = 1$						3	MEDP [34]
	det	$c_i = 1$	$b_i = 1$						4	MEDP [34]
	$\circ$	$c_i = 1$						alt	18	LP + rounding and partitioning [35]
K_n	$\circ$	$c_i = 1$	$b_i = 1$			$w_i = 1$			9	MEDP [23]
	$\circ$	$c_i = 1$	$b_i = 1$						27	MEDP [41]
$*$	$\circ$	$c_i = 1$	$b_i = 1$			$w_i = 1$			$O(\sqrt{m})$	MEDP on graph with m edges [67]

Table 2.1: *Overview of known approximation ratios for CAC problems*

Survey articles containing the results regarding these problems can be found in [16, 66].

Tree Networks

For chain networks the path coloring problem can be solved optimally in polynomial time. The paths in chain networks can be viewed as intervals. Their intersection graph is an interval graph that can be colored optimally in polynomial time [48].

Similar to interval coloring in chains, the path coloring problem in undirected trees can be interpreted as coloring the conflict graph of paths in trees. However, Golumbic and Jamison [50] proved $\mathcal{NP}$-hardness for coloring the edge intersection graphs of paths in trees, which implies that path coloring in undirected trees is $\mathcal{NP}$-hard as well. Tarjan [82] gave a $3/2$-approximation algorithm for coloring these intersection graphs. Using the edge-coloring algorithm by Nishizeki and Kashiwagi [75], Erlebach and Jansen [38] showed how to obtain an approximation algorithm for path coloring in undirected trees that achieves an asymptotic approximation ratio of $11/10$ and an absolute approximation ratio of $4/3$.

Mihail, Kaklamanis, and Rao [73] gave the first approximation algorithm for path coloring in bidirected trees. Their algorithm used at most $(15/8)L$ colors for coloring a set of paths with maximum load L, which was improved in a sequence of papers. An improvement to $(7/4)L$ was independently discovered by Kaklamanis and Persiano [57] and by Kumar and Schwabe [68]. Erlebach et al. [40, 58] further improved this bound to $\lceil (5/3)L \rceil$. By showing a lower bound of $\lfloor (5/3)L \rfloor$ for every local greedy algorithm, Jansen [56] proved that the above algorithm is essentially optimal within the class of local greedy algorithms.

Ring Networks

The path coloring problem in $\mathcal{NP}$-hard in undirected [36] and in bidirected rings [83, 36]. Raghavan and Upfal [77] observed that a simple edge-removal technique yields a 2-approximation for path coloring in undirected ring networks. They also showed that this technique can be generalized to trees of rings. This gives a 3-approximation algorithm for path coloring in undirected tress of rings, and a $10/3$-approximation algorithm in bidirected trees of rings.

Coloring a set of fixed paths in undirected or bidirected ring networks is equivalent to the problem of coloring circular-arc graphs. Garey et al. [47] proved this problem to be $\mathcal{NP}$-hard. A $5/3$-approximation algorithm for the circular-arc coloring problem is given in [79].

On-line Path Coloring

In on-line path coloring the requests are presented to the on-line algorithm in some externally specified sequence, and the algorithm must immediately assign a path and a color to the request without any knowledge about the future requests. The goal is to use as few colors as possible. For fixed paths this problem becomes the on-line graph coloring problem with the conflict graph of the paths as input. A survey of results for on-line graph coloring can be found in [63].

We concentrate on the coloring of interval graphs. If the maximum clique size of the interval graph is ω, it is clear that ω colors are necessary for a proper coloring. In the off-line case, ω colors are also sufficient [49]. In the on-line case, a classical result is due to Kierstead and Trotter [65]. They presented an on-line algorithm for coloring interval graphs that uses at most $3\,\omega - 2$ colors and also showed that this is indeed best possible.

Another line of research on on-line interval coloring aimed at analyzing the First-Fit algorithm. This algorithm assigns each arriving interval the smallest color that is different from the colors of all previously processed intervals intersecting the current interval. It is known that First-Fit may need at least $4.4\,\omega$ colors [29] on some instances and is therefore not optimal. A linear upper bound of $40\,\omega$ was first presented by Kierstead [62] and later improved to $25.72\,\omega$ by Kierstead and Qin [64].

Bartal and Leonardi [15] study the on-line path coloring problem in trees, trees of rings, and meshes. They obtain deterministic algorithms with competitive ratios $O(\log n)$ for all these topologies, if the networks consist of n nodes each.

Chapter 3

A Linear-Time Optimal Algorithm for CAC in Chains

This chapter is concerned with the efficient implementation of a greedy algorithm for the following CAC problem in chain networks with arbitrary capacities.

Problem	<Chain $\mid b_i = 1, w_i = 1$>
Given:	A chain network with n nodes and arbitrary edge capacities and a set $\mathcal{R}$ of ℓ paths specified by their left and right endnodes. Each path has unit profit, and asks for one unit of bandwidth on all of its edges.
Goal:	Compute a feasible subset $Q \subseteq \mathcal{R}$ of maximum cardinality.

We number the nodes in the chain network (V, E) from 0 to $n - 1$ along the chain. Next we order the paths in non-decreasing order of their right endnodes, and call this ordering the *greedy order* of the paths. We remark that this order is not unique, if at least two paths have the same right endnode. See Figure 3.1 for an example.

It is folklore that the problem <Chain $\mid b_i = 1, w_i = 1$> is solved optimally by the following *greedy algorithm* that processes the paths one by one in greedy order. If adding the current path to the previously accepted paths does not

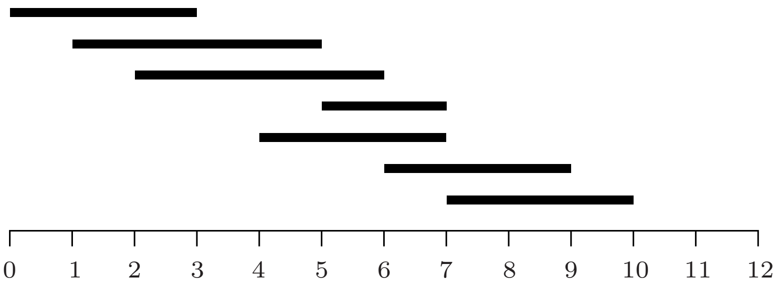


Figure 3.1: *A set of paths in greedy order (from top to bottom).*

violate any capacity constraint on its edges, the path is accepted; otherwise it is rejected.

Let $m = n - 1$ denote the number of edges in the chain. While an $O(m \cdot \ell)$ implementation of the greedy algorithm is straightforward, we show in the following that it can even be implemented in linear time $O(\ell)$, provided that a sorted list of all path endpoints is given (which can be computed in time $O(m + \ell)$ using bucket-sort). From the sorted list of path endpoints it is easy to determine the greedy order of the paths in linear time $O(\ell)$.

By a clever implementation of the greedy algorithm, Carlisle and Lloyd [22] obtained a linear-time algorithm for the case of equal edge capacities, that is for $<\text{Chain}, c_i = k \mid b_i = 1, w_i = 1>$. We outline their algorithm in the following section, and show in Section 3.2 how to adapt it to the case of arbitrary capacities.

Let $c_{max} = \max\{c(e) : e \in E\}$ denote the maximum edge capacity. Without loss of generality, we can assume $c_{max} \leq \ell$, since more than ℓ units of capacity cannot be used by ℓ paths.

3.1 The Algorithm for Equal Capacities by Carlisle and Lloyd

In this section we describe the linear-time implementation of the greedy algorithm due to Carlisle and Lloyd [22]. All edges have the same capacity $C := c_{max}$.

The main idea of their algorithm is to actually compute a C-coloring of the accepted paths and to maintain the *leader* of each color in a data structure. A path is called the *leader* of some color, if it is the greatest path in greedy order

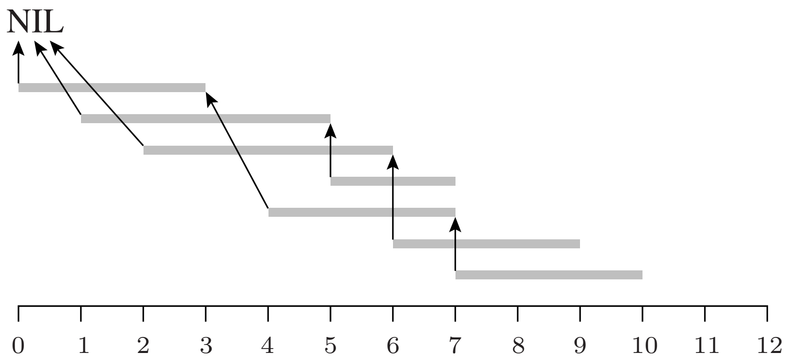


Figure 3.2: *The preferred leader $adj(p)$ for every path p.*

colored with that color so far. When a path p is processed, the data structure is used to determine the greatest leader in greedy order that does not intersect the current path p. This leader is called the *best fit* leader for the path p, and is denoted by $leader(p)$. If no such leader exists, p is rejected. Otherwise, p is assigned the color of $leader(p)$ and becomes the new leader of that color.

The difficulty is in finding the best fit leader in amortized constant time, since examining all colors would take $O(\log C)$ time per path using binary search. Carlisle and Lloyd show in [22] how the union-find data structure of Gabow and Tarjan [44] can be used for this purpose. They start by computing a preferred leader $adj(p)$ for every path p, which is the greatest path in greedy order ending to the left of p (see Figure 3.2). This can be accomplished in linear time by scanning through the paths in greedy order (which they assume to be given).

Initially, every path constitutes a singleton set in the union-find data structure. Throughout the algorithm, the paths that have already been processed are contained in sets consisting of a collection of consecutive paths in greedy order, and the paths that are not yet processed are in singleton sets. The representative of a set is always the smallest path in greedy order contained in that set. As the algorithm proceeds, this representative of a set will be either a leader, if the paths in the set have already been processed, or, otherwise, a path that has not been processed yet at that time. Thus, when the algorithm processes a path p and finds that $adj(p)$ is not a leader, the representative of the set containing $adj(p)$ must be the best fit leader for p. This representative can be determined by performing the find-operation $\text{find}(adj(p))$. To handle the case that no best fit leader for p exists, a fictitious leader to the left of all intervals can be introduced.

Using this data structure, the algorithm works as follows. When a path p is processed and $adj(p)$ is really a leader, then $adj(p)$ is clearly the best fit leader for p. Otherwise, $adj(p)$ has either been rejected or is no longer a leader. Then the operation $\text{find}(adj(p))$ is used to determine the largest leader in greedy order ending no later than $adj(p)$. This is the best fit leader for p in this case. If such a leader $q = leader(p)$ is found, the path p is colored with the color of q and the union-find data structure is updated. The path q is no longer the leader of its color, since p is the new leader now. Accordingly, its set is merged with the set containing the predecessor of q in greedy order using a union-operation on these sets. Since the predecessor of q is smaller than q in greedy order, after the union-operation, q is no longer the representative for the resulting set. The fact that the path p is the new leader of its color is reflected in the data structure, since p continues to be the representative of its singleton set. On the other hand, if no leader is found, the path p is rejected and the sets containing p and the predecessor of p in greedy order are merged by a union-operation. This has the effect that the rejected path p will never be returned as a leader for the following paths. We refer to [22] for a detailed explanation why this yields a correct implementation of the greedy algorithm in the case where all edge capacities are equal.

3.2 Adaptation to Arbitrary Capacities

Now we adapt this approach to the case of arbitrary capacities. As a first step, we add dummy paths to the instance to fill up the $C - c(e_i)$ units of extra capacity on every edge e_i as shown in Figure 3.3 for an example. In this new instance, we set all edge capacities equal to C and compute an optimal solution among all solutions that contain all dummy paths. Removing the dummy paths from the solution yields an optimal solution for the original problem, since no edge capacity will be violated.

Before we explain how to modify the above algorithm in such a way that all dummy paths are colored, we treat the dummy paths in more detail. They can be computed by scanning the chain from left to right and deciding at each node how many dummy paths should start or end here. If the edges to the left and to the right of the current node are e_{i-1} and e_i, then $c(e_i) - c(e_{i-1})$ dummy paths end at this node, if $c(e_i) > c(e_{i-1})$, and $c(e_{i-1}) - c(e_i)$ dummy paths begin at the node otherwise.

In order to achieve an overall linear running-time, the number of dummy paths must be $O(\ell)$. However, there are capacity functions where $\Omega(n \cdot \ell)$ dummy

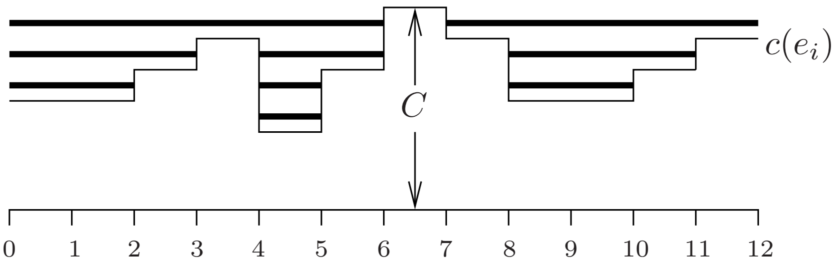


Figure 3.3: *The dummy paths for a given capacity function.*

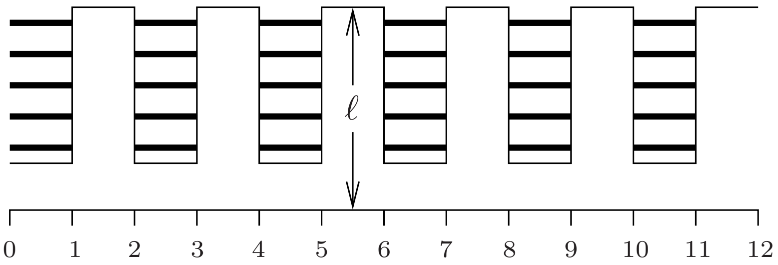


Figure 3.4: *A capacity function alternating between* 1 *and* ℓ *requires* $\Omega(n \cdot \ell)$ *dummy paths.*

paths are needed as shown in Figure 3.4.

To make sure that we add only $O(\ell)$ dummy paths, we introduce the following preprocessing step that flattens the capacity function. We scan the nodes of the chain from left to right. Let $n(i)$ denote the number of original paths that have node i as their left endpoint. For each edge e_i we set the new capacity $c'(e_i)$ to the minimum of the original capacity $c(e_i)$ and $c'(e_{i-1})+n(i)$, where we take $c'(e_{-1}) = 0$. Hence, a decrease in the original capacity function is replicated by the new capacity function, while an increase is limited to the number of paths starting at the current node. A larger increase could not be filled out by the $n(i)$ starting paths anyway. We have $c'(e_i) \leq c(e_i)$ for all edges e_i. Further, any subset of paths that is feasible for the capacity function c is also feasible for the new capacity function c'. To see the latter, note that the number of paths using edge e_i in any feasible solution for the capacity function c is at most $c(e_{i-1})+n(i)$. The new capacity function c' can clearly be computed in linear time.

Lemma 3.1 *With the new capacity function* c'*, the number of dummy paths added by the algorithm is* $O(\ell)$*.*

Proof. Let us define the *total increase* of the capacity function c' to be the sum of the values $\max\{c'(e_i) - c'(e_{i-1}), 0\}$ for $i = 0, \ldots, m-1$, where we take $c'(e_{-1}) = 0$. By definition of c', the total increase of c' is at most ℓ, since every increase by 1 can be charged to a different path. Now consider the dummy paths added by the algorithm. Every dummy path ends at an increase by 1 of the capacity function c' or because the right end of the chain is reached. Therefore, there can be at most $\ell + C$ dummy paths, which is $O(\ell)$ because we assumed $C = c_{max} \leq \ell$. □

After preprocessing the capacity function and adding the dummy paths, we compute a maximum C-colorable subset of paths in which all dummy paths are colored. It is clear that then the set of colored original paths forms an optimal solution in the original chain (with capacities $c(e_i)$ or $c'(e_i)$).

We must modify the algorithm of [22] to make sure that all dummy paths are accepted and colored. First, we compute a sorted list of the endpoints of all paths, including the dummy paths, such that for every node i, the right endpoints of paths ending at node i come before the left endpoints of paths starting at node i. The endpoints are processed in this order. Now the idea is to process original paths at their right endpoints and dummy paths at their left endpoints to make sure that all dummy paths are accepted and colored.

We now describe the modified algorithm. First, we initialize the data structure as in [22] with C *virtual* paths that serve as initial leaders for colors 1 to C, and a *fictitious* leader f. The C virtual paths are considered to be located to the left of node 0, e.g., paths from node -2 to node -1, and the fictitious leader is to the left of the C virtual paths, e.g., a path from -4 to -3. We extend the greedy order to these $C+1$ additional paths by taking first the fictitious leader, then the C virtual paths, and then the original and dummy paths. During the execution of the algorithm, paths whose best fit leader is the fictitious leader will be rejected. After the initialization, all paths (including the virtual paths and the fictitious leader) are in singleton sets of the union-find data structure, and the C virtual paths and the fictitious leader are considered to have been processed.

The algorithm maintains at any point the last path whose right endpoint has already been processed. This path is called *last* and is stored in a variable with the same name. As part of the initialization, we store the last virtual path in *last*.

Now the endpoints of the real paths (original paths and dummy paths) are processed in sorted order as follows. Let x be the path endpoint currently being processed and let p be the respective path. We distinguish two cases:

Case 1: x is the left endpoint of p.

Then we set $adj(p)$ to be the path stored in *last*, since this path would be the preferred leader for the path p. If p is a dummy path, we want to color p immediately and perform a find operation on $adj(p)$ to find $q = leader(p)$. We color p with the color of q and perform a union operation to merge the set containing q with the set containing the predecessor of q in greedy order, because q is no longer a leader. If p is not a dummy path, nothing needs to be done for p now, because p will be colored (or rejected) later when its right endpoint is processed.

Case 2: x is the right endpoint of p.

Then p is stored in *last*, since it is now the last path whose right endpoint has already been processed. If p is an original path, we want to color it now, if possible. Therefore, we perform a find operation on $adj(p)$ in order to find its leader. If such a leader q is found, the color of p is set to the color of q, and the set containing q is merged with the set containing the predecessor of the path q in greedy order. Otherwise, the find operation on $adj(p)$ has returned the fictitious leader meaning that no leader was found for p. In this case, the path p is rejected and the set containing p is merged with the set containing the predecessor of the path p. If p is a dummy path, p has already been colored at its left endpoint, so nothing needs to be done for p anymore.

The union-find data structure of [44] is applicable, since the structure of the potential union operations is a tree (actually, even a chain). To see this, observe that we always merge a set containing a path p with the set containing the immediate predecessor of p in greedy order. Therefore, the algorithm runs in time linear in the number of all paths including the dummy paths.

We refer to the resulting algorithm as the *union-find based algorithm* and show that it is a correct implementation of the greedy algorithm (in the sense that it accepts the same subset of the original paths).

Lemma 3.2 *The union-find based algorithm is a correct implementation of the greedy algorithm.*

Proof. First, observe that the sets in the union-find data structure are maintained such that the following property is ensured: the representative of the set containing a path that has already been processed is indeed the leader with rightmost right endpoint among all leaders that do not come after the path in greedy order (or the fictitious leader if no such leader exists). Now it is easy to see that the union-find based algorithm, denoted by U from here on, computes a C-colorable subset of the paths that includes all dummy paths. We only need to show that the algorithm accepts the same original paths as the greedy algorithm.

Consider the first path p in greedy order for which the result of U differs from the output of the greedy algorithm. It must be the case that p is accepted by the greedy algorithm, but rejected by U. Algorithm U processes p at its right endpoint. Assume that U does not accept p. Then the leaders of all C colors must intersect p. Let A be the set of all paths (dummy paths and original paths) that were accepted by U before processing p. Following arguments used in [22], we will show that $A \cup \{p\}$ creates load $C + 1$ on some edge, which is a contradiction to the fact that the greedy solution (which contains all original paths in $A \cup \{p\}$ because of the choice of p) is feasible.

Let e be the rightmost edge of the leader r whose right endpoint is as far left as possible among the C leaders intersecting p. Assume that there is a color i, $1 \leq i \leq C$, such that no path of color i contains the edge e. Then the set of paths colored with i must contain a path p_1 to the left of e (possibly the virtual path colored i) and a path p_2 to the right of e (which exists by the choice of r). Choose such p_1 and p_2 with rightmost right endpoint and leftmost left endpoint, respectively. At the time when p_2 was colored by U, p_1 was the leader of color i and r was the leader of a different color. Since r does not intersect p_2 and contains the edge e, which is strictly to the right of p_1, the path p_1 could not have been the best fit leader for p_2 (since r is a better leader). This contradicts the fact that U colored p_2 with color i. We conclude that the edge e is contained in a path of every color and also in the path p, implying that $A \cup \{p\}$ creates load $C + 1$ on edge e. Thus, we arrive at the desired contradiction. $\square$

Summing up, our algorithm does a linear-time preprocessing of the capacity function, then adds $O(\ell)$ dummy paths in linear time, and finally uses an adapted version of the algorithm by Carlisle and Lloyd [22] to run the greedy algorithm in linear time.

Theorem 3.3 *The greedy algorithm solves the call admission control problem* $<$*Chain* $|\, b_i = 1,\, w_i = 1>$ *optimally and can be implemented to run in time* $O(n + \ell)$*, where* n *is the number of nodes in the chain and* ℓ *is the number of given paths.*

Chapter 4

Call Admission Control in Rings

This chapter deals with call admission control problems in rings. As a main result we present a polynomial-time algorithm that solves the CAC problem $<$Ring, det $|\, b_i = 1, w_i = 1>$ optimally. We outline how this algorithm can be implemented efficiently using the linear-time algorithm for solving $<$Chain $|\, b_i = 1, w_i = 1>$ described in Section 3.2.

We start with a motivational discussion including two applications.

4.1 Motivation

The ring topology is a fundamental network topology that is frequently encountered in practice. Therefore, the problems we consider in this chapter apply to various types of existing communication networks with ring topology. One example are ring networks that support bandwidth reservation, like ATM networks for example, in which the route taken by a request is determined by some other mechanism and cannot be modified by the CAC algorithm. Another example are bidirectional self-healing rings with full protection. In such rings, one direction of the ring (say, clockwise) is used to route all accepted requests during normal operation, and the other direction is used only in case of a link failure in order to reroute the active connections that are affected. In all-optical WDM ring networks with w wavelengths that have a wavelength converter in at least one node, any set of lightpaths with maximum link load w

can be established simultaneously [83]. Thus, call admission control in such networks can also be modeled as $<$Ring, $c_i = w$, det $|$ $b_i = 1$, $w_i = 1$$>$, where all edge capacities are equal to w.

4.1.1 Applications

Call admission control in rings is closely related to periodic scheduling problems, which are of the following kind. Assume that there are jobs that have to be executed regularly every day at the same time on some fixed number k of machines. Each machine can execute only one job at a time. The jobs are specified by their starting and their finishing time and last less than 24 hours. For instance, a job could start at 8am and finish at 3pm, but a job could also last till the following day, e.g. from 9pm to 6am. Each job is either executed regularly every day or it is rejected. The goal of the periodic scheduling problem is to accept as many jobs as possible under the constraint that at any time at most k jobs are executed simultaneously. This problem can be translated in an obvious manner to a CAC problem in ring networks. The ring represents the time during one day. If we place a node in the ring at each starting and finishing time of the jobs, we can interpret each job as a call from the node representing its starting time to the node representing its finishing time. We obtain a CAC problem $<$Ring, $c_i = k$, det $|$ $b_i = 1$, $w_i = 1$$>$, and our optimal algorithm can be used to compute an optimal periodic schedule. Of course, this can be generalized to the case, where the number of machines varies throughout the day (but is the same every day), since our CAC algorithm is capable of handling arbitrary edge capacities.

Another problem to which our algorithm applies comes from cyclical scheduling of work shifts studied by Hochbaum and Levin [55]. In their scenario at each time slot there is a requirement for a certain number of workers, which must be present at that time. The work schedules are organized in shifts, where each shift is a collection of consecutive time slots and incurs a specific cost. The goal of their problem is to assign workers to the different shifts in such a way that the requirements are met and the cost of the assignment is minimized. Hochbaum and Levin model this problem as a bounded multicover problem (MCB) and formulate it as an integer linear program of the following form.

$$
\begin{aligned}
\text{(MCB)}\quad \text{Min.}\quad & \sum_{j=1}^{m} c_j x_j \\
\text{s.t.}\quad & \sum_{j=1}^{m} a_{ij} x_j \geq b_i,\ i = 1, \ldots, n, \\
& 0 \leq x_j \leq u_j,\ j = 1, \ldots, m, \\
& x_j \text{ integer.}
\end{aligned}
$$

The variables x_j, $j = 1, \ldots, m$, stand for the number of workers assigned to the shifts. Each shift j, $j = 1, \ldots, m$, has a cost c_j and an upper bound u_j on the number of workers assigned to it. Each column of the 0-1-matrix A represents one shift as a consecutive interval of ones, if the first element in each column is considered to be the successor of the last element of that column, i.e. the columns are either consecutive or cyclical. We say that the matrix A has the circular-ones property in its columns. The bounds b_i specify the requirements of workers for each time slot i, $i = 1, \ldots, n$.

We will show how our algorithm for $\langle$Ring, det $|\, b_i = 1, w_i = 1\rangle$ can be used to solve MCB problems optimally in polynomial time provided that $c_j = 1$ for all j and the u_j's are polynomially bounded in n and m. This requires some transformations. First, we need 0-1 variables instead of integer variables. This is accomplished by replacing each variable x_j by u_j copies of itself and setting the upper bounds of the resulting variables to 1. We also blow up the matrix A by substituting each column by u_j copies of itself. Since the u_j's are bounded, the resulting instance has a size that is polynomial in n and m. Now, we substitute $y_j := 1 - x_j$ for each new variable to get the following problem:

$$
\begin{aligned}
\text{min.}\quad & m - \sum_{j=1}^{m} y_j \\
\text{s.t.}\quad & \sum_{j=1}^{m} a_{ij} y_j \leq -b_i + \sum_{j=1}^{m} a_{ij},\ i = 1, \ldots, n, \\
& 0 \leq y_j \leq 1,\ j = 1, \ldots, m, \\
& y_j \text{ integer.}
\end{aligned}
$$

Here, m denotes the number of new variables, which is equal to $\sum_j u_j$ in terms of the original problem. The c_j's disappear because they are assumed to be 1.

The sum $\sum_{j=1}^{m} a_{ij}$ counts the number of shifts that have a 1 at time slot i. This number should be at least b_i, because otherwise the MCB problem would be infeasible. Therefore, the right hand sides $-b_i + \sum_{j=1}^{m} a_{ij}$ of the constraints are positive fixed values depending only on b and the matrix A. We denote them by c_i in the following. Regarding the values of the variables y_i in an optimal solution, the objective function "$\text{min. } m - \sum_{j=1}^{m} y_j$" is equivalent to "$\text{max. } \sum_{j=1}^{m} y_j$". Thus, the problem translates to

$$\begin{aligned} \text{max.} \quad & \sum_{j=1}^{m} y_j \\ \text{s.t.} \quad & \sum_{j=1}^{m} a_{ij} y_j \leq c_i, \ i = 1, \ldots, n, \\ & 0 \leq y_j \leq 1, \ j = 1, \ldots, m, \\ & y_j \text{ integer,} \end{aligned}$$

which is the natural ILP formulation for $<$Ring, det $|\, b_i = 1, w_i = 1>$. Each column of the matrix A represents a path, i.e. the 1's indicate the edges used by this path. The capacity of the edge i is given by c_i. Our algorithm can be used to compute an optimal solution for this problem. If in this solution k_j of the u_j variables y_j belonging to the shift j are equal to 1, scheduling $u_j - k_j$ workers for shift j constitutes an optimal solution for the original MCB problem. Altogether, this yields the claimed polynomial-time optimal algorithm for MCB with unit costs and polynomially bounded u_j-values.

Note that the above reduction from MCB problems to CAC problems in rings works also in the weighted case, i.e. for MCB instances with arbitrary costs c_j for the shifts.

Hochbaum and Levin [55] show that the unbounded version of the MCB problem (with $u_j = \infty$ for all j) can be solved optimally in polynomial time provided that all b_i's are equal. For the general MCB, they present a 2-approximation algorithm.

4.2 An Optimal Algorithm for CAC in Rings

The call admission control problem considered in this section is defined as follows.

Problem	<Ring, det $\mid b_i = 1, w_i = 1$>
Given:	A ring network with n nodes and arbitrary edge capacities and a set $\mathcal{R}$ of ℓ pre-determined paths. Each path has unit profit, and asks for one unit of bandwidth on all its edges.
Goal:	Compute a feasible subset $Q \subseteq \mathcal{R}$ of maximum cardinality.

A ring network with n nodes is modeled by an undirected graph (V, E) that is a cycle on the nodes $V = \{0, \ldots, n-1\}$. We imagine this cycle drawn in the plane with the nodes labeled clockwise. The edge $e_i \in E$, $0 \leq i < n$, connects the two neighboring nodes i and $(i+1) \bmod n$ and is associated with its positive capacity $c(e_i)$.

The ℓ paths are given as a set $\mathcal{R} = \{p_1, \ldots, p_\ell\}$. Every $p_i \in \mathcal{R}$ is an ordered pair of nodes $p_i = (s_i, u_i)$ from V^2 with $s_i \neq u_i$. The path p_i contains all edges from the *source node* s_i to the *target node* u_i in clockwise direction. The paths represent connection requests whose acceptance requires the reservation of one unit of bandwidth on all edges of the path. A *feasible solution* is a set $Q \subseteq \mathcal{R}$ such that for every edge $e \in E$, the number of paths in Q that contain e is at most $c(e)$. The paths in Q are called *accepted*, and the objective is to maximize the number of accepted paths.

Whenever we talk about a set of paths, we always allow that the set is actually a multi-set, i.e., that it may contain several instances of the same path.

We can assume without loss of generality that $n \leq 2\ell$, since every node in the ring that is not an endpoint of a path can be removed. Figure 4.1 shows a set of 7 paths in a ring network on 8 nodes. For example, the path p_1 is given by the pair $(0, 3)$ of nodes and contains the edges e_0, e_1, and e_2.

In the following we introduce some more notations that will help us to explain and to analyze the algorithm. For a subset $Q \subseteq \mathcal{R}$, the *ringload* $L(Q, e_i)$ of an edge e_i with respect to Q is the number of paths in Q that contain the edge e_i, i.e. $L(Q, e_i) := |\{p \in Q : e_i \in p\}|$. Therewith, feasibility can be expressed in terms of the ringload. A subset $Q \subseteq \mathcal{R}$ is called feasible, if and only if the ringload does not exceed the capacity of any edge, i.e. $L(Q, e_i) \leq c(e_i)$ for all $e_i \in E$.

By opening the ring at node 0, we partition the set $\mathcal{R}$ of paths into two disjoint

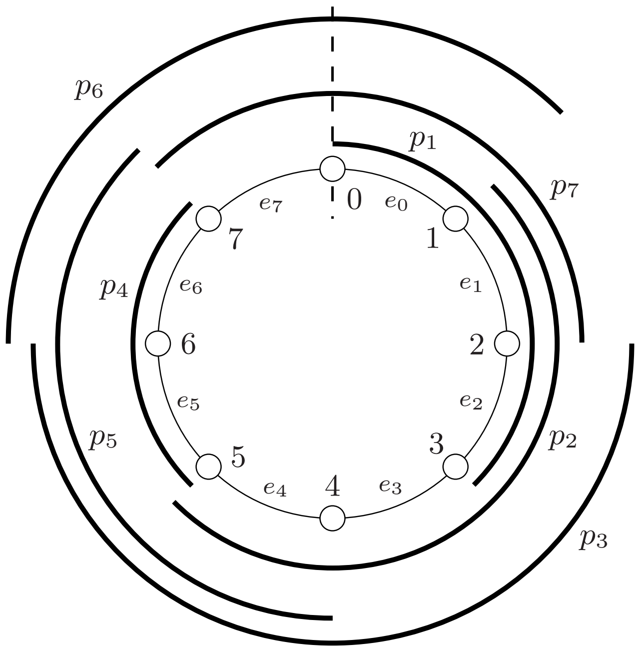


Figure 4.1: *A set of paths in a ring network with 8 nodes.*

subsets $\mathcal{R}_1$ and $\mathcal{R}_2$, where $\mathcal{R}_1$ is the set of paths that do not have node 0 as an internal node and $\mathcal{R}_2$ are the remaining paths, i.e., the paths passing through node 0. Each path in $\mathcal{R}_2$ consists of two pieces: the *head* of the path extends from its source node to node 0, and the *tail* spans from node 0 to its target node. To simplify the explanation, we introduce an additional node n and identify it with node 0. From now on, the paths with target node 0 are treated as if they ended at node n. With this twist in notation we get a simple characterization for the sets $\mathcal{R}_1$ and $\mathcal{R}_2$. The set $\mathcal{R}_1$ consists of all paths $p_i \in \mathcal{R}$ with $s_i < u_i$, whereas all paths $p_i \in \mathcal{R}$ obeying $s_i > u_i$ constitute the set $\mathcal{R}_2$. Note that the disjoint union $\mathcal{R}_1 \dot{\cup} \mathcal{R}_2$ equals $\mathcal{R}$. In the example of Figure 4.1 the set $\mathcal{R}_1$ consists of the paths $p_1, \ldots, p_5$, and the set $\mathcal{R}_2$ is composed of the paths p_6 and p_7.

We define a linear ordering on the paths in $\mathcal{R}$ as follows. All paths in $\mathcal{R}_1$ are strictly less than all paths in $\mathcal{R}_2$. Within both subsets we order the paths by increasing target nodes, resolving ties arbitrarily. We call this ordering *greedy*. In the above example the paths $p_1, \ldots, p_7$ are numbered according to that greedy order.

Instead of the ring network the algorithm considers a chain of $2n$ edges consisting of two copies of the ring glued together as shown in Figure 4.2. The

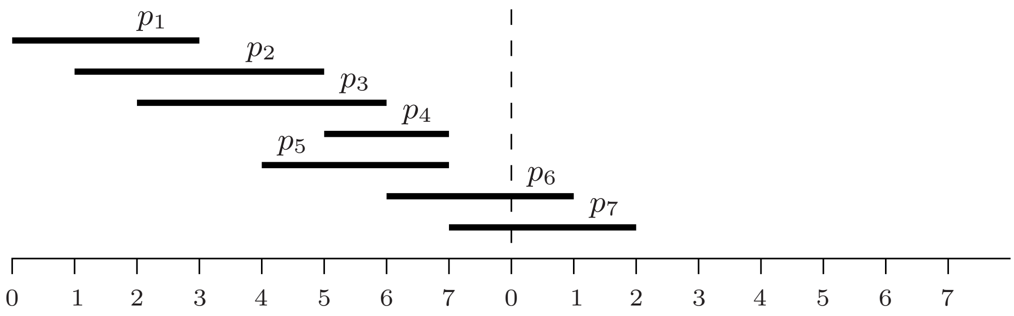


Figure 4.2: *The same set of paths in the chain with 16 edges.*

chain begins with the first copy of e_0 and ends with the second copy of e_{n-1}. The tails of the $\mathcal{R}_2$-paths extend in the second copy of the ring, while the $\mathcal{R}_1$-paths and the heads of the $\mathcal{R}_2$-paths are situated in the first copy. Note that the greedy order of the paths in the ring corresponds to the greedy ordering by non-decreasing right endpoints in the constructed chain of length $2n$ (as described in Chapter 3).

For a given set Q of paths, we define $L_1(Q, e_i)$ and $L_2(Q, e_i)$ to be the load of the paths in Q on the first copy of e_i and their load on the second copy of e_i, respectively. Thus, the paths in $\mathcal{R}_1$ and the heads of the paths in $\mathcal{R}_2$ contribute to the load values $L_1(\mathcal{R}, e_i)$ of the first copy of the ring. The tails of the $\mathcal{R}_2$-paths determine the load values $L_2(\mathcal{R}, e_i)$. The ringload L is simply the sum of L_1 and L_2. We use this definition of L_2 to introduce the central notion of profiles.

Definition 4.1 (Profiles) *A profile is a non-increasing sequence of n natural numbers for the edges $e_0, e_1, \ldots, e_{n-1}$. For a set Q of paths, the profile π_Q of Q is the sequence of n load values L_2 for the edges $e_0, \ldots, e_{n-1}$ in the second copy of the ring,*

$$\pi_Q := L_2(Q, e_0) \cdots L_2(Q, e_{n-1}).$$

With $\pi_Q(e_i)$ we denote the profile values $L_2(Q, e_i)$ for all edges $e_i \in E$. The empty profile is zero everywhere. For profiles π and π', we write $\pi \leq \pi'$ if $\pi(e_i) \leq \pi'(e_i)$ for all edges $e_i \in E$.

In the instance shown in Figure 4.2, only the paths p_6 and p_7 potentially contribute to profile values, because they extend into the second copy of the ring. For example the profile of all paths in $\mathcal{R}$ is given by the sequence $\pi_{\mathcal{R}} = 21000000$.

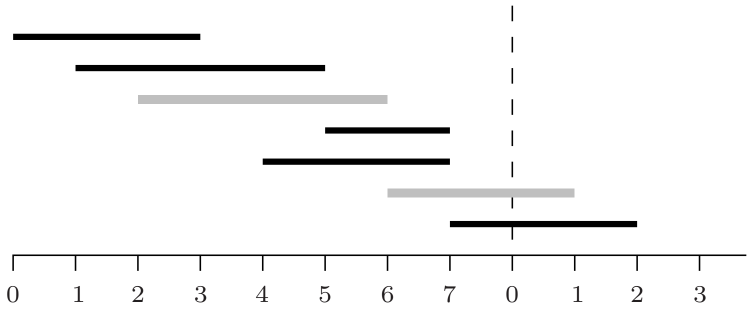
A set Q of paths is called *chain-feasible* if it does not exceed the capacity of any edge in the constructed chain of length $2n$. In other words, Q is *chain-feasible* if it does not exceed the capacities in both copies of the ring, i.e. $L_1(Q, e_i) \leq c(e_i)$ and $L_2(Q, e_i) \leq c(e_i)$ hold for all $e_i \in E$. It is called *chain-feasible for (starting) profile* π if it is chain-feasible and in the first copy of the ring the stronger inequalities $L_1(Q, e_i) + \pi(e_i) \leq c(e_i)$ hold for all $e_i \in E$. Observe that a set Q of paths is feasible (in the ring), if and only if it is chain-feasible for starting profile π_Q, which is the profile generated by the set Q itself.

4.2.1 The Algorithm

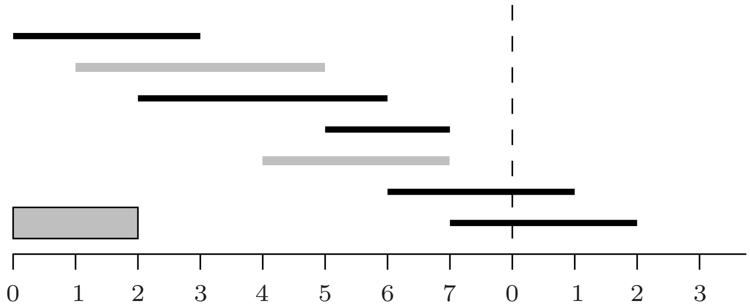
The goal of the algorithm is to find a feasible subset of paths in $\mathcal{R}$of maximum size. The algorithm builds a chain of $2n$ edges consisting of two copies of the ring glued together. It sorts the paths in $\mathcal{R}$ according to the greedy order. The heart of our algorithm is a decision procedure that, given a parameter k, decides whether there exists a feasible solution $Q \subseteq \mathcal{R}$ of size k or not. Clearly, the maximum k can be found by several calls to this procedure. The decision procedure operates in rounds, each of which is an execution of the *greedy algorithm* explained in Section 3, which processes the paths one by one in greedy order. If adding the current path to the previously accepted paths does not violate any capacity constraint on its edges, the path is accepted; otherwise it is rejected.

We are now ready to describe the decision procedure. We start with the empty profile. The decision procedure works in rounds. In each round, it computes a greedy solution of k paths for a given profile as follows. It initializes both copies of the ring with the edge capacities $c(e_i)$ and subtracts the profile values from the initial capacities of the edges in the first copy, since these capacities are occupied by the profile. Then, it starts to place k paths using the greedy algorithm. If the procedure runs out of paths before it can select k of them, there is no feasible solution of size k. It answers "no" and stops. Otherwise, let Q_i denote the candidate set of k chosen paths in round i. By construction, the set Q_i is chain-feasible for the given starting profile, but not necessarily feasible in the ring, since the tails of the chosen $\mathcal{R}_2$-paths together with the selected paths in $\mathcal{R}_1$ and the heads of the chosen $\mathcal{R}_2$-paths may violate some capacity constraints.

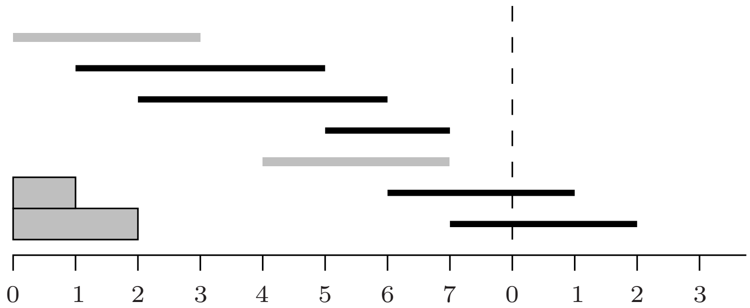
At the end of round i, the procedure compares the profile of Q_i with the profile of the previous round. If both are equal, the paths in Q_i form a feasible solution of size k. The procedure outputs Q_i, answers "yes," and stops. Otherwise,



(a) The candidate set Q_1 of the first round.



(b) The candidate set Q_2 of the second round. The profile π_{Q_1} is shown as a grey rectangle.



(c) The set Q_3 forms a feasible solution.

Figure 4.3: *The decision procedure in rounds. Accepted paths are plotted black, whereas rejected paths are shown in light grey.*

the procedure uses the profile of Q_i as the starting point for the round $i+1$. As we will prove later, the profiles of such a greedily chosen Q_i serve as a lower bound for any feasible solution in the sense that there exists no feasible solution with a smaller profile.

Figure 4.3 illustrates the execution of the decision procedure for the ring network and the paths shown in Figure 4.1. Let the capacities be $c(e_i) = 2$ for all edges e_i, $i = 0, \ldots, 7$. We ask for a feasible solution consisting of $k = 5$ paths. The paths are always processed in greedy order, which can be seen in Figure 4.2. We start with the empty profile. In the first round the paths p_1 and p_2 are accepted. The path p_3 is rejected, because it violates the capacity constraint of the edge e_2 after the paths p_1 and p_2 have already been accepted. The paths p_4 and p_5 are both accepted, and the path p_6 is rejected, because otherwise the edge e_6 would be overloaded. Finally, the path p_7 is accepted to form the candidate set $Q_1 = \{p_1, p_2, p_4, p_5, p_7\}$ of 5 paths shown in Figure 4.3(a). The profile of Q_1 is 1 on the first two edges e_0 and e_1, and 0 elsewhere. Q_1 is not feasible because the load $L(Q_1, e_1) = 3$ exceeds the capacity $c(e_1) = 2$.

The procedure starts the second round, this time with the profile of Q_1 as the starting profile. In this round the procedure accepts the paths in $Q_2 = \{p_1, p_3, p_4, p_6, p_7\}$ illustrated in Figure 4.3(b). The path p_2 is rejected this time, because the edge e_1 is saturated by the profile of Q_1 and the path p_1 that is already accepted. The path p_5 is rejected because of the capacity constraint of the edge e_5. The profile of Q_2 is 2 for the edge e_0, 1 for the edge e_1, and 0 elsewhere. Like Q_1, the candidate set Q_2 is not feasible. Its load on the edge e_0 is 3, violating the capacity constraint for edge e_0, which has capacity $c(e_0) = 2$.

In the third round, the procedure starts with the profile of Q_2, and accepts the paths in $Q_3 = \{p_2, p_3, p_4, p_6, p_7\}$ depicted in Figure 4.3(c). Their profile is again 2 for the edge e_0, 1 for the edge e_1, and 0 elsewhere. Since this resulting profile π_{Q_3} is equal to the starting profile π_{Q_2}, the set Q_3 is a feasible solution for the ring of size 5. The procedure outputs Q_3, and stops.

4.2.2 Correctness of the Algorithm

The decision procedure will generate a sequence of profiles and chain-feasible solutions

$$\pi_0 \quad Q_1 \quad \pi_1 \quad Q_2 \quad \pi_2 \quad \ldots,$$

where π_0 is the empty profile we start with, and Q_i denotes the chain-feasible solution computed in round i. We set $\pi_i := \pi_{Q_i}$.

We represent a chain-feasible solution A by the indices of the chosen paths in greedy order. A chain-feasible set A of k paths corresponds to a k-vector $A = (a_1, a_2, \dots, a_k)$, where a_i is the index of the i-th path in A according to the greedy order. If A and B are two chain-feasible solutions, we write $A \le B$ if $a_i \le b_i$ for all $1 \le i \le k$.

Note that $A \le B$ implies $\pi_A \le \pi_B$. This can be seen by comparing the i-th path in A with the i-th path in B: Since their indices a_i and b_i satisfy the condition $a_i \le b_i$ for all i, the paths in A contribute no more to the profile values $\pi_A(e_j)$ than the paths in B add to their respective profile values $\pi_B(e_j)$ for all edges e_j. Thus, $\pi_A \le \pi_B$.

From $\pi \le \pi'$ it follows easily that any chain-feasible solution for profile π' is also chain-feasible for profile π. In the following, we call a solution A that is chain-feasible for profile π *minimal*, if for any other solution B that is chain-feasible for π and has the same cardinality as A, we have $A \le B$.

Lemma 4.2 (Optimality of greedy algorithm) *Let π be some starting profile. If there exists a solution of size k that is chain-feasible for profile π, then the greedy algorithm computes a minimal solution among all such solutions.*

Proof. Let Q be any chain-feasible solution for profile π of size k. We transform Q step by step into the greedy solution G by replacing paths in Q by paths in G with smaller index. This is done during the execution of the greedy algorithm as it processes the paths in greedy order. We maintain the invariants that Q is always a chain-feasible solution of size k and that Q is equal to G with respect to the paths that have been processed so far.

Initially, the invariants clearly hold. Suppose the invariants hold up to path p_{i-1}, and the greedy algorithm processes the path p_i.

If adding the path p_i violates some capacity constraint, p_i is not selected by the greedy algorithm. Because of the second invariant, the path p_i is not in Q either. Otherwise, the path p_i is chosen by the greedy algorithm. We distinguish two cases:

Case 1: $p_i \in Q$.

Since the path p_i is in both G and Q, no transformation is needed. Q remains feasible, and the invariants are met.

Case 2: $p_i \notin Q$.

In this case a transformation is needed. From the set of paths in Q with indices larger than i, we select a path p_j with the smallest source node (starting leftmost). We transform Q by replacing p_j by p_i. Since $j > i$, the index j in Q is reduced to i. We have to check the invariants. Since Q and G are clearly equal with respect to the paths that have been processed so far, we only have to verify the feasibility of Q. If the path p_i is contained in p_j, the invariant clearly holds, since replacing p_j by p_i does not affect feasibility. Otherwise, consider the remaining capacities. The edges to the left of the path p_j do not matter, because p_j has the smallest source node among all paths in Q greater than p_i. On the edges in the intersection of the paths p_i and p_j, taking either path p_i or p_j does not affect the capacities. Finally, we even gain one unit of capacity on all edges between the target node of the path p_i and the target node of the path p_j, since $i < j$. Altogether, Q is again feasible. The invariants hold.

At the end of the greedy algorithm, Q equals G. During the transformation we always replaced paths $p_j \in Q$ by paths $p_i \in G$ with $i < j$. This implies that G is less than or equal to the initial chain-feasible solution Q, i.e. $G \leq Q$ holds. $\square$

Lemma 4.3 *The sequence of profiles generated by the decision procedure is monotonically increasing, i.e., we have $\pi_i \leq \pi_{i+1}$ for all i.*

Proof (by induction). For $i = 0$, the claim holds, since the profile π_0 is the empty profile. Assume that the claim holds for $i-1$. The induction hypothesis $\pi_{i-1} \leq \pi_i$ implies that the greedy solution Q_{i+1}, which is chain-feasible for profile π_i, is also chain-feasible for the profile π_{i-1}. Because Q_i is the greedy solution for the profile π_{i-1}, we obtain $Q_i \leq Q_{i+1}$ by Lemma 4.2. Therefore, $\pi_i \leq \pi_{i+1}$. $\square$

Lemma 4.4 *If a feasible solution Q^* with k paths exists, then each profile in the sequence of profiles generated by the decision procedure is bounded by the profile of Q^*, i.e., we have $\pi_i \leq \pi_{Q^*}$ for all i.*

Proof (by induction). Because π_0 is the empty profile, the case $i = 0$ holds trivially. Now suppose $\pi_i \leq \pi_{Q^*}$ holds for some i. Since the set Q^* is chain-feasible for the profile π_{Q^*} generated by itself, it is also chain-feasible for the profile π_i. Then, the greedy solution Q_{i+1} satisfies $Q_{i+1} \leq Q^*$ by Lemma 4.2, which immediately implies $\pi_{i+1} \leq \pi_{Q^*}$. $\square$

Lemma 4.5 *The decision procedure gives correct results and terminates after at most $n \cdot c(e_0)$ rounds.*

Proof. Assume first that there exists a feasible solution Q^* with k paths. By Lemma 4.4, the profile of the chain-feasible solutions computed by the algorithm always stays below the profile of Q^*. By Lemma 4.3, in each round the profile either stays the same or grows. If the profile stays the same, a feasible solution has been found by the algorithm. If the profile grows, the algorithm will execute the next round, and after finitely many rounds, a feasible solution will be found.

Now assume that the answer is "no." Again, the profile grows in each round, so there can be only finitely many rounds until the algorithm does not find k paths anymore and returns "no."

We have $\sum_{j=0}^{n-1} \pi_{Q_i}(e_j) \leq n \cdot \pi_{Q_i}(e_0) \leq n \cdot c(e_0)$ for every generated profile π_{Q_i}, since profiles are non-increasing sequences and each Q_i is chain-feasible. As the profile grows in each round, the number of rounds is bounded by $n \cdot c(e_0)$. □

Theorem 4.6 *There is an algorithm that solves <Ring, det* $|\, b_i = 1$, $w_i = 1$> *optimally in time* $O(\ell n c_{min} \log \ell)$*, where* n *is the number of nodes in the ring,* ℓ *is the number of paths, and* c_{min} *is the minimum edge capacity (which can be assumed to be at most* ℓ *without loss of generality).*

Proof. By Lemma 4.5, we have a decision procedure with $n \cdot c(e_0)$ rounds to decide whether there exists a feasible solution with k paths. This can be improved to $n \cdot c_{min}$ rounds by labeling the nodes such that $c(e_0)$ equals the minimum edge capacity c_{min}. Each round is a pass through the ℓ given paths in greedy order, which is shown in Section 3.2 to be implementable in linear time $O(n + \ell) = O(\ell)$ (recall that $n \leq 2\ell$ without loss of generality). Given the decision procedure, we can use binary search on k to determine the maximum value for which a feasible solution exists with $O(\log \ell)$ calls of the decision procedure, since k ranges between 0 and ℓ. Thus, the total running-time of our algorithm is bounded by $O(\ell n c_{min} \log \ell)$. □

A minor improvement in the number of calls of the decision procedure may be obtained on certain instances as follows. We first run the greedy algorithm of Section 3.2 on the paths in $\mathcal{R}_1$. This yields an optimal feasible subset Q of $\mathcal{R}_1$. Let $q = |Q|$. Then we know that the size of an optimal feasible subset of $\mathcal{R}$ lies in the interval $[q, q + \min\{|\mathcal{R}_2|, c_{min}\}]$ (possibly after relabeling the nodes). Hence, the number of calls of the decision procedure is reduced to $O(\log \min\{|\mathcal{R}_2|, c_{min}\})$.

Chapter 5

Call Admission Control in Stars

In this chapter we consider call admission control problems in star networks. After a motivating discussion we give a review of the so-called *local ratio technique* introduced by Bar-Noy et al. [13] in the context of resource allocation and scheduling. In Section 5.3 we present a constant-factor approximation algorithm for the variant of CAC in stars with arbitrary edge capacities, in which the calls have time specifications, arbitrary bandwidth requirements, and arbitrary profits. This resolves an open problem due to Erlebach [35]. Our algorithm is randomized and achieves an approximation ratio of 18. It is fast, because it does not involve linear programming like earlier algorithms in this field [35], but is based on the local ratio technique instead. Our algorithm can be generalized to accommodate alternatives for the calls, in which case the approximation ratio increases to 24 as proved in Section 5.3.1. In Section 5.3.2 we show how these algorithms can be derandomized.

On the complexity side, we prove CAC in star networks to be $\mathcal{APX}$-hard even for very restricted variants of the problem. This is accomplished by reducing the $\mathcal{APX}$-complete bounded maximum 3-dimensional matching problem [59] to the restricted CAC variant. We also present a somewhat weaker $\mathcal{NP}$-hardness proof by a reduction from the well-known 3-CNF satisfiability problem of Boolean formulas. These hardness results are presented in Section 5.4.

In Section 5.5 we propose two constant-factor approximation algorithms that approximate these restricted variants within factors 2 and 3 from the respective optimal values. Additionally, we show in Section 5.6 that a parameterized version of one of these problems is fixed-parameter tractable.

5.1 Motivation

A lot of work on CAC considered the scenario that a connection request is presented to the network only at the time when the connection should be established. This model ignores the possibility that resources might be requested ahead of time when they are needed. A natural concept to incorporate this possibility is advance reservation [52, 71], which is attractive for both the network provider and the users. The network provider has the advantage of a higher flexibility and can use better algorithms to maximize the profit gained from the accepted calls. The user benefits from a guaranteed quality of service (QoS) once his/her call is accepted. QoS is desirable for example in the context of video-conferencing.

Interestingly, CAC algorithms for star networks apply to general networks, if the provider has rented the capacity of his/her network from another provider according to the hose model of bandwidth reservations. In the hose model, there are ingress and egress limits for each border router of the network, and the network guarantees that it has enough bandwidth for any traffic pattern that does not violate the ingress and egress limits (see [32]). Thus, the network is equivalent to a star network for the purpose of CAC. In [27] a scalable distributed CAC scheme based on the hose model of bandwidth reservations is described where CAC algorithms for stars are applicable.

It is an important task to investigate CAC problems with arbitrary capacities on the links of the network. For example, there is a constant-factor approximation algorithm for <Chain, $c_i = 1 \mid$ > [13], but for the variant <Chain $\mid$ > with no restriction on the bandwidth requirements, no constant-factor approximation algorithm is known so far. Constant-factor approximations for this problem are known only under the "no-bottleneck" assumption, that the maximum bandwidth requirement is not larger than the minimum edge capacity, i.e. for <Chain $\mid b_{max} \leq c_{min}$> [24, 25]. Before our results, the situation for star networks was very similar. A constant-factor approximation algorithm was known for <Star, $c_i = 1 \mid t_i$>, and an $O(\log C)$-approximation for <Star $\mid t_i$>, where $C = c_{max}/c_{min}$ is the ratio of the maximum edge capacity to the minimum edge capacity. These approximation algorithms are presented in a paper of Erlebach [35], who raised the open question whether there is a constant-factor approximation algorithm for $<\text{Star} \mid t_i>$. We answer this question in the affirmative by presenting such an algorithm in Section 5.3, thus resolving the undetermined situation.

5.2 The Local Ratio Technique

Our approximation algorithm is based on the local ratio technique which was developed by Bar-Yehuda and Even [14] and first used in our context by Bar-Noy et al. [13]. The general framework can be described as follows. Assume that we are given a profit vector $\mathbf{w} \in \mathbf{R}^n$ and a set $\mathcal{F}$ of feasibility constraints on vectors $\mathbf{x} \in \mathbf{R}^n$. A vector $\mathbf{x} \in \mathbf{R}^n$ is a *feasible* solution for the problem $(\mathcal{F}, \mathbf{w})$, if it satisfies all the constraints in $\mathcal{F}$. The *value* of a feasible solution $\mathbf{x}$ is the inner product $\mathbf{w} \cdot \mathbf{x}$. For a maximization problem, a feasible solution $\mathbf{x}$ is an r-approximation, if $\mathbf{w} \cdot \mathbf{x} \geq (1/r) \cdot \mathbf{w} \cdot \mathbf{x}^*$, where $\mathbf{x}^*$ is an *optimal* solution, i.e. a solution whose value is maximal among all feasible solutions. The power of the technique is based on the following theorem.

Theorem 5.1 *(Local Ratio [13]) Let $\mathcal{F}$ be a set of constraints and let* $\mathbf{w}$, $\mathbf{w}'$ *and* $\mathbf{w}''$ *be profit vectors such that* $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$. *Then, if* $\mathbf{x}$ *is an r-approximation with respect to $(\mathcal{F}, \mathbf{w}')$ and with respect to $(\mathcal{F}, \mathbf{w}'')$, then* $\mathbf{x}$ *is an r-approximation with respect to $(\mathcal{F}, \mathbf{w})$.*

For computing approximate solutions we will use the so-called *unified algorithm* proposed in [13]. We generalize our problem by allowing negative profits as well.

The unified algorithm:

1. Delete all calls with non-positive profit.

2. If no calls remain, return $Q = \emptyset$.

3. Otherwise, select a call i and decompose $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$. The choice of i and the decomposition depends on the problem at hand and will be discussed below.

4. Solve the problem recursively using $\mathbf{w}''$ as the profit vector. Let Q'' be the set returned.

5. If $Q'' \cup \{i\}$ is a feasible solution, return $Q = Q'' \cup \{i\}$. Otherwise, return $Q = Q''$.

We call a feasible solution i-*maximal*, if it either contains the call i, or it does not contain the call i, but adding i to the solution will render it infeasible. The following lemma analyzes the quality of the solution produced by the algorithm.

Lemma 5.2 *([13]) Let r be a constant. Suppose that in the algorithm above the choice of i and the decomposition $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$ is always such that:*

(1) $w''_i = 0$, and

(2) every i-maximal solution is an r-approximation with respect to $\mathbf{w}'$.

Then, the algorithm terminates and the solution Q produced is an r-approximation with respect to $\mathbf{w}$.

Proof. First note that since $w''_i = 0$, the call i will be deleted in the recursive call, and the algorithm will eventually terminate. Since Q is i-maximal, the condition (2) implies that Q is an r-approximation with respect to $\mathbf{w}'$. It remains to show that Q is also an r-approximation with respect to $\mathbf{w}''$ (then apply the local ratio theorem). We proceed by induction on the number of recursive calls of the algorithm. At the basis of the recursion, the algorithm returns the empty set, since no calls remain. This is clearly an optimal solution and, hence, also an r-approximation. For the induction step, assume that Q'' is an r-approximation with respect to $\mathbf{w}''$. Since $w''_i = 0$ and Q is either Q'' or $Q'' \cup \{i\}$, it follows that Q is an r-approximation with respect to $\mathbf{w}''$ as well. By the local ratio theorem (Theorem 5.1), Q is also an r-approximation with respect to $\mathbf{w}$. □

In the next section we apply the unified algorithm to a call admission control problem in star networks, in which all calls j are associated with a starting time t_j and a duration d_j. The selection of the call i made by the algorithm depends on these time specifications. We will always choose the call i that has the minimum end-time $t_i + d_i$ among all calls in question.

By Lemma 5.2 we only need to find a constant $r > 1$ such that every i-maximal solution is an r-approximation with respect to $\mathbf{w}'$. To do so, we will derive an upper bound w_{opt} on the optimum $\mathbf{w}'$-profit and a lower bound w_{max} on the $\mathbf{w}'$-profit of every i-maximal solution. The ratio $r = w_{opt}/w_{max}$ is then an upper bound on the approximation ratio of the algorithm.

5.3 Constant-factor Approximation Algorithms

In this section we first give a randomized constant-factor approximation algorithm for <Star | t_i> that achieves an approximation ratio of 18. In Section 5.3.1 we show how to accommodate call alternatives by presenting a 24-approximation algorithm for <Star | t_i, alt>. We are closing this section with Section 5.3.2, in which we explain how these two algorithms can be derandomized to obtain deterministic approximation algorithms.

We start with the problem definition.

Problem	<Star	t_i>
Given:	A star network with arbitrary edge capacities and a set $\mathcal{R}$ of calls. Each call is specified by its source and destination node, and is associated with a starting time, a duration, a positive bandwidth requirement, and a profit.
Goal:	Compute a feasible subset $Q \subseteq \mathcal{R}$ of maximum total profit.

A star network consists of a set of outer nodes, each of which is connected exclusively to a unique center node. It is modeled by a simple, undirected graph $G = (V, E)$, whose vertex set $V = \{0, 1, \ldots, n\}$ represents the nodes of the network with vertex 0 being the center node. The edge set E consists of the edges $e_i = \{i, 0\}$, for $i = 1, \ldots, n$, corresponding to the links in the star network that connect an outer vertex i to the center node 0. The capacities of the links are given as a capacity function $c : E \to \mathbf{N}$ that maps each edge $e \in E$ to its positive capacity $c(e)$.

A call i is specified by a tuple $(s_i, u_i, b_i, t_i, d_i, w_i)$ consisting of a source node $s_i \in V$, a destination node $u_i \in V$, a positive bandwidth requirement $b_i \in \mathbf{R}^+$, a starting time $t_i \in \mathbf{N}$, a duration $d_i \in \mathbf{N}$, and a profit $w_i \in \mathbf{R}$. In stars, there exists a unique path between the source node s_i and the destination node u_i, and we denote the set of (at most 2) edges on this path by *path*(i).

In this section we present an 18-approximation algorithm for <Star | t_i>, that is an algorithm for a given star network and a set of calls that runs in polynomial time and always outputs a feasible solution whose profit is at least a $1/18$ fraction of the profit of an optimal solution.

First, we make sure that every call uses exactly two edges. For every outer vertex $i \in \{1, \ldots, n\}$ we introduce a new vertex i' and connect it to the center node by the new edge $\{i', 0\}$ with the capacity $c(\{i, 0\})$ of the original edge from vertex i to the center vertex. Then we reroute all calls using only one edge $e = \{i, 0\}$ to use both edges e and $e' = \{i', 0\}$. Since the edges e and e' have the same capacity and e' is a new edge, this modification does not affect the feasibility of solutions. Now all calls in the set $\mathcal{R}$ use two edges.

We partition the calls in $\mathcal{R}$ into three classes according to their bandwidth requirements. Consider a call $i \in \mathcal{R}$ and denote by e and f the edges that it uses. We classify the call i to be

- a *small* call, if it uses at most half of the capacity on both of its edges, i.e. $b_i \leq c(e)/2$ and $b_i \leq c(f)/2$,
- a *big* call, if it uses more than half of the capacity on both of its edges, i.e. $b_i > c(e)/2$ and $b_i > c(f)/2$, and
- a *mixed* call, otherwise.

We denote the set of small, big, and mixed calls by $\mathcal{R}_{small}$, $\mathcal{R}_{big}$, and $\mathcal{R}_{mixed}$, respectively.

We give a constant-factor approximation for each of the three classes implying a constant-factor approximation for the original problem.

The algorithm for the original problem is then as follows. Partition the calls in $\mathcal{R}$ into the three classes and solve the problem for each of the classes separately. Output the solution with the largest profit among the three solutions for the classes.

Small calls

We approximate this set using the unified algorithm of Section 5.2. Let i be the call with minimum end-time $t_i + d_i$ selected by the algorithm in step 3 and denote by e and f the edges that call i uses, such that $c(e) \leq c(f)$. We decompose the profit function $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$ by defining $\mathbf{w}'$ to be

$$w'_j = w_i \cdot \begin{cases} 1 & \text{if } j = i, \\ \alpha \cdot (c(f) - b_i) \cdot b_j & \text{if } i \text{ and } j \text{ intersect on edge } e \\ & \text{at time } t_i + d_i, \\ \alpha \cdot (c(e) - b_i) \cdot b_j & \text{if } i \text{ and } j \text{ intersect only on edge } f \\ & \text{at time } t_i + d_i, \\ 0 & \text{otherwise,} \end{cases}$$

where α is a number that will be determined later. Note that $w''_i = 0$. The values of the profit function $\mathbf{w}'' = \mathbf{w} - \mathbf{w}'$ may be non-positive.

Lemma 5.3 *The set of small calls admits an approximation ratio of* 4.

Proof. Due to Lemma 5.2 it suffices to show that every i-maximal solution is an 4-approximation with respect to $\mathbf{w}'$. Let Q^* be a $\mathbf{w}'$-optimal solution. In case i is not in Q^*, the contribution of other calls in Q^* using edge e is at most $w_i \cdot \alpha \cdot (c(f) - b_i) \cdot c(e)$, since the total bandwidth of all these calls intersecting at time $t_i + d_i$ is at most $c(e)$. Likewise the contribution on edge f is at most $w_i \cdot \alpha \cdot (c(e) - b_i) \cdot c(f)$.

In case i belongs to Q^*, the call i adds w_i to the $\mathbf{w}'$-profit of Q^*, and uses b_i of the capacity of the edges e and f. The contribution of other calls in Q^* to the $\mathbf{w}'$-profit is then at most $w_i \cdot \alpha \cdot (c(f) - b_i) \cdot (c(e) - b_i)$ on edge e, and $w_i \cdot \alpha \cdot (c(e) - b_i) \cdot (c(f) - b_i)$ on edge f.

If we set $X := (c(f) - b_i) \cdot (c(e) - b_i)$, the $\mathbf{w}'$-profit of Q^* is at most

$$w_i \cdot \max\{\alpha \cdot (c(f) - b_i) \cdot c(e) + \alpha \cdot (c(e) - b_i) \cdot c(f), 1 + 2 \cdot \alpha \cdot X\}.$$

To derive a lower bound on the $\mathbf{w}'$-profit of an i-maximal solution Q, we distinguish two cases. If the call i is in Q, it contributes w_i itself. Thus the $\mathbf{w}'$-profit of Q is at least w_i. Otherwise, the call i is blocked by other calls in Q. This means that the total bandwidth of the calls preventing call i from being accepted must be greater than $c(e) - b_i$ on edge e or greater than $c(f) - b_i$ on edge f, respectively. Hence, the $\mathbf{w}'$-profit of the blocking calls is at least $w_i \cdot \alpha \cdot X$ (no matter on which of the edges e or f the call i is blocked). The minimum $w_i \cdot \min\{1, \alpha \cdot X\}$ of both expressions is a lower bound on the $\mathbf{w}'$-value of every i-maximal solution.

Altogether, the approximation ratio r is given by

$$r = \frac{\max\{\alpha \cdot (c(f) - b_i) \cdot c(e) + \alpha \cdot (c(e) - b_i) \cdot c(f), 1 + 2 \cdot \alpha \cdot X\}}{\min\{1, \alpha \cdot X\}},$$

which for

$$\alpha = \frac{1}{X} = \frac{1}{(c(f) - b_i) \cdot (c(e) - b_i)}$$

evaluates to

$$r = \max\left\{\frac{c(e)}{c(e) - b_i} + \frac{c(f)}{c(f) - b_i}, 3\right\}.$$

Finally, the fact that i is a small call, i.e. $b_i \leq c(e)/2$ and $b_i \leq c(f)/2$, implies that r is at most 4. □

Big calls

In the case of big calls, no two calls using the same edge simultaneously can be in a feasible solution. Therefore we may assume that $b_j = 1$ for all big calls j and that $c(e) = 1$ for all edges $e \in E$.

To define the decomposition $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$, we set $\mathbf{w}'$ to be

$$w'_j = w_i \cdot \begin{cases} 1 & \text{if } j = i, \\ \alpha & \text{if } i \text{ and } j \text{ intersect at time } t_i + d_i, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 5.4 *The set of big calls admits an approximation ratio of* 2.

Proof. The value of a $\mathbf{w}'$-optimal solution is w_i, if it contains the call i, and at most $2\alpha \cdot w_i$ otherwise, because the calls intersecting i at time $t_i + d_i$ can contribute at most $\alpha \cdot w_i$ on each of the two edges of the call i.

On the other hand, the $\mathbf{w}'$-value of the solution produced by the unified algorithm (and of every i-maximal solution) is at least w_i, if the call i is accepted in that solution, or at least $\alpha \cdot w_i$, if the call i is blocked by some call intersecting call i at time $t_i + d_i$.

Hence, the approximation ratio for the big calls is given by

$$r = \frac{\max\{1, 2\alpha\}}{\min\{1, \alpha\}},$$

which, for $\alpha = 1$, equals 2. □

Mixed calls

Mixed calls use at most half of the capacity of one of their two edges, and need more than half of the capacity of the other edge. They cause the logarithmic factor in the approximation ratio shown by Erlebach in [35].

Here, we present a randomized procedure to approximate the mixed calls within a constant factor of the optimum. Our method can easily be derandomized as discussed below.

We perform the following random experiment on the star network. Every outer node $v \in \{1, \ldots, n\}$ is put independently with probability $1/2$ to a set A and with probability $1/2$ to a set B.

In the following we consider only those mixed calls that have one endpoint in the set A and the other endpoint in the set B. We denote this set of calls by $\mathcal{R}_{A,B}$. The probability that a mixed call belongs to $\mathcal{R}_{A,B}$ is exactly $1/2$. This implies that the expected profit of an optimal solution for the set $\mathcal{R}_{A,B}$ is at least half the profit of an optimal solution for all mixed calls. Hence, we lose a factor of 2 in expectation.

The advantage of this random experiment is that all the calls in $\mathcal{R}_{A,B}$ connect a vertex in A with a vertex in B, and we have the following well-defined ordering on the edges used by the calls in $\mathcal{R}_{A,B}$. For a call $i \in \mathcal{R}_{A,B}$, let $e(i)$ denote the edge of the path of i connecting its endpoint in A with the center node 0, and let $f(i)$ denote the edge of the path between the center node and its endpoint in B.

We further partition the calls in $\mathcal{R}_{A,B}$ according to their bandwidths. If a mixed call $i \in \mathcal{R}_{A,B}$ uses more than half of the capacity of the edge $e(i)$ (and at most half of the capacity of the edge $f(i)$), we put the call i to the set $\mathcal{R}_A$. Otherwise, namely if the call i uses at most half of the capacity of the edge $e(i)$ (and more than half of the capacity of the edge $f(i)$), we put the call i to the set $\mathcal{R}_B$.

We claim the following.

Lemma 5.5 *Each of the sets $\mathcal{R}_A$ and $\mathcal{R}_B$ admits an approximation ratio of* 3.

Proof. To approximate the sets $\mathcal{R}_A$ and $\mathcal{R}_B$ we again use the unified algorithm. For the set $\mathcal{R}_A$ we decompose $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$ by specifying $\mathbf{w}'$ to be

$$w'_j = w_i \cdot \begin{cases} 1 & \text{if } j = i, \\ \alpha/2 & \text{if } i \text{ and } j \text{ intersect on edge } e(i) \text{ at time } t_i + d_i, \\ \alpha \cdot \frac{b_j}{c(f(i))} & \text{if } i \text{ and } j \text{ intersect only on edge } f(i) \text{ at time } t_i + d_i, \\ 0 & \text{otherwise.} \end{cases}$$

Remember that all calls in $\mathcal{R}_A$ that use the edge $e(i)$ need more than half of the capacity of the edge $e(i)$.

Therefore, a $\mathbf{w}'$-optimal solution accepts on edge $e(i)$ either the call i itself with $\mathbf{w}'$-profit w_i or at most one other call that intersects call i on the edge $e(i)$ and gives profit $w_i \cdot \alpha/2$. On the edge $f(i)$ the total bandwidth of accepted calls is bounded by $c(f(i))$ yielding a bound of $w_i \cdot \alpha$ on the optimal $\mathbf{w}'$-profit on that edge.

An i-maximal solution either contains the call i with $\mathbf{w}'$-profit w_i or the call i is blocked by calls with profit at least $w_i \cdot \alpha/2$ (on either edge $e(i)$ or $f(i)$).

Hence, the approximation ratio r is given by

$$r = \frac{\max\{1, \alpha/2\} + \alpha}{\min\{1, \alpha/2\}},$$

which, for $\alpha = 2$, gives $r = 3$.

Because of symmetry reasons, the set $\mathcal{R}_B$ can be approximated with the same ratio by just exchanging the middle two cases in the decomposition of the profit function. □

After approximating both sets $\mathcal{R}_A$ and $\mathcal{R}_B$ by the unified algorithm, we output the solution that has a larger profit.

Corollary 5.6 *The set of mixed calls admits an approximation ratio of* 12.

Proof. The larger solution for the sets $\mathcal{R}_A$ and $\mathcal{R}_B$ has profit at least $1/6$ times the profit of an optimal solution for the set $\mathcal{R}_{A,B}$, which in expectation, is a factor $1/2$ away from the optimal solution for all mixed calls. □

This yields

Theorem 5.7 *The above algorithm is an* 18*-approximation algorithm for the problem* $<$*Star* $|\ t_i>$.

Proof. For a set $Q \subseteq \mathcal{R}$ of calls we denote by $\mathit{OPT}(Q)$ the profit of an optimal solution for the set Q, and by $\mathit{ALG}(Q)$ the profit of the solution produced by the algorithm described above on the input Q.

Let *OPT*= $\mathit{OPT}(\mathcal{R})$ be the optimal profit of a feasible solution for all calls in $\mathcal{R}$, and let $\mathit{OPT}|_{\mathit{small}}$, $\mathit{OPT}|_{\mathit{big}}$ and $\mathit{OPT}|_{\mathit{mixed}}$ denote the restrictions of the optimal profit to the set of small calls, big calls, and mixed calls, respectively. Thus, we have $\mathit{OPT} = \mathit{OPT}|_{\mathit{small}} + \mathit{OPT}|_{\mathit{big}} + \mathit{OPT}|_{\mathit{mixed}}$, and one of the following cases holds true. Either $\mathit{OPT}|_{\mathit{small}} \geq 2/9 \cdot \mathit{OPT}$, or $\mathit{OPT}|_{\mathit{big}} \geq 1/9 \cdot \mathit{OPT}$, or $\mathit{OPT}|_{\mathit{mixed}} \geq 2/3 \cdot \mathit{OPT}$. Otherwise the restricted profits would not add up to *OPT*.

Case 1: $\mathit{OPT}|_{\mathit{small}} \geq 2/9 \cdot \mathit{OPT}$.

It holds that

$$\begin{aligned} 2/9 \cdot \mathit{OPT} &\leq \mathit{OPT}|_{\mathit{small}} \\ &\leq \mathit{OPT}(\mathcal{R}_{\mathit{small}}) \\ &\leq 4 \cdot \mathit{ALG}(\mathcal{R}_{\mathit{small}}) \qquad \text{(by Lemma 5.3)} \\ &\leq 4 \cdot \mathit{ALG}(\mathcal{R}). \end{aligned}$$

Hence, the algorithm is a 18-approximation in this case.

Case 2: $OPT|_{big} \geq 1/9 \cdot OPT$.

In this case, we get a similar chain of inequalities, namely $1/9 \cdot OPT \leq OPT|_{big} \leq OPT(\mathcal{R}_{big}) \leq 2 \cdot ALG(\mathcal{R}_{big}) \leq 2 \cdot ALG(\mathcal{R})$, where the second last inequality is given by Lemma 5.4, and the last inequality is true, because the algorithm outputs the solution with largest profit. Thus, the algorithm is a 18-approximation also in this case.

Case 3: $OPT|_{mixed} \geq 2/3 \cdot OPT$.

We have $2/3 \cdot OPT \leq OPT|_{mixed} \leq OPT(\mathcal{R}_{mixed}) \leq 12 \cdot ALG(\mathcal{R}_{mixed}) \leq 12 \cdot ALG(\mathcal{R})$, leading to the approximation ratio 18 as well.

All cases together prove the theorem. □

5.3.1 Call Alternatives

Our approximation algorithm can be generalized to accommodate alternatives for the calls with only a slight increase of the approximation ratio. The problem we look at is defined as follows.

Problem	<Star | t_i, alt>
Given:	A star network with arbitrary edge capacities and a set $\mathcal{R}$ of calls, where each call i is actually a set $A(i)$ of its call alternatives. Each call alternative is specified by its source and destination node, and is associated with a starting time, a duration, a positive bandwidth requirement, and a profit.
Goal:	Compute a feasible subset $Q \subseteq \bigcup_{i \in \mathcal{R}} A(i)$ of maximum total profit.

Call alternatives allow the specification of several alternatives for establishing a connection request. For example, a connection can be established either from 8am to 11am gaining some profit or from 2pm to 4pm with a possibly different profit. For different alternatives of a call we allow all parameters to be different. A solution obeying the capacity constraints is feasible, if it contains at most one of the alternatives per call. The goal is to find the feasible solution with largest profit.

Similar to the partition of the calls into different classes, we partition the set of all call alternatives into the set of small call alternatives, the set of big call alternatives and the set of mixed call alternatives.

In fact, we the same algorithm as above. The choice of i is now always the call alternative i with minimum end-time $t_i + d_i$ among all call alternatives in question. The decompositions of the profit functions for the different classes of calls are given in the proof below. We have the following lemma.

Lemma 5.8 *Allowing call alternatives the approximation ratios for the set of small call alternatives, big call alternatives, and mixed call alternatives are* 5, 3*, and* 16*, respectively.*

Proof. Let $A(i)$ denotes the set of all alternatives for the call i. For a call alternative i, $k = \mathit{call}(i)$ denotes the call in $\mathcal{R}$ that corresponds to the alternative i, i.e. the call $k \in \mathcal{R}$ with $i \in A(k)$.

Assume that the call alternative i selected in step 3 of the unified algorithm uses the edges e and f with $c(e) \leq c(f)$. Let k be the corresponding call $\mathit{call}(i)$. For small call alternatives the decomposition is given by

$$w'_j = w_i \cdot \begin{cases} 1 & \text{if } j \in A(k), \\ \alpha \cdot \frac{b_j}{c(e)} & \text{if } j \notin A(k) \text{ and } i \text{ and } j \text{ intersect} \\ & \text{on edge } e \text{ at time } t_i + d_i, \\ \alpha \cdot \frac{b_j}{c(f)} & \text{if } j \notin A(k) \text{ and } i \text{ and } j \text{ intersect} \\ & \text{only on edge } f \text{ at time } t_i + d_i, \\ 0 & \text{otherwise.} \end{cases}$$

The value of a $\mathbf{w}'$-optimal solution is at most $w_i + 2\alpha \cdot w_i$, since such a solution can accept at most one call alternative from $A(k)$ contributing w_i and the call alternatives intersecting i on the edges e and f at time $t_i + d_i$ add at most $2\alpha \cdot w_i$.

The profit of an i-maximal solution is at least $\min\{w_i, \alpha/2 \cdot w_i\}$, because the call alternative i is either accepted (contributing w_i) or blocked by other call alternatives. Since i is a small call alternative, the bandwidth needed to block i on either edge e or edge f must be at least half of the capacity of that edge, giving a value of $\alpha/2 \cdot w_i$.

For $\alpha = 2$, the approximation ratio

$$r = \frac{1 + 2\alpha}{\min\{1, \alpha/2\}}$$

equals 5.

For big call alternatives, we can assume as before that in a feasible solution no two call alternatives use the same edge simultaneously. Therefore we may assume that $b_j = 1$ for all big call alternatives j and that $c(e) = 1$ for all edges $e \in E$. We decompose the profit function as follows.

$$w'_j = w_i \cdot \begin{cases} 1 & \text{if } j \in A(k), \\ \alpha & \text{if } j \notin A(k) \text{ and } i \text{ and } j \text{ intersect at time } t_i + d_i, \\ 0 & \text{otherwise.} \end{cases}$$

This yields an upper bound on the optimal $\mathbf{w}'$-profit of $w_i + 2\alpha \cdot w_i$, whereas every i-maximal solution has a $\mathbf{w}'$-profit of at least $\min\{w_i, \alpha \cdot w_i\}$. Setting the value α to 1, the approximation ratio

$$r = \frac{1 + 2\alpha}{\min\{1, \alpha\}} = 3$$

is obtained.

For the set of mixed call alternatives we perform the same random experiment as in Section 5.3, which can be derandomized as described in Section 5.3.2. Then we approximate each of the sets $\mathcal{R}_A$ and $\mathcal{R}_B$ within a factor 4 from optimal. Let $e(i)$ and $f(i)$ be defined as above. For the call alternatives in $\mathcal{R}_A$ the decomposition $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$ is given by

$$w'_j = w_i \cdot \begin{cases} 1 & \text{if } j \in A(k), \\ \alpha/2 & \text{if } j \notin A(k) \text{ and } i \text{ and } j \text{ intersect} \\ & \text{on edge } e(i) \text{ at time } t_i + d_i, \\ \alpha \cdot \frac{b_j}{c(f(i))} & \text{if } j \notin A(k) \text{ and } i \text{ and } j \text{ intersect} \\ & \text{only on edge } f(i) \text{ at time } t_i + d_i, \\ 0 & \text{otherwise.} \end{cases}$$

Therewith, the maximum $\mathbf{w}'$-profit that a feasible solution may obtain is at most $w_i + 3/2 \cdot \alpha \cdot w_i$. On the other hand, every i-maximal solution has a value of at least $\min\{w_i, \alpha/2 \cdot w_i\}$.

For $\alpha = 2$, we get an approximation ratio of

$$r = \frac{1 + 3/2 \cdot \alpha}{\min\{1, \alpha/2\}} = 4$$

for the set $\mathcal{R}_A$, and similarly also for the set $\mathcal{R}_B$. By taking the larger of both solutions, we are at most a factor 8 from optimal. In addition to that, we lost a factor of 2 when restricting the set $\mathcal{R}_{mixed}$ of all mixed call alternatives to the set $\mathcal{R}_{A,B}$. Hence, the overall approximation ratio for the mixed call alternatives is 16, and the lemma follows. □

Theorem 5.9 *The call admission control problem* $<$*Star* $|\, t_i$*, alt*$>$ *admits an approximation ratio of* 24*.*

Proof. The proof of this theorem is analogous to the proof of Theorem 5.7. Let *OPT*= $OPT(\mathcal{R})$ be the optimal profit of a feasible solution for all calls in $\mathcal{R}$, and let $OPT|_{small}$, $OPT|_{big}$ and $OPT|_{mixed}$ denote the restrictions of the optimal profit to the set of small call alternatives, big call alternatives, and mixed call alternatives, respectively. Thus, we have $OPT = OPT|_{small} + OPT|_{big} + OPT|_{mixed}$, and one of the following cases holds true. Either $OPT|_{small} \geq 5/24 \cdot OPT$, or $OPT|_{big} \geq 1/8 \cdot OPT$, or $OPT|_{mixed} \geq 2/3 \cdot OPT$. Otherwise the restricted profits would not add up to *OPT*.

Case 1: $OPT|_{small} \geq 5/24 \cdot OPT$.

It holds that

$$\begin{aligned} 5/24 \cdot OPT &\leq OPT|_{small} \\ &\leq OPT(\mathcal{R}_{small}) \\ &\leq 5 \cdot ALG(\mathcal{R}_{small}) \qquad \text{(by Lemma 5.8)} \\ &\leq 5 \cdot ALG(\mathcal{R}). \end{aligned}$$

Hence, the algorithm is a 24-approximation in this case.

Case 2: $OPT|_{big} \geq 1/8 \cdot OPT$.

In this case, we get a similar chain of inequalities, namely $1/8 \cdot OPT \leq OPT|_{big} \leq OPT(\mathcal{R}_{big}) \leq 3 \cdot ALG(\mathcal{R}_{big}) \leq 3 \cdot ALG(\mathcal{R})$, where the second last inequality is given by Lemma 5.8. Thus, the algorithm is a 24-approximation also in this case.

Case 3: $OPT|_{mixed} \geq 2/3 \cdot OPT$.

We have $2/3 \cdot OPT \leq OPT|_{mixed} \leq OPT(\mathcal{R}_{mixed}) \leq 16 \cdot ALG(\mathcal{R}_{mixed}) \leq 16 \cdot ALG(\mathcal{R})$, leading to the approximation ratio 24 as well.

We have a 24-approximation in all cases, thus proving the theorem. □

5.3.2 Derandomization

In this section we show how the random process for partitioning the mixed calls can be derandomized by reducing the size of the sample space. We refer to [8, Chapter 15] for an overview of this topic.

Interestingly, the standard derandomization method based on conditional probabilities is not applicable to our problem. This method considers the nodes in turn. When processing node i, it computes the conditional expectations for the remaining nodes if the node i is put into the set A and likewise if the node i is put into the set B. The larger of these two conditional expectations drives the decision where the vertex i is put to.

We are looking for a partition of the set of outer vertices into two sets A and B such that at least half of the profit of an optimal solution for all mixed calls is contributed by calls that have one endpoint in A and the other endpoint in B. In a different perspective, we can interpret an optimal solution Q of calls as a weighted graph $G = (V, E, w)$ as follows. The vertex set V is the set $\{1, \ldots, n\}$ of outer vertices of the star. Two vertices are joined by an edge in E, if Q contains a call connecting these two vertices. The weight $w(e)$ of an edge $e = \{u, v\}$ is the sum of profits of calls in Q that connect the vertices u and v. The desired partition is then a cut (A, B) in the graph G whose weight is at least $1/2 \cdot \sum_{e \in E} w(e)$, where the weight of a cut is given by the sum of the weights of the edges that have one endpoint in each A and B. Thus, our problem is essentially that of finding a cut with large weight in an unknown graph G, since the optimal solution Q defining G is not known, and computing it is an $\mathcal{NP}$-hard task. Hence, there is no way of computing the conditional expectations in question.

From this viewpoint of cuts, we present a method of specifying at most $2n-1$ cuts in such a way that at least one of them has the desired property no matter how the optimal solution Q looks like.

In the analysis of our randomized algorithm, we used two properties of our random assignment. Firstly, the probability that a node v is assigned to the set A is $1/2$, and secondly the pairwise independence of the events, which guarantees that each call "survives" the experiment (i.e. its two endpoints are put into different sets) with probability $1/2$.

We will now construct a polynomial-size sample space that preserves these two properties. If we choose assignments uniformly at random from this sample space, our previous analysis remains valid. Thus, if we exhaustively search the sample space, we are guaranteed to find an assignment of nodes to the sets A and B such that the profit of an optimal solution for calls in $\mathcal{R}_{A,B}$ is at least half the profit of an optimal solution for all mixed calls.

As a sample space we will consider partitions induced by so-called Hadamard matrices. A Hadamard matrix H_λ of order λ is a $\lambda \times \lambda$ matrix with $+1$ and -1 elements such that $H_\lambda \cdot H_\lambda^T = \lambda I_\lambda$, where I_λ is the identity matrix of order λ. This implies that its rows and its columns are pairwise orthogonal. Unless $\lambda = 1$ the order of a Hadamard matrix is even and there is a simple construction of Hadamard matrices whose orders are powers of 2 due to Sylvester [81], which, in general, works as follows. If H_μ and $H_\nu = (h_{ij})$ are Hadamard matrices of orders μ and ν, respectively, then the matrix obtained by replacing each $h_{ij} = 1$ with H_μ and each $h_{ij} = -1$ with $-H_\mu$ is a Hadamard matrix of order $\mu\nu$. Starting with the Hadamard matrix

$$H_\nu = H_2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix},$$

we recursively obtain the Hadamard matrices

$$H_{2^k} = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix}$$

of order 2^k for $k > 0$. For example, we have

$$H_4 = \begin{pmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{pmatrix} = \begin{pmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{pmatrix}.$$

Assume that n is of the form $n = 2^k - 1$ for some $k > 0$. If not, round it up to the next such number. Each column of the Hadamard matrix H_{2^k} defines a partition of the nodes to the sets A and B as follows. Let $a_1, \ldots, a_n$ denote the elements of some column of H_{2^k}, excluding the top element of that column. That is, if we number the rows of H_{2^k} beginning with 0, a_1 is always taken from row 1, a_2 is always taken from row 2, an so on. The partition corresponding to that column puts the vertex i to the set A, if $a_i = +1$, and to the set B, if $a_i = -1$.

It is somehow common knowledge that the partitions according to the columns of any Hadamard matrix H ensure the two properties needed in the analysis of our random experiment. Here we prove the properties just for the Hadamard matrices H_{2^k}, for which we have an easy construction.

If we choose a column of H_{2^k} uniformly at random, i.e. each column with probability 2^{-k}, and consider its corresponding partition, then every vertex i

is put to the set A with probability exactly $1/2$. This is true, because every row i, $i = 1, \ldots, n$, contains an equal number of $+1$'s and -1's, which immediately follows from the fact that in H_{2^k} every row i is orthogonal to the row 0 consisting of $+1$ entries only.

For the pairwise independence we need that, if we vertically match two distinct rows, the number of pairs that are $\genfrac{}{}{0pt}{}{+1}{+1}$ or $\genfrac{}{}{0pt}{}{-1}{-1}$ equals the number of pairs that are $\genfrac{}{}{0pt}{}{+1}{-1}$ or $\genfrac{}{}{0pt}{}{-1}{+1}$. If we include all rows of a Hadamard matrix, this follows immediately from the fact that any two rows are orthogonal to each other, since the pairs $\genfrac{}{}{0pt}{}{+1}{+1}$ and $\genfrac{}{}{0pt}{}{-1}{-1}$ contribute 1 to the scalar product of the rows, whereas the pairs $\genfrac{}{}{0pt}{}{+1}{-1}$ and $\genfrac{}{}{0pt}{}{-1}{+1}$ contribute -1. However, the way we defined the partitions, we have to look at any two distinct rows excluding the top row. For the Hadamard matrices H_{2^k}, we actually prove the following stronger statement.

Lemma 5.10 *If we vertically match any two rows (excluding the top row) of H_{2^k}, we have an equal number of $\genfrac{}{}{0pt}{}{+1}{+1}$, $\genfrac{}{}{0pt}{}{+1}{-1}$, $\genfrac{}{}{0pt}{}{-1}{+1}$, and $\genfrac{}{}{0pt}{}{-1}{-1}$ pairs.*

Proof. We will prove this claim using induction on k. In the base case of $k = 2$, each of the pairs appears exactly once, if we look at two rows (excluding the top row) of H_4, which is shown above. In the induction step we assume that each pair of rows, excluding the top row, of $H_{2^{k-1}}$ has an equal number of $\genfrac{}{}{0pt}{}{+1}{+1}$, $\genfrac{}{}{0pt}{}{+1}{-1}$, $\genfrac{}{}{0pt}{}{-1}{+1}$, and $\genfrac{}{}{0pt}{}{-1}{-1}$ pairs. We consider the pairs of rows of

$$H_{2^k} = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix},$$

and distinguish the following cases.

- If both rows are in the top half of H_{2^k}, the claim follows from the induction hypothesis.
- If both rows are in the bottom half of H_{2^k} and neither row is the top row of that half, the claim holds for the left halves of the pair by the induction hypothesis. Since the right halves are inversions of the left halves, the claim also holds for the right halves, and therefore applies to the pair of rows along their entire length.
- If one row is taken from the top half of H_{2^k} and the other is taken from the bottom half, they correspond to different rows in $H_{2^{k-1}}$, and neither row corresponds to the top row of $H_{2^{k-1}}$, the claim holds for the left halves by the induction hypothesis. For the right halves, apply the induction hypothesis to the original rows and replace $\genfrac{}{}{0pt}{}{+1}{+1}$ with $\genfrac{}{}{0pt}{}{+1}{-1}$, $\genfrac{}{}{0pt}{}{+1}{-1}$ with $\genfrac{}{}{0pt}{}{+1}{+1}$, $\genfrac{}{}{0pt}{}{-1}{+1}$ with $\genfrac{}{}{0pt}{}{-1}{-1}$, and $\genfrac{}{}{0pt}{}{-1}{-1}$ with $\genfrac{}{}{0pt}{}{-1}{+1}$. Again, this proves the claim for the pair of rows along their entire length.

- If one row is taken from the top half of H_{2^k} and the other row is taken from the bottom half and they correspond to the same row in $H_{2^{k-1}}$, we find exactly 2^{k-2} $\genfrac{}{}{0pt}{}{+1}{+1}$ and 2^{k-2} $\genfrac{}{}{0pt}{}{-1}{-1}$ pairs in the left halves of the rows and 2^{k-2} $\genfrac{}{}{0pt}{}{+1}{-1}$ and 2^{k-2} $\genfrac{}{}{0pt}{}{-1}{+1}$ pairs in the right halves of the rows.

- If one of the rows is in the bottom half of H_{2^k} and corresponds to the top row of $H_{2^{k-1}}$, we find $+1$ matched with an equal number of $+1$'s and -1's in the left halves and -1 matched with an equal number of $+1$'s and -1's in the right halves of the rows.

The claim follows. □

Thus, if we choose a column of H_{2^k} uniformly at random, the probability that a call "survives" (i.e. its two endpoints are put into different sets) is $1/2$.

Altogether, this guarantees the existence of a partition such that the value of an optimal solution for the calls in $\mathcal{R}_{A,B}$ is at least half of the optimal profit for all mixed calls. The new sample space contains less than $2n$ possibilities. Trying out all of them incurs an additional factor of $O(n)$ in the running time.

5.4 $\mathcal{APX}$-Hardness

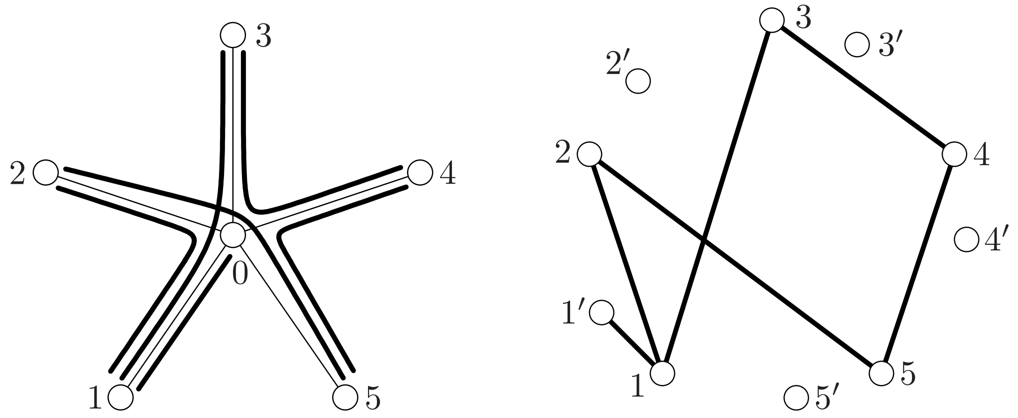
In this section we prove that already a very restricted variant of call admission control in star networks is $\mathcal{APX}$-hard. In Section 5.4.1 we provide an alternative proof showing $\mathcal{NP}$-hardness for a kindred variant.

The restricted problem variant we consider is the following.

Problem	$<\text{Star}, c_i = 1 \mid b_i = 1, t_i \in \{0,1,2\}, d_i = 2, w_i = 1>$
Given:	A star network with unit edge capacities and a set $\mathcal{R}$ of calls. Each call $i \in \mathcal{R}$ has a starting time $t_i \in \{0,1,2\}$, a duration $d_i = 2$, unit profit, and asks for one unit of bandwidth on its edges.
Goal:	Compute a feasible subset $Q \subseteq \mathcal{R}$ of maximum cardinality.

In the following we refer to this problem as *restricted variant* of CAC in stars, because many parameters are fixed to be small constants. A set $Q \subseteq \mathcal{R}$ is a feasible solution for the restricted variant, if at any time at most one call per edge is active.

We allow three different starting times because the cases with one or two different starting times can be solved optimally in polynomial time. To see this,



(a) A star with calls shown as paths.

(b) The corresponding matching graph representation.

Figure 5.1: *Two equivalent representations. The time intervals are omitted.*

we switch to an equivalent representation, which we call the *matching graph* representation. We transform the star network G together with the set $\mathcal{R}$ of calls into the *matching graph* $G' = (V', E')$ representing the same instance. In G' we double every outer vertex of G, i.e. V' contains two vertices v and v' for every outer vertex $v \in V \setminus \{0\}$. The calls in $\mathcal{R}$ are represented by edges in G' in the following way. For every call $i \in \mathcal{R}$ from an outer vertex s_i to an outer vertex u_i in G, we add the edge $\{s_i, u_i\}$ to E'. We still have to deal with the requests ending at the center vertex 0. If $i \in \mathcal{R}$ is such a call from the center vertex 0 to some outer vertex $v \in V \setminus \{0\}$, we represent it by the edge $\{v, v'\}$ in E'. Furthermore, all edges in E' are associated with the time intervals of the calls they represent. Note that the matching graph G' is not necessarily a simple graph, i.e., it may have parallel edges. See Figure 5.1 for an example of both representations. The problem of accepting calls in the star network is reduced to the selection of edges in the matching graph G'. Recall the feasibility condition for a set of calls. A set $Q \subseteq \mathcal{R}$ is a feasible solution for the problem, if at any time, at most one path per edge is active. In the terminology of the matching graph G', a subset $F \subseteq E'$ of edges is feasible, if the corresponding set Q of paths forms a feasible solution in G. That is, at any point in time, every vertex in V' is contained in at most one edge in F that is active at that time.

This leads to the following.

Proposition 5.11 *If all calls in $\mathcal{R}$ overlap in time (or have no time specification), then a maximum matching in the matching graph G' solves the problem optimally. A maximum matching can be computed in polynomial time.*

Proof. If all calls overlap in time, there is a point t in time, when all edges in E' are active. At time t, a set $F \subseteq E'$ of edges is feasible, if no two edges in F are incident to the same vertex. Thus, we can ignore the time intervals associated with the edges in E', and every matching in G' gives a feasible solution for the problem. The optimal solution is obtained by a maximum matching in G'. If we first replace possible multi-edges by single edges, we can compute a maximum matching in $O(n^{5/2})$ time [33, 72], where n is the number of vertices of that graph. In fact, any graph can be obtained as the matching graph of an instance of our restricted variant. □

Proposition 5.11 settles the case where all paths have the same starting time (or no time specification). If we allow two possible starting times while keeping all durations equal, there are two possible cases: the two time intervals are either disjoint or intersect. If the two time intervals intersect, we can apply Proposition 5.11 and compute a maximum matching at the time when all calls are active. The paths corresponding to the edges in the maximum matching constitute an optimal solution. If the two possible time intervals are disjoint, we compute two maximum matchings M_1 for the set $\mathcal{R}_1$ of requests having the first time interval, and M_2 for the set $\mathcal{R}_2$ of requests with the second time interval. By Proposition 5.11, M_1 and M_2 are optimal solutions for $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively, and their union $M_1 \cup M_2$ forms an optimal solution for that case, since there are no conflicts between calls in M_1 and M_2.

If we allow three or more different starting times, the restricted variant becomes difficult to solve, which is expressed in the next theorem.

Theorem 5.12 *The problem <Star, $c_i = 1 \mid b_i = 1,\ t_i \in \{0,1,2\},\ d_i = 2,\ w_i = 1$> is $\mathcal{APX}$-hard.*

Corollary 5.13 *More general versions of call admission control problems in stars including <Star $\mid t_i$> are $\mathcal{APX}$-hard as well.*

In particular, this theorem implies that there is no polynomial-time approximation scheme for the restricted and its more general variants unless $\mathcal{P} = \mathcal{NP}$. Thus, the constant-factor approximations we presented are best possible (except for the constants maybe) in this sense.

We will prove the theorem by an approximation preserving reduction (AP-reduction), which is defined as follows (see also [11]).

Definition 5.14 (AP-reduction) *Let P_1 and P_2 be two optimization problems in NPO. For a solution y to an instance x of P_i, let $\mathrm{ratio}_{P_i}(x, y)$ denote the ratio between the value of an optimal solution to x and the value of y (or the reciprocal of this ratio, whichever is larger than 1). The problem P_1 is called AP-reducible to P_2 if two functions f and g and a positive constant $\alpha > 1$ with the following properties exist:*

(i) For any instance x of P_1 and for any rational $r > 1$, $f(x, r)$ is an instance of P_2.

(ii) For any instance x of P_1 and for any rational $r > 1$, if there is a solution to x, then there is also a solution to $f(x, r)$.

(iii) For any instance x of P_1, for any rational $r > 1$, and for any solution y to $f(x, r)$, $g(x, y, r)$ is a solution to x.

(iv) f and g are computable in polynomial time for any fixed rational r.

(v) For any instance x of P_1, for any rational $r > 1$, and for any solution y to $f(x, r)$, $\mathrm{ratio}_{P_2}(f(x, r), y) \leq r$ implies $\mathrm{ratio}_{P_1}(x, g(x, y, r)) \leq 1 + \alpha(r - 1)$.

The properties (i)-(iv) ensure that there are polynomial time transformations f and g that map instances of P_1 to instances of P_2 and solutions for instances of P_2 back to solutions for the original instance of P_1, respectively. The heart of the AP-reduction is given by property (v) which intuitively says that an r-approximation algorithm for P_2 implies the existence of a $(1 + \alpha(r - 1))$-approximation algorithm for P_1.

In the sequel we present an AP-reduction from the maximum 3-dimensional matching problem, which is the following optimization problem.

Maximum 3-dimensional matching problem	
Given:	A set $D \subseteq X \times Y \times Z$, where X, Y, and Z are disjoint sets.
Goal:	Find a matching $M \subseteq D$ of maximum cardinality, such that no two elements in M agree in any coordinate.

The maximum 3-dimensional matching problem is $\mathcal{APX}$-complete, even if each of the elements in X, Y and Z may occur in at most three triples in D [59]. We refer to the latter problem as the bounded maximum 3-dimensional matching problem.

In this bounded version of the problem, each triple can intersect at most six other triples implying that the maximum matching contains at least $|D|/7$ triples. Moreover, we have

Lemma 5.15 *There is a greedy procedure that computes a* 3*-approximation for the maximum 3-dimensional matching problem.*

Proof. The following greedy procedure computes a 3-approximation for the bounded maximum 3-dimensional matching problem.

GREEDY MATCHING(D)
  $M = \emptyset$
  while $D \neq \emptyset$ **do**
    Choose an arbitrary triple $d \in D$ and remove all triples from D
    that share a coordinate with d
    $M = M \cup \{d\}$
  return M.

Let M^* be a maximum matching for D. We charge every triple $d \in M^*$ to a triple in the solution M of the greedy algorithm according to the rules:

(1) If $d \in M$, charge d to itself.

(2) If $d \notin M$, charge d to a triple in M that is in conflict with d and was accepted by the greedy algorithm before the triple d was processed.

Now, we bound the number of triples charged to the same triple in M. Let $d' \in M$ be any triple in M. We consider two cases:

Case 1: $d' \in M^*$.

Then d' is charged only once by rule (1). No other triple $d \in M^*$, $d \neq d'$ can be charged to d' by rule (2), because M^* is a matching.

Case 2: $d' \notin M^*$.

Then d' is charged according to rule (2) by triples in M^* that are in conflict with d' and are processed after d' has been processed. Since d' has only three coordinates and M^* is a matching, there can be at most three such triples.

Altogether, every triple in M receives a charge of at most 3 units. This implies that $|M^*| \leq 3 \cdot |M|$, showing that the greedy solution is a 3-approximation.
□

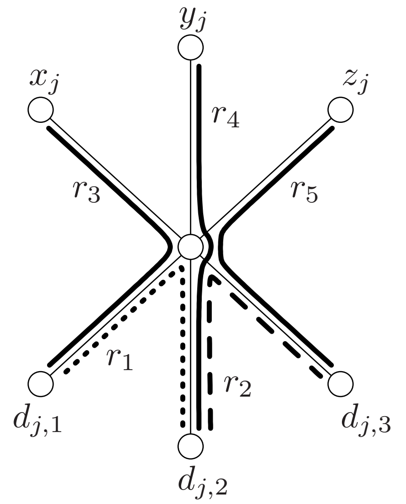


Figure 5.2: *The building block for a triple* $d_j = (x_j, y_j, z_j)$.

Let $D \subseteq X \times Y \times Z$ be an instance of the bounded maximum 3-dimensional matching problem. The function f of the AP-reduction is given by the following construction of an instance of the restricted CAC variant. It will not depend on the parameter r.

Let vertex 0 be the center vertex of the star. For every element $x_i \in X$, we add the vertex x_i to the star and connect it to vertex 0 by an edge $\{x_i, 0\}$. We do the same for every $y_i \in Y$ and for every $z_i \in Z$. For each triple $d_j = (x_j, y_j, z_j) \in D$, we add three more vertices $d_{j,1}$, $d_{j,2}$, and $d_{j,3}$ to the star and connect them to the center by the three edges $\{d_{j,1}, 0\}$, $\{d_{j,2}, 0\}$, and $\{d_{j,3}, 0\}$. In addition, we add the following 5 requests to $\mathcal{R}$ (see Figure 5.2):

- $r_1 = (d_{j,1}, d_{j,2})$ with time interval $[0, 2)$
- $r_2 = (d_{j,2}, d_{j,3})$ with time interval $[2, 4)$
- $r_3 = (d_{j,1}, x_j)$ with time interval $[1, 3)$
- $r_4 = (d_{j,2}, y_j)$ with time interval $[1, 3)$
- $r_5 = (d_{j,3}, z_j)$ with time interval $[1, 3)$

Note that the dotted request r_1 and the dashed request r_2 have disjoint time intervals, whereas the solid requests r_3, r_4, and r_5 do not share any edge. The idea behind this is the following. For every triple $d_i \in D$, a solution either accepts the two request r_1 and r_2 without affecting any edge belonging to the sets X, Y, Z, but blocking the requests r_3, r_4 and r_5, or it accepts the three requests r_3, r_4 and r_5 at the price of blocking all three edges belonging to the triple d_i. More than three requests per triple are not feasible.

Lemma 5.16 *Let $D \subseteq X \times Y \times Z$ be an instance of the bounded maximum 3-dimensional matching problem, and let $(G, \mathcal{R})$ be the corresponding instance of the restricted CAC variant obtained by the above construction. There is a feasible solution for $(G, \mathcal{R})$ that accepts $2|D| + k$ requests, if and only if D has a matching of size k.*

Proof. Suppose there is a feasible solution Q for the instance $(G, \mathcal{R})$ of size $2|D| + k$. Since no more than three requests in Q belong to the same triple, there are at least k triples for which three requests are in Q. The only possibility for one triple $d_i = (x_i, y_i, z_i)$ to have three of its requests accepted, is the choice that accepts the three requests containing the vertices x_i, y_i and z_i. But then, these vertices are blocked for the requests of all other triples. The feasibility of Q implies that all k triples are disjoint. Hence, they form a matching of size k.

Conversely, if there is a matching $M \subseteq D$ of size k, we can construct a feasible solution Q for the instance $(G, \mathcal{R})$ as follows. For every triple $d_i \in M$, put the three requests r_3, r_4 and r_5 into Q. Since the triples in M are disjoint, Q is feasible so far. For each of the remaining triples in $D \setminus M$, we can safely add the two requests r_1 and r_2 to Q without creating any conflict. Thus, Q is feasible by construction, and consists of $2|D| + k$ requests. □

The function g of the AP-reduction takes as arguments the instance D of the bounded maximum 3-dimensional matching problem, a solution Q of the instance $(G, \mathcal{R})$ and the parameter r, which will not be used. It first computes the greedy solution M_1 for the instance D. Secondly, it composes a matching M_2 out of the accepted requests in the solution Q. Whenever all three requests r_3, r_4 and r_5 corresponding to some triple $d_i \in D$ are in Q, it adds the triple d_i to the matching M_2. The value of $g(D, Q, r)$ is given by the larger of the two matchings M_1 and M_2. Thus, $|g(D, Q, r)| = \max\{|M_1|, |M_2|\}$. In addition, it holds $|M_1| \geq |M^*|/3$ by Lemma 5.15, where M^* is a maximum matching for D, and $|M_2| \geq |Q| - 2|D|$ by Lemma 5.16.

Until now the properties (i) – (iv) of the AP-reduction are satisfied, and we will show that property (v) holds with $\alpha = 43$. Therefore, let M^* be a maximum matching for the instance D. Then by Lemma 5.16 an optimal solution Q for $(G, \mathcal{R})$ consists of $|M^*| + 2|D|$ requests. Assume that we have an r-approximation for $(G, \mathcal{R})$, that is a solution Q that contains at least $(|M^*| + 2|D|)/r$ requests.

If $r \geq 45/43$, the inequality $|g(D, Q, r)| \geq |M_1| \geq |M^*|/3$ shows that g computes a 3-approximation. Since $3 = 1 + 43(\frac{45}{43} - 1) \leq 1 + 43(r - 1)$, property (v) with $\alpha = 43$ holds in this case.

Otherwise $r < 45/43$. From $|Q| \geq (|M^*| + 2|D|)/r$, we get

$$\begin{aligned} |Q| &\geq \frac{2r|D| + |M^*| - 2(r-1)|D|}{r} \\ &= 2|D| + \frac{|M^*| - 2(r-1)|D|}{r} \\ &\geq 2|D| + \frac{|M^*|(1 - 14(r-1))}{r}, \end{aligned}$$

where we used $|D| \leq 7|M^*|$ (which holds in the bounded version) in the last inequality. Because

$$\begin{aligned} |g(D, Q, r)| &\geq |M_2| \\ &\geq |Q| - 2|D| \\ &\geq (1 - 14(r-1))|M^*|/r, \end{aligned}$$

we get that the ratio $\mathrm{ratio}_{P_1}(D, g(D, Q, r))$ is at most

$$\begin{aligned} \frac{|M^*|}{(1 - 14(r-1))|M^*|/r} &= \frac{r}{1 - 14(r-1)} \\ &= \frac{r}{15 - 14r} \\ &= 1 + \frac{15}{15 - 14r}(r - 1) \\ &\leq 1 + 43(r - 1), \end{aligned}$$

where the last inequality holds for $1 < r < 45/43$. Again, property (v) is fulfilled with $\alpha = 43$, which completes the proof of the theorem.

5.4.1 An Alternative Proof of Hardness

In this section we give an alternative and weaker proof of hardness for the following CAC variant in star networks.

Problem	<Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$>
Given:	A star network with unit edge capacities and a set $\mathcal{R}$ of calls. Each call $i \in \mathcal{R}$ is associated with an arbitrary starting time and an arbitrary duration, and has unit bandwidth and unit profit.
Goal:	Compute a feasible subset $Q \subseteq \mathcal{R}$ of maximum cardinality.

More precisely, we will prove the problem <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$> to be $\mathcal{NP}$-hard. Of course, this result is an immediate consequence of from Theorem 5.12, but the reduction presented here shows the direct connection between CAC in stars and the satisfiability problem of Boolean formulas.

Theorem 5.17 *The problem <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$> is $\mathcal{NP}$-hard.*

We prove this theorem by a reduction from 3-CNF-SAT, which is the following decision problem.

3-**CNF satisfiability problem** (3-**CNF-SAT**)	
Given:	A Boolean formula ϕ in conjunctive normal form with at most three literals per clause.
Question:	Is there an assignment of truth values to the variables such that ϕ is satisfied?

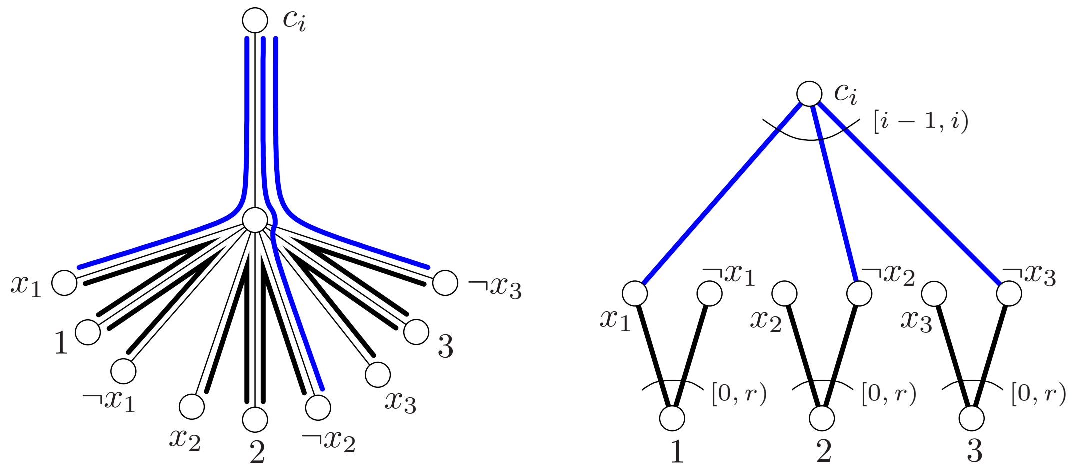
3-CNF-SAT is known to be $\mathcal{NP}$-complete [30].

Suppose we are given a Boolean formula $\phi = c_1 \wedge \ldots \wedge c_r$ in the variables $x_1, \ldots, x_s$. Let $C = \{c_1, \ldots, c_r\}$ denote the set of clauses of ϕ. We now describe how to construct a star network $G = G(\phi)$ and a set of requests $\mathcal{R} = \mathcal{R}(\phi)$ from the given formula ϕ.

Let vertex 0 be the center vertex of the star. For every variable x_i the star network G contains the three vertices i, x_i and $\neg x_i$, all connected to the center vertex by the edges $\{i, 0\}, \{x_i, 0\}$ and $\{\neg x_i, 0\}$. For every clause $c_i \in C$, we add a vertex c_i and connect it to the center by the edge $\{c_i, 0\}$ (see Figure 5.3(a)).

The set $\mathcal{R}$ of requests contains two requests for each variable x_i. One request connects the vertices x_i and i, the second goes from $\neg x_i$ to i. Both requests have the time interval $[0, r)$, where r is the number of clauses.

Additionally, $\mathcal{R}$ contains three requests for each clause $c_i \in C$, each connecting the vertex c_i with the vertex corresponding to a different of the three



(a) The part of $G(\phi)$ showing the requests for the clause $c_i = x_1 \vee \neg x_2 \vee \neg x_3$.

(b) The corresponding matching graph with time intervals for the requests.

Figure 5.3: *The construction of the graph $G(\phi)$ for the Boolean formula ϕ.*

literals of the clause c_i. For example, if the clause $c_i = x_1 \vee \neg x_2 \vee \neg x_3$, we add the requests $(c_i, x_1), (c_i, \neg x_2)$ and $(c_i, \neg x_3)$ to $\mathcal{R}$ as shown in Figure 5.3(a). The time intervals for the requests corresponding to clause c_i are $[i-1, i)$.

Lemma 5.18 *Let ϕ be an instance of* 3-CNF-SAT *with s variables and r clauses, and let $(G, \mathcal{R})$ be the corresponding instance of the problem* <*Star, $c_i = 1 \mid b_i = 1$, t_i, $w_i = 1$*>. *$(G, \mathcal{R})$ has a feasible solution of size $s+r$, if and only if there is a satisfying truth assignment for ϕ.*

For convenience, we represent the instance $(G, \mathcal{R})$ in the matching graph representation $G' = (V', E')$ defined above. V' consists of the vertices i, x_i, and $\neg x_i$ for every variable x_i and the vertices c_i for every clause $c_i \in C$. The two requests for each variable x_i constitute the two edges $\{i, x_i\}$ and $\{i, \neg x_i\}$ in E'. The three clause requests for a clause $c_i \in C$ form three edges in E', each connecting the vertex c_i with a different vertex corresponding to the literals contained in c_i. See Figure 5.3(b) for the matching graph representation of the clause c_i. Note that edges belonging to different clauses have disjoint time intervals, and therefore do not conflict with each other.

Proof (of Lemma 5.18). As observed above, a feasible solution is a subset $F \subseteq E'$ of edges that constitute a matching in G' at any point in time. For every variable x_i, both edges belonging to that variable are incident to the vertex i and have the same time specification. Hence, at most one of them can belong to F. The same is true for the edges belonging to each clause $c_i \in C$, because they all meet at the vertex c_i at the same time. Hence, $s + r$ is an upper bound on the size of any feasible solution F.

Suppose there is a feasible solution $F \subseteq E'$ of size $s + r$. The discussion above shows that s edges of F are incident to the vertices $1, \ldots, s$ belonging to the variables, and r edges of F are incident to the clause vertices $c_1, \ldots, c_r$. Hence, every vertex i for $i = 1, \ldots, s$ is incident to exactly one edge $f_i \in F$. We construct the following assignment of truth values to the variables. We set the variable $x_i =$ FALSE, if f_i connects the vertex i to the vertex x_i, and $x_i =$ TRUE, if f_i connects the vertices i and $\neg x_i$.

It remains to show that this truth assignment satisfies ϕ. We know that every clause vertex c_i for $i = 1, \ldots, r$ is incident to exactly one edge in F, connecting vertex c_i to one of its literals l_i with time interval $[i - 1, i)$. We show that all these literals l_i are TRUE in our truth assignment.

Suppose some l_i is FALSE. According to the construction of the truth values, the vertex l_i must be incident to an edge f_j with time interval $[0, r)$. Thus, we have two edges incident to vertex l_i with overlapping time intervals $[0, r)$ and $[i - 1, i)$ for $1 \leq i \leq r$. But this contradicts the fact that F is a feasible solution. Hence, every clause contains at least one true literal in our truth assignment, and ϕ is satisfiable.

Conversely, assume that the formula ϕ has a satisfying truth assignment $A : x_i \rightarrow \{\text{TRUE}, \text{FALSE}\}$. We construct a feasible solution F in the following way. For every variable x_i, F contains the edge $\{i, x_i\}$, if $A(x_i) =$ FALSE, and the edge $\{i, \neg x_i\}$, if $A(x_i) =$ TRUE. Since A satisfies ϕ, there must be at least one true literal l_i in every clause c_i. Thus, we can add the edge $\{c_i, l_i\}$ for each clause $c_1, \ldots, c_r$ without creating any conflict. Thus, F forms a feasible solution and has size $s + r$. □

This lemma proves the Theorem 5.17. If we could solve the CAC problem <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$> in polynomial time, we could also decide 3-CNF-SAT efficiently, which is not possible unless $\mathcal{P} = \mathcal{NP}$.

Using a similar reduction from 3-CNF-SAT, it is in fact possible to prove $\mathcal{NP}$-hardness for the even more restricted variant, in which all calls have equal durations (<Star, $c_i = 1 \mid b_i = 1, t_i, d_i = k, w_i = 1$>). But then the reduction becomes more complex.

5.5 Constant-factor Approximations for Two Variants

In this section we propose a simple greedy algorithm and prove that it achieves an approximation ratio of 2 for <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$>, which was proved to be $\mathcal{NP}$-hard in Section 5.4.1. Further we apply the unified algorithm described in Section 5.2 to the CAC problem <Star $\mid b_i = 1, t_i$> and obtain a 3-approximation algorithm for this variant.

5.5.1 A Greedy 2-Approximation Algorithm

For good approximate solutions of instances of the call admission control problem <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$>, we analyze the following greedy algorithm, which processes the requests in non-decreasing order of their end-times.

GREEDY-ALGORITHM($\{r_1, \ldots, r_\ell\}$)
  $Q = \emptyset$
  for all requests r_i in order of non-decreasing end-times $t_i + d_i$
    if r_i is not in conflict with requests in Q **then**
      $Q = Q \cup \{r_i\}$ *(accept r_i)*
  return Q.

Theorem 5.19 *The greedy algorithm is a 2-approximation algorithm for <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$>.*

Proof. Let *OPT* be an optimal solution for <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$>. We charge every request $r \in OPT$ to a request in the solution Q of the greedy algorithm according to the rules:

(1) If $r \in Q$, charge r to itself.

(2) If $r \notin Q$, charge r to a request in Q that is in conflict with r and was accepted by the greedy algorithm before request r was processed.

Now, we bound the number of requests charged to the same request in Q. Let $r' \in Q$ be any request in Q. We consider two cases:

Case 1: $r' \in OPT$

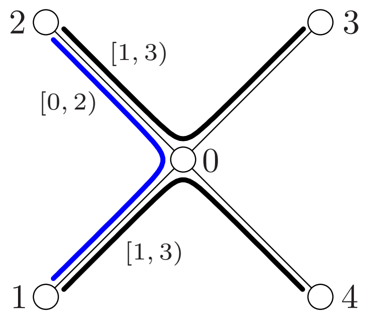


Figure 5.4: *The greedy algorithm selects the path with time interval* $[0, 2)$*, while the optimal solution consists of the two other paths.*

Then r' is charged only once by rule (1). No other request $r \in$ *OPT*, $r \neq r'$ can be charged to r' by rule (2), because *OPT* is a feasible solution.

Case 2: $r' \notin$ *OPT*

Then r' is charged according to rule (2) by requests in *OPT* that are in conflict with r' and are processed after r' has been processed. Since r' contains at most two edges and *OPT* is a feasible solution, there can be at most two such requests.

Altogether, every request in Q receives a charge of at most 2 units. This implies that $|OPT| \leq 2 \cdot |Q|$, showing that the greedy algorithm is a 2-approximation. $\square$

Figure 5.4 shows the small example, where the ratio 2 is actually attained.

5.5.2 A 3-Approximation Algorithm for $<$Star | $b_i = 1, t_i$$>$

We present a 3-approximation algorithm for the variant $<$Star | $b_i = 1, t_i$$>$, which is again based on the local ratio theorem. First we introduce the problem formally.

Problem <Star | $b_i = 1,\, t_i$>

Given: A star network with arbitrary edge capacities and a set $\mathcal{R}$ of calls. Each call is specified by its source and destination node, and is associated with a starting time, a duration, and a profit, and needs one unit of bandwidth.

Goal: Compute a feasible subset $Q \subseteq \mathcal{R}$ of maximum total profit.

As a preprocessing step we ensure that every call uses exactly two edges. This can be accomplished by introducing new nodes and edges to the star as explained in Section 5.3. Then, we approximate this problem using the unified algorithm presented in Section 5.2. In step 3 of the algorithm, we always select the call i with minimum end-time $t_i + d_i$ among all calls in question. Let e and f denote the edges used by the call i such that $c(e) \leq c(f)$. The decomposition of the profit function $\mathbf{w} = \mathbf{w}' + \mathbf{w}''$ is this time given by

$$w'_j = w_i \cdot \begin{cases} 1 & \text{if } j = i, \\ \alpha \cdot c(f) & \text{if } i \text{ and } j \text{ intersect on edge } e \text{ at time } t_i + d_i, \\ \alpha \cdot c(e) & \text{if } i \text{ and } j \text{ intersect only on edge } f \text{ at time } t_i + d_i, \\ 0 & \text{otherwise,} \end{cases}$$

where the parameter α will be determined later.

Theorem 5.20 *Let c_{max} denote the maximum edge capacity of an instance of the problem <Star | $b_i = 1,\, t_i$>. Then, if $c_{max} \leq 2$, there is a 2-approximation algorithm for this instance. If $c_{max} > 2$, the instance can be approximated within factor $(3 \cdot c_{max} - 2)/c_{max} < 3$ from optimal.*

Proof. Due to Lemma 5.2 it suffices to bound the quality of i-maximal solutions with respect to $\mathbf{w}'$. For this purpose let Q^* be a $\mathbf{w}'$-optimal solution. If the call i is not in Q^*, the $\mathbf{w}'$-profit of other calls in Q^* using edge e is at most $w_i \cdot \alpha \cdot c(f) \cdot c(e)$. Likewise the contribution on the edge f is bounded by $w_i \cdot \alpha \cdot c(e) \cdot c(f)$. On the other hand, if the call i belongs to Q^*, it adds w_i to the $\mathbf{w}'$-profit of Q^* and uses one unit of capacity on both edges e and f. Therefore the other calls in Q^* contribute at most $w_i \cdot \alpha \cdot c(f) \cdot (c(e) - 1)$ on the edge e and at most $w_i \cdot \alpha \cdot c(e) \cdot (c(f) - 1)$ on the edge f.

An i-maximal solution Q either contains the call i with $\mathbf{w}'$-profit w_i or the call i is blocked by other calls in Q. In the latter case either the edge e or the edge f must be used to its full extend, since the capacity values are integral and every call requires exactly one unit of bandwidth. Hence, the $\mathbf{w}'$-value of Q is at least $w_i \cdot \alpha \cdot c(f) \cdot c(e)$, if the call i is blocked on the edge e, or $w_i \cdot \alpha \cdot c(e) \cdot c(f)$, if the call i is blocked on the edge f.

If we set $X := c(e) \cdot c(f)$, the approximation ratio r is given by

$$r = \frac{\max\{2\alpha \cdot X, 1 + \alpha \cdot c(f) \cdot (c(e) - 1) + \alpha \cdot c(e) \cdot (c(f) - 1)\}}{\min\{1, \alpha \cdot X\}}.$$

For

$$\alpha = \frac{1}{X} = \frac{1}{c(e) \cdot c(f)},$$

the approximation ratio evaluates to

$$r = \max\left\{2, 1 + \frac{c(e) - 1}{c(e)} + \frac{c(f) - 1}{c(f)}\right\}.$$

Since each of the terms $\frac{c(e)-1}{c(e)}$ and $\frac{c(f)-1}{c(f)}$ is at most $(c_{max} - 1)/c_{max}$, we have

$$r \leq \max\left\{2, \frac{3 \cdot c_{max} - 2}{c_{max}}\right\},$$

which establishes the theorem. □

5.6 Fixed-parameter Tractability

In this section, we consider the problem <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$> of the Sections 5.4.1 and 5.5.1 from the viewpoint of parameterized complexity. Instead of maximizing the number of accepted requests, we equivalently ask for feasible solutions that minimize the number of rejected requests. By taking the number of rejected requests as parameter, we get the following parameterized version of the problem.

Problem	<Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$>–$k$
Given:	A star network with unit edge capacities, a set $\mathcal{R}$ of ℓ calls, and an integer $k > 0$. Each call is associated with a starting time and a duration, needs one unit of bandwidth, and has unit profit.
Goal:	Compute a feasible subset $Q \subseteq \mathcal{R}$ that rejects at most k calls, or decide that no such subset exists.

Theorem 5.21 *The problem <Star,* $c_i = 1 \mid b_i = 1$*,* t_i*,* $w_i = 1$*>–*k *is fixed-parameter tractable (FPT).*

Proof. Let an instance of <Star, $c_i = 1 \mid b_i = 1, t_i, w_i = 1$>–$k$ be given by the triple $(G, \mathcal{R}, k)$, where G denote the graph corresponding to the underlying star network. We are going to define the *conflict graph* H of an instance $(G, \mathcal{R}, k)$. The vertices of H are the calls in $\mathcal{R}$. Two vertices v_i and v_j are connected by an edge in H, if the corresponding calls r_i and r_j are in conflict, i.e., they share an edge of the star at the same time.

According to this definition of the conflict graph H, a feasible subset $Q \subseteq \mathcal{R}$ of calls corresponds to an independent set of vertices in H. This implies that no two vertices of H corresponding to calls in Q span an edge in H. The other way round, every edge of H is incident on at least one vertex corresponding to a call in the complement $\mathcal{R} \setminus Q$. Hence, the vertices corresponding to $\mathcal{R} \setminus Q$ form a vertex cover in H.

Thus checking whether there exists a feasible solution $Q \subseteq \mathcal{R}$ rejecting at most k calls, is equivalent to the question, whether the conflict graph H has a vertex cover of size at most k.

The vertex cover problem is well-known to be fixed-parameter tractable [31]. The currently best algorithm has running time $O(kn + 1.271^k k^2)$ for determining, whether a graph on n vertices has a vertex cover of size at most k [26]. □

This proof does not use the structure of the star network at all. The only properties needed are pre-determined calls and the fact that feasible solutions contain at most one call per edge at any time. Hence, for the same call characteristics, the theorem follows immediately for tree networks, since paths in trees are unique.

Without time specifications for the calls, the parameterized versions of call admission control problems in arbitrary directed or undirected graphs are fixed-parameter tractable, if the edge capacities are bounded by a constant, and the paths are pre-determined [10, Proposition 1]. The proof of this result applies the method of *bounded search trees* [31].

Chapter 6

On-line Coloring of Intervals with Bandwidth

Motivated by resource allocation problems in communication networks, we investigate the problem of on-line interval coloring in the case where the intervals have bandwidth requirements in $(0, 1]$. In a legal coloring the total bandwidth of intersecting intervals with the same color must not exceed 1. The main result of this chapter is an on-line algorithm for this problem that achieves a constant competitive ratio of 195. It is presented in Section 6.3. Our algorithm is a combination of an optimal on-line algorithm for coloring interval graphs [65] and First-Fit coloring, for which we generalize the analysis of Kierstead [62] to the case of non-unit bandwidth.

We start with some words of motivation including some applications in Section 6.1, followed by preliminaries and notations given in Section 6.2 that are needed in the analysis of our algorithms.

6.1 Motivation

On-line coloring of intervals is a classical problem whose investigation has led to a number of interesting insights into the power and limitations of on-line algorithms. Intervals are presented to the on-line algorithm in some externally specified order, and the algorithm must assign each interval a color that is different from the colors of all previously presented intervals intersecting the current interval. The goal is to use as few colors as possible.

We generalize this problem to the case in which the intervals have bandwidth requirements, or simply bandwidths, in $(0, 1]$. Now a set of intervals over the real line can be assigned the same color, if for any point r on the real line, the sum of the bandwidths of its intervals containing r is at most 1. The special case where every interval has bandwidth 1 corresponds to the original on-line interval coloring problem. Our problem is thus a simultaneous generalization of on-line interval coloring and on-line bin-packing. With bin-packing we mean the following optimization problem.

Minimum Bin-Packing Problem	
Given:	A set U of items, a size $s(u) \in \mathbf{N}$ for each $u \in U$, and a positive integer bin capacity B.
Goal:	Find a partition of U into disjoint sets $U_1, \ldots U_K$ such that the sum of sizes of items in each U_i is at most B and the number K of sets is minimized.

The sets are also called bins, because of the interpretation that the elements of each set U_i fit into a bin of size B. The decision problem corresponding to the minimum bin-packing problem is $\mathcal{NP}$-complete in the strong sense [46]. For a given instance of the minimum bin-packing problem, we can create for each item $u_i \in U$ an interval i with bandwidth requirement $s(u_i)/B$. If all these intervals extend from 0 to 1 say (or overlap), the minimum number of colors needed to color them equals the minimum number of bins used for packing the original items optimally. This one-to-one correspondence between the minimum bin-packing problem and the coloring problem for intervals with bandwidths yields that the off-line version of the latter problem is $\mathcal{NP}$-hard. By a nice reduction from partition one can show that the off-line version of interval coloring with bandwidths cannot be approximated within $3/2 - \varepsilon$ for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$ [46].

Besides its theoretical interest, investigating the on-line coloring problem for intervals with bandwidth requirements is motivated by several applications.

First, imagine a communication network with line topology. The bandwidth of each link is partitioned into channels, where each channel has capacity 1. The channels could be different wavelengths in an all-optical WDM (wavelength-division multiplexing) network or different fibers in an optical network supporting SDM (space-division multiplexing), for example. Connection requests with bandwidth requirements arrive on-line, and each request must be assigned to a channel without exceeding the capacity of the channel on any of the links of the connection. We assume that the network nodes do not support switching of traffic from one channel to another channel (which is the case if only add-drop multiplexers are used). Then a connection request from a to b corresponds to an interval $[a, b)$ with the respective bandwidth requirement, and the problem of minimizing the number of required channels to serve all requests is just our on-line coloring problem for intervals with bandwidth requirements.

A related scenario in a line network is that the connection requests have unit durations and the goal is to serve all connections in a schedule of minimum duration. In this case, the colors correspond to time slots, and the total number of colors corresponds to the schedule length. This is a special case of the call scheduling problem considered, for example, in [43, 37].

Finally, an interval could represent a time period during which a job must be processed, and the bandwidth of the interval could represent the fraction of a resource (machine) that the job needs during its execution. At any point in time, a machine can execute jobs whose bandwidths (here, resource requirements) sum up to at most 1. If jobs arrive on-line (before the actual schedule starts) and have to be assigned to a machine immediately, with the goal of using as few machines as possible, we again obtain our on-line interval coloring problem with bandwidth requirements.

6.2 Preliminaries

We consider the following generalization of on-line interval coloring. We are given a collection $\mathcal{I}$ of closed intervals over the real numbers $\mathbf{R}$, where each interval i is associated with a bandwidth requirement $b(i)$, where $0 < b(i) \leq 1$. In an on-line scenario these intervals are processed one by one in some externally determined sequence. Whenever an interval i arrives, the algorithm knows the previously processed intervals and the colors assigned to them, but it knows nothing about the unprocessed intervals arriving in the future. Based on this knowledge the algorithm irrevocably assigns a color $f(i)$ to the interval i, in such a way that for every color x and every point $r \in \mathbf{R}$ the sum of the bandwidths of the intervals containing r that are assigned color x is at most 1. The goal is to use as few colors as possible.

We compare the performance of an on-line algorithm with the number of colors used by an optimal off-line algorithm, denoted by *OPT*. More precisely, we call an on-line algorithm A c-competitive if $A(\mathcal{I}) \leq c \cdot \mathit{OPT}(\mathcal{I})$ for all input sequences $\mathcal{I}$.

Let $\mathcal{I} = \{i_1, \ldots, i_\ell\}$ be the collection of intervals and let $b(i_t)$ denote the

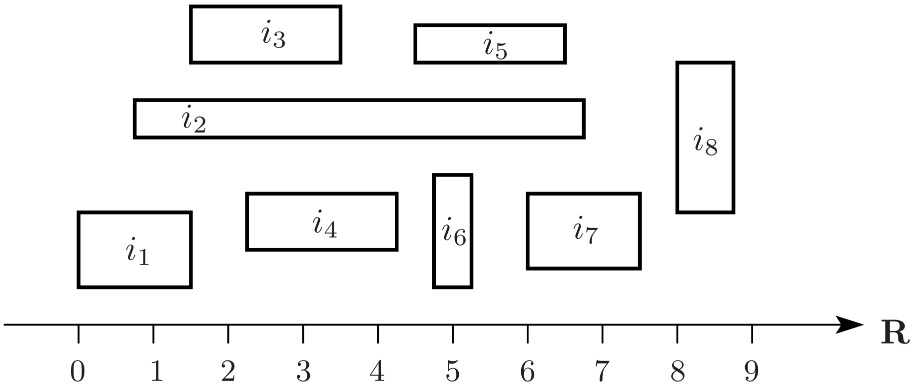


Figure 6.1: *The set $\mathcal{I} = \{i_1, \ldots, i_8\}$ of intervals.*

bandwidth requirement of interval i_t, i.e. $0 < b(i_t) \leq 1$ for $t = 1, \ldots, \ell$. See Figure 6.1 for an example with $\ell = 8$ intervals depicted as rectangles. Their projections onto the real line are exactly the intervals over the real numbers, and the height of a rectangle shows the bandwidth requirement of the corresponding interval. Let $(\mathcal{I}, <)$ be the partial order of the intervals in $\mathcal{I}$, where the relation $i < j$ holds if and only if the right endpoint of interval i is less than the left endpoint of interval j, e.g. we have $i_4 < i_5$ in Figure 6.1. Let $L \subseteq \mathcal{I}$ be a subset of intervals. For an interval $i \in \mathcal{I}$, we write $i < L$ ($L < i$) if for all intervals $j \in L$, $i < j$ ($j < i$) holds. The *neighborhood* of an interval $i \in \mathcal{I}$ is the set of intervals in $\mathcal{I}$ that are different from i and intersect i. It is denoted by $N(i)$. In the example, the neighborhood of interval i_4 is $N(i_4) = \{i_2, i_3\}$.

The *density* $D(r \mid L)$ of L at a point $r \in \mathbf{R}$ is the sum of the bandwidths of the intervals in L that contain the point r, i.e. $D(r \mid L) = \sum_{r \in j \in L} b(j)$. The density $D(i \mid L)$ of an interval $i \in \mathcal{I}$ with respect to L is the minimum density $D(r \mid L)$ over all points r in the interval i, i.e. $D(i \mid L) = \min\{D(r \mid L) : r \in i\}$. With $D(L)$ we denote the maximum density of L, that is $D(L) = \max\{D(r \mid L) : r \in \mathbf{R}\}$. In our example shown in Figure 6.1, the density $D(3 \mid \mathcal{I}) = b(i_2) + b(i_3) + b(i_4)$. The density of the interval i_3 with respect to $\mathcal{I}$ is $D(i_3 \mid \mathcal{I}) = b(i_2) + b(i_3)$ and it is achieved at the point 2 for example. Finally, the maximum density $D(\mathcal{I}) = b(i_2) + b(i_5) + b(i_6)$ is achieved at point 5.

Further, we need some notation for classifying intervals in terms of their left-to-right order $(\mathcal{I}, <)$. A chain C is a sequence of intervals, where $i < j$ holds for any two consecutive intervals i and j in this sequence C. Therewith, we define the *height* $h(i \mid L)$ of an interval i with respect to L to be the length of the longest chain C in L such that $C < i$. Similarly, the *depth* $d(i \mid L)$ is the length of the longest chain C in L such that $i < C$. The *centrality* of an interval i with respect to L is then given by $c(i \mid L) = \min\{h(i \mid L), d(i \mid L)\}$. Looking again at the example presented in Figure 6.1, the height of interval i_4 with respect to $\mathcal{I}$ is $h(i_4 \mid \mathcal{I}) = 1$ since the longest chain consists only of the interval i_1. The corresponding depth $d(i_4 \mid \mathcal{I})$ equals 3 because of the chain $i_6 < i_7 < i_8$ of length 3. Hence, the centrality $c(i_4 \mid \mathcal{I})$ is given by $\min\{1, 3\} = 1$.

Notice that for a subset $L \subseteq N(j)$, if an interval i contains an endpoint of the interval j then $c(i \mid L) = 0$, and if $c(i \mid L) \geq 1$ then the interval i is contained in the interval j, i.e. $i \subset j$.

For our analysis we need two lemmas regarding the existence of an interval with high density in a clique of intervals. The first lemma is a generalization of a lemma that appeared in [84, 54] for the case of interval graphs, i.e. for the setting where all intervals have unit bandwidth.

Lemma 6.1 *Let $\mathcal{I}$ be a collection of intervals. If L is a clique in the intersection graph of $\mathcal{I}$ (meaning that any two intervals in L intersect), then there exists an interval $i \in L$ such that*

$$D(i \mid L) \geq \frac{D(L)}{2} \,.$$

Proof. Let $L = \{i_1, \ldots, i_t\}$. Since the intervals in L form a clique, there is a point on the real line that is contained in every interval in L and we have

$$D(L) = \max\{D(r \mid L) : r \in \mathbf{R}\} = \sum_{m=1}^{t} b(i_m).$$

We build the sequence S of bandwidths of the intervals in L from left to right. Whenever an interval i_m starts or ends, we append its bandwidth $b(i_m)$ to the sequence S. Thus, the length of the sequence S is $2t$ since every bandwidth of intervals in L appears twice, once for the left endpoint and once for the right endpoint. Starting from the beginning of S, let S_1 be the largest initial subsequence of elements in S obeying the condition that their sum is less than $D(L)/2$. Likewise, let S_2 be the largest subsequence starting from the end of S (going backwards) under the condition that the sum of elements in S_2 is less than $D(L)/2$.

Now, we claim that there exists an interval i whose bandwidth appears neither in the subsequence S_1 nor in the subsequence S_2. This is the case, because otherwise all bandwidths of intervals would appear either in S_1 or in S_2, which, in turn, means that their total sum would be at least $D(L)$ contradicting the fact that each of the two subsequences sums up to less than $D(L)/2$.

Every such interval i has a density $D(i \mid L) \geq D(L)/2$. This follows from the construction of the sequences S_1 and S_2 and from the fact that in a left-to-right scan through a clique of intervals all left interval endpoints precede all right interval endpoints. □

Lemma 6.2 *Let L be a clique of intervals each of which is assigned a different color. If every interval in L has density at least $\rho \leq 1$ within its color class, then there exists an interval $i \in L$ such that the density of i with respect to the union of the present color classes is at least $\rho \cdot |L|/2$.*

Proof. This lemma can be proven similar to Lemma 6.1 by building a sequence S from a left-to-right scan of the intervals in L, except that this time we append the value ρ whenever an interval in L starts or ends. But we will prove it by induction on the size of L. In the base cases $|L| = 1$ and $|L| = 2$ any interval in L has the desired property, since $|L|/2 \leq 1$. For the induction step, let j be the interval of L with the smallest left endpoint and let k be the interval of L with the largest right endpoint. If these intervals j and k are identical, the rest of the proof works exactly the same, but only one interval is removed in the inductive step.

Using the induction hypothesis, we choose an interval $i \in L \setminus \{j, k\}$ such that the density of the interval i with respect to the union of color classes of intervals in $L \setminus \{j, k\}$ is at least $\rho \cdot (|L|/2 - 1)$. Since the intervals j and k intersect, the interval i is contained in $j \cup k$, and because the intervals j and k have density ρ within their color classes, the density of the interval i with respect to the union of all present color classes is at least $\rho \cdot |L|/2$. □

Finally, if $N = (n_1, \ldots, n_t)$ is a sequence of length t we denote the initial subsequence of N of length k by $N_k = (n_1, \ldots, n_k)$. The result $(n_1, \ldots, n_t, n)$ of appending n to the end of N is denoted by $N \circ n$.

6.3 The On-line Coloring Algorithm

In this section we first present our on-line algorithm and then analyze its performance in terms of competitive analysis.

Our algorithm works with two sets C_1 and C_2 of disjoint colors. Let $\sigma = \sigma_1\sigma_2\cdots$ be any sequence of intervals. Each σ_i is associated with a bandwidth $b(\sigma_i)$, $0 < b(\sigma_i) \leq 1$. Whenever an interval σ_i arrives, the algorithm processes it according to its bandwidth $b(\sigma_i)$:

If $b(\sigma_i) \leq 1/2$, the algorithm assigns the interval σ_i a color from the set C_1 of colors using the First-Fit principle. The First-Fit assignment is made by coloring the interval σ_i with color α, where α is the smallest color in C_1 such that for every $r \in \sigma_i$, the sum of the bandwidths of intervals previously colored α and containing r is at most $1 - b(\sigma_i)$.

If $b(\sigma_i) > 1/2$, the algorithm assigns the interval σ_i a color from the set C_2 using the optimal on-line interval coloring algorithm of Kierstead and Trotter [65]. Note that the intervals having bandwidth exceeding $1/2$ have the same property as intervals without bandwidth requirement, namely, if two intervals intersect, they have to receive different colors in a proper coloring.

We remark that the partition into two classes is indeed necessary in our approach, because First-Fit can be arbitrarily bad for intervals with bandwidths in $(0, 1]$ as shown in Figure 6.2. Essentially the same example appears also in [43].

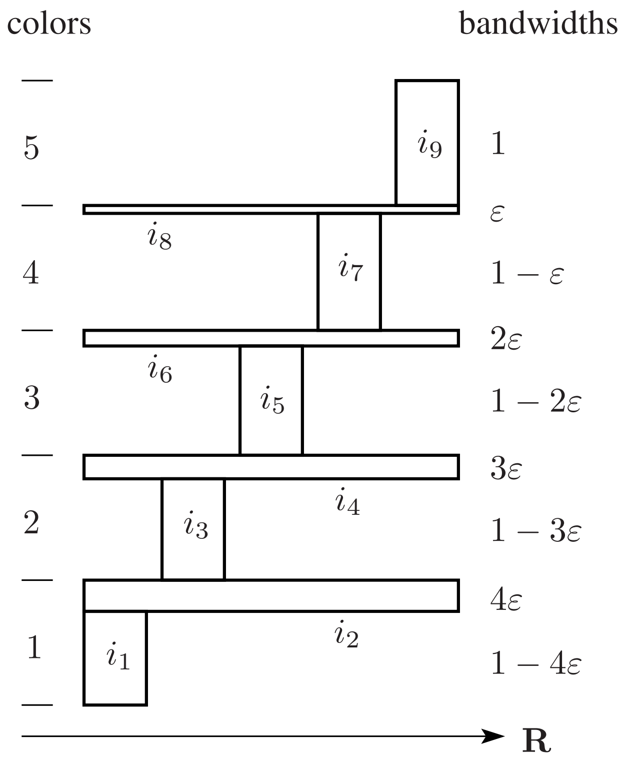
Theorem 6.3 *The algorithm given above solves the on-line problem of coloring intervals with bandwidths using at most* 195 *times as many colors as the optimal off-line algorithm.*

For the proof of this theorem we need two lemmas which we will prove below. Fix a sequence σ and let $\mathcal{I}$ be the set of all intervals appearing in σ. Let $\mathcal{I}_1 \subset \mathcal{I}$ be the intervals with bandwidth at most $1/2$, and let $\mathcal{I}_2 \subset \mathcal{I}$ be the intervals with bandwidth exceeding $1/2$. We have $\mathcal{I} = \mathcal{I}_1 \dot\cup \mathcal{I}_2$.

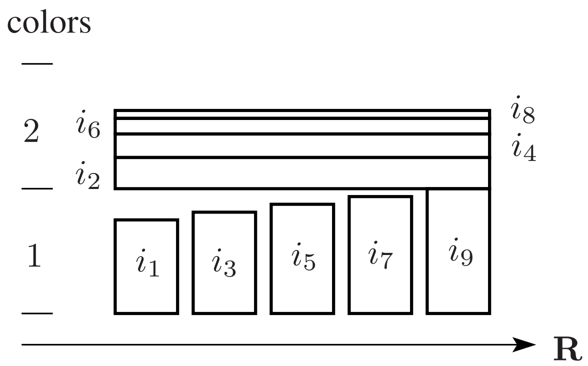
Lemma 6.4 *If the number of colors that the algorithm uses from the set C_1 is at least* 2*, then this number is less than* 192 *times the maximum density $D(\mathcal{I}_1)$ of intervals in $\mathcal{I}_1$.*

Lemma 6.5 *The number of colors from the set C_2 that are used by the algorithm is less than* 3 *times the number of colors used by an optimal off-line algorithm for coloring the intervals in $\mathcal{I}_2$.*

Proof (of Theorem 6.3). Let *OPT* denote the number of colors used by an optimal off-line algorithm. Clearly, this number is at least as large as the maximum density of the intervals, i.e. $D(\mathcal{I}) \leq$ *OPT*. With *OPT*$(\mathcal{I}_1)$ and *OPT*$(\mathcal{I}_2)$ we denote the number of colors assigned by an optimal off-line algorithm to the intervals in $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. Since *OPT*$(\mathcal{I}_1)$ and *OPT*$(\mathcal{I}_2)$ count the number of colors for restricted sets of intervals, *OPT*$(\mathcal{I}_1) \leq$ *OPT* and *OPT*$(\mathcal{I}_2) \leq$ *OPT* hold true.



(a) The First-Fit algorithm requires many colors for the sequence $i_1, i_2, \ldots, i_9$.



(b) The optimal off-line algorithm needs only two colors.

Figure 6.2: *For bandwidth requirements in* $(0, 1]$ *the First-Fit algorithm can perform arbitrarily badly.*

By Lemma 6.4, the algorithm colors the intervals in $\mathcal{I}_1$ using the First-Fit algorithm with less than $192 \cdot OPT(\mathcal{I}_1)$ many colors. This is true, because $OPT(\mathcal{I}_1) \geq D(\mathcal{I}_1)$ and if the First-Fit algorithm needs just one color for coloring the intervals in $\mathcal{I}_1$, then also the optimal off-line algorithm needs one color.

By Lemma 6.5, the algorithm colors the intervals in $\mathcal{I}_2$ using an optimal on-line algorithm for coloring interval graphs with less than 3 times as many colors as an optimal off-line algorithm.

Thus, the number C of colors used by the algorithm is

$$\begin{aligned} C &< 192 \cdot OPT(\mathcal{I}_1) + 3 \cdot OPT(\mathcal{I}_2) \\ &\leq 192 \cdot OPT + 3 \cdot OPT \\ &= 195 \cdot OPT. \end{aligned}$$

Hence, the algorithm is 195-competitive. $\square$

Let us start with the proof of Lemma 6.5.

Proof (of Lemma 6.5). The optimal on-line interval coloring algorithm uses at most $3 \cdot \omega(\mathcal{I}_2) - 2$ colors for coloring the intervals in $\mathcal{I}_2$ [65]. Since the bandwidth of the intervals in $\mathcal{I}_2$ is greater than $1/2$, any two intersecting intervals must receive different colors. Therefore any off-line algorithm needs at least $\omega(\mathcal{I}_2)$ many colors for coloring the intervals in $\mathcal{I}_2$. $\square$

Now we will prove Lemma 6.4, using a rather technical induction.

Proof (of Lemma 6.4). In this proof we generalize a proof of Kierstead, who proved in [62] that First-Fit uses less than $40\,\omega$ colors for interval coloring, i.e. the case where all bandwidths are 1. We extend the proof to the case where all intervals have bandwidth at most $1/2$.

The main idea of the proof is the following: If the First-Fit coloring uses $x \geq 192 \cdot y$ colors, there is a point $r \in \mathbf{R}$ where the density $D(r \mid \mathcal{I}_1)$ exceeds y. For $y \geq D(\mathcal{I}_1)$ we would arrive at a contradiction, since—by definition of $D(\mathcal{I}_1)$—there will not be any point whose density with respect to $\mathcal{I}_1$ exceeds $D(\mathcal{I}_1)$. Therefore the number of colors used by the First-Fit algorithm is $x < 192 \cdot D(\mathcal{I}_1)$.

Let $f(i)$ denote the color that First-Fit assigns to interval i, and let $C(q) = \{i \in \mathcal{I} : f(i) = q\}$ be the set of intervals with color q. For a range of colors we denote by $C[p, q] = \cup_{p \leq r \leq q} C(r)$ all intervals whose colors are within that range.

In order to find a point with high density in $\mathcal{I}_1$, we construct a sequence I of intervals $i_1 \supset \cdots \supset i_t$ and disjoint blocks $B_1, \ldots, B_t$ of intervals such that

$i_m \in B_m$ and the sum of densities $D(i_1 \mid B_1) + \cdots + D(i_m \mid B_m)$ is at least some number s_m, for $m = 1, \ldots, t$. In this construction, every block B_m consists of intervals from the neighborhood of the interval i_{m-1} whose colors are within a certain range. More formally, $B_m = C[x - 192 \cdot r_m + 1, x - 192 \cdot r_{m-1}] \cap N(i_{m-1})$ where $r_m < s_m$, for $m = 1, \ldots, t$. Since the intervals in I form a decreasing sequence, the density of the interval i_t is large, i.e. $D(i_t \mid C[x - 192 \cdot r_t + 1, x]) \geq s_t$. The numbers r_m and s_m will be chosen in such a way that $192 \cdot r_m$ and $192 \cdot s_m$ are integral.

In the induction step we would like to find an interval $i_{t+1} \subset i_t$ with high density in B_{t+1}. However, there may not be any interval with high density in B_{t+1}, or the only intervals with high density in B_{t+1} may overlap i_t. Before we explain below how to deal with these cases, we introduce some more notation.

Let $I = (i_1, \ldots, i_t)$ be a sequence of intervals and let $N = (n_1, \ldots, n_t)$ be a sequence of real numbers, where each n_m is of the form 2^{k-2} for some integer $k \geq 0$. In particular, we have $n_m \geq 1/4$ for all $m = 1, \ldots, t$. The following parameters are defined on the pair (N, I) for $1 \leq m \leq t$:

- Let s_m be the sum of the first m numbers of the sequence N, i.e. $s_m = \sum_{j=1}^{m} n_j$, and set $s_0 = 0$. Note that $4 \cdot s_m$ is integral, since all n_j's are of the form 2^{k-2}.
- Let $r_m = s_{m-1} + n_m/2 < s_m$, and set $r_0 = 0$. Observe that $8 \cdot r_m$ is integral for the same reason as above.
- Let B_m be the disjoint blocks of intervals defined by $B_m = C[x - 192 \cdot r_m + 1, x - 192 \cdot r_{m-1}] \cap N(i_{m-1})$ for $m \geq 1$, where $i_0 = \mathbf{R}$. B_m consists of all intervals in $\mathcal{I}_1$ that intersect the previous interval i_{m-1} of the sequence I and have a color in the given range.
- Let c_m be the centrality, and d_m be the density of the interval i_m within the intervals in B_m, i.e. $c_m = c(i_m \mid B_m)$ and $d_m = D(i_m \mid B_m)$.

We call a pair (N, I) *admissible* if the following conditions hold for $1 \leq m \leq t$:

(1) The interval i_m is contained in the set B_m, i.e. $i_m \in B_m$.

(2) The centrality of the interval i_m with respect to B_m is at least 2, i.e. $c_m \geq 2$, if $m > 1$.

(3) The number $n_{m+1} \geq n_m 4^{2-c_{m+1}}$, if $m < t$.

(4) The density of the interval i_m with respect to B_m is at least n_m, i.e. $d_m \geq n_m$.

Notice that if (N, I) is an admissible pair, then the density of the interval i_t is at least s_t, i.e. $D(i_t \mid \mathcal{I}_1) \geq s_t$. Since the blocks B_m are disjoint, and the intervals i_m are contained in each other by (2), the density of the interval i_t is at least the sum of the densities d_m, which in turn are at least n_m by (4). By definition s_t constitutes exactly this sum.

We shall prove the lemma by first showing that there exists an admissible pair (N, I), and if (N, I) is such an admissible pair and the number x of used colors fulfills $x \geq 192 \cdot s_t$ then there exists an admissible pair (N', I') such that $s'_{t'} > s_t$. Here and below primes indicate elements and parameters of the new admissible pair.

For the basis of the induction we need an admissible pair. Let i be any interval colored x.

If $b(i) = 1/2$, then $N = (1/2)$ and $I = (i)$ is an admissible pair consisting of $t = 1$ interval. Its parameters are as follows: n_1 and therefore s_1 equal $1/2$, and $r_1 = n_1/2 = 1/4$. Since the interval $i_1 = i$ is colored x, it is contained in the block $B_1 = C[x - 48 + 1, x]$. Further its density d_1 in B_1 is at least $1/2 = n_1$, since the interval i_1 itself contributes a bandwidth $b(i_1) = 1/2$. The conditions (2) and (3) do not play a role for $t = 1$.

If $b(i) < 1/2$, then because of First-Fit there is a point $r \in i$, where intervals colored $x - 1$ intersect, whose total bandwidth exceeds $1/2$. Otherwise the interval i would have been colored $x - 1$. These intervals form a clique L and according to Lemma 6.1, there exists an interval $j \in L$ with density $D(j \mid L) > 1/4$. Then $I = (j)$ and $N = (1/4)$ is an admissible pair. Again we have to check the conditions. The numbers n_1 and s_1 are $1/4$, whereas $r_1 = n_1/2 = 1/8$. Since the interval $i_1 = j$ is colored $x - 1$, it is contained in the block $B_1 = C[x - 24 + 1, x]$. The condition (4) is fulfilled, because the density of the interval is $d_1 > 1/4 = n_1$.

Now fix an admissible pair (N, I). We begin the process of constructing (N', I'). Let $B = C[x - 192 \cdot s_t + 1, x - 192 \cdot r_t] \cap N(i_t)$.

The bandwidth of the interval i_t is $b(i_t) \leq 1/2$. Since $i_t \in B_t$, the color $f(i_t)$ of the interval i_t is larger than $x - 192 \cdot r_t$. According to First-Fit, there exists for each color $\alpha \in [x - 192 \cdot s_t + 1, x - 192 \cdot r_t]$ a point $r_\alpha \in i_t$ such that there are intervals with color α intersecting at r_α, whose bandwidths sum up to more than $1/2$. Otherwise the interval i_t would have received a smaller color. Since $s_t - r_t = n_t/2$, there are at least $96 \cdot n_t$ places r_α where intervals with total bandwidth greater than $1/2$ intersect, and according to Lemma 6.1, we find at least $96 \cdot n_t$ intervals of different colors each with density exceeding $1/4$ within its color class. Let M denote this set of intervals.

We partition M into level sets according to the left-to-right ordering of the intervals. Let $L_k = \{i \in M : h(i \mid M) = k \leq d(i \mid M)\}$ and $R_k = \{i \in M : h(i \mid M) > k = d(i \mid M)\}$. For an interval $i \in M$, $i \in L_k \cup R_k$ or $i \in M \setminus \bigcup_{j<k}(L_j \cup R_j)$ implies that the centrality $c(i \mid M)$ of i with respect to M is at least k.

By the pigeon hole principle (at least) one of the following four cases holds true. Otherwise M would contain fewer than $96 \cdot n_t$ intervals.

(i) There is an interval $i \in M$ with high centrality $c(i \mid M) \geq 3 + \lceil \log_4 n_t \rceil$.

(ii) There is a large central level set, i.e. $|L_k| \geq 8n_t 4^{2-k}$ or $|R_k| \geq 8n_t 4^{2-k}$ for some k such that $2 \leq k \leq 2 + \lceil \log_4 n_t \rceil$.

(iii) There is a large first level set, i.e. $|L_1| \geq 8n_t$ or $|R_1| \geq 8n_t$.

(iv) There is an extra large outer level set, i.e. $|L_0| \geq 28n_t$ or $|R_0| \geq 28n_t$.

Case (i): There is an interval $i \in M$ with high centrality.

Then we extend the sequence I by this interval i. The length of the sequence increases by 1. Hence, we set $t' = t + 1$, $i'_{t+1} = i$, and $n'_{t+1} = 1/4$, because the interval i'_{t+1} has density greater than $1/4$ within its color class.

We have to show that $N' = N \circ n'_{t+1}$ and $I' = I \circ i'_{t+1}$ form an admissible pair (N', I'). For condition (1) we have to observe that the interval i'_{t+1} is chosen from B which is a subset of B'_{t+1}, because $r'_{t+1} = s_t + n'_{t+1}/2 > s_t$. Hence, $i'_{t+1} \in B'_{t+1}$. Condition (2) holds, because $c'_{t+1} \geq 3 + \lceil \log_4 n_t \rceil \geq 2$ since $n_t \geq 1/4$. The number $n'_{t+1} = 1/4 = 4^{-1} = n_t 4^{2-3-\log_4 n_t} \geq n_t 4^{2-c'_{t+1}}$ fulfills the condition (3), and the density $d'_{t+1} \geq 1/4 = n'_{t+1}$, which establishes condition (4). The density of the interval $i'_{t'}$ with respect to $\mathcal{I}_1$ is at least $s'_{t'} = s_t + 1/4$.

Case (ii): There is a large central level set.

Here we can without loss of generality assume that $|L_k| \geq 8n_t 4^{2-k}$ for some k such that $2 \leq k \leq 2 + \lceil \log_4 n_t \rceil$. The other case is exactly symmetric. Note that L_k is a clique in $\mathcal{I}_1$ since $i, j \in M$ and $i < j$ implies that $h(i \mid M) < h(j \mid M)$. Furthermore, each interval in L_k

has a density greater $1/4$ within its color. Thus, by applying Lemma 6.2 with $\rho = 1/4$ we can find an interval $i \in L_k$ with density $D(i \mid B) > n_t 4^{2-k}$. We extend the sequence I by this interval i and set the values $t' = t + 1$, $i'_{t+1} = i$, and $n'_{t+1} = n_t 4^{2-k}$. Note that $n'_{t+1} \geq 1/4$ since $n'_{t+1} = n_t 4^{2-k} \geq n_t 4^{2-2-\lceil \log_4 n_t \rceil} \geq n_t 4^{-\log_4 n_t - 1} = 1/4$.

Again, we have to verify that $N' = N \circ n'_{t+1}$ and $I' = I \circ i'_{t+1}$ form an admissible pair (N', I'). The condition (1) is satisfied, because the interval i'_{t+1} is chosen from B, which is a subset of B'_{t+1}, since $r'_{t+1} = s_t + n'_{t+1}/2 > s_t$. The centrality $c'_{t+1} \geq k \geq 2$ and the number $n'_{t+1} = n_t 4^{2-k}$ fulfill the conditions (2) and (3). The condition (4) concerning the density of the interval i'_{t+1} holds, because $d'_{t+1} > n_t 4^{2-k} = n'_{t+1}$.

Hence, the density of the interval $i'_{t'}$ with respect to $\mathcal{I}_1$ is at least $s'_{t'} = s_t + n_t 4^{2-k}$.

Case (iii): There is a large first level set.

Because of symmetry reasons, we can without loss of generality assume that $|L_1| \geq 8n_t$. For the same reason as in Case (ii) the set L_1 is a clique in $\mathcal{I}_1$, and each interval in L_1 has a density greater $1/4$ within its color class. Hence, by Lemma 6.2 (again with $\rho = 1/4$) there exists an interval $i \in L_1$ with density $D(i \mid B) > n_t$.

Since this interval i with high density in B'_{t+1} might have a centrality of one only, thus violating the condition (2), we replace the interval i_t with the interval i. We set $t' = t$, $i'_t = i$, and $n'_t = 2n_t$.

Then the new block $B'_t \supseteq B_t \cup B$, since $r'_t = s_{t-1} + n'_t/2 = s_t$. Also the interval $i'_t \subset i_t$, because i'_t has centrality 1 in the subset M of the neighborhood $N(i_t)$. Hence, $i'_t \in B'_t$ establishing the condition (1). The condition (2) simply holds, because we have $c'_t \geq c_t \geq 2$, if $t > 1$. For condition (3) a rough estimate suffices, namely $n'_t > n_t \geq n_{t-1} 4^{2-c_t} \geq n'_{t-1} 4^{2-c'_t}$. The density condition (4) is satisfied, because $d'_t \geq D(i \mid B) + d_t > 2n_t = n'_t$.

In total, the density of the interval $i'_{t'}$ with respect to $\mathcal{I}_1$ is at least $s'_{t'} = s_t + n_t$.

Case (iv): There is an extra large outer level set.

Again, we can without loss of generality assume that $|L_0| \geq 28n_t$. Each interval in L_0 has density greater $1/4$ within its color class. Let e be the leftmost point that is contained in the interval i_t as well as in all intervals in L_0. We define $T := B_t \cup B$. Then, the density

$D(e \mid T)$ of the point e with respect to T is at least the density of the interval i_t with respect to B_t plus the density of e within the set L_0. Hence, $D(e \mid T) \geq D(i_t \mid B_t) + D(e \mid L_0) > n_t + 7n_t = 8n_t$, and according to Lemma 6.1 we find an interval j_1 in the clique of intervals in T containing e, such that $D(j_1 \mid T) \geq 4n_t$.

If $t = 1$, we will replace the interval i_t with the interval j_1 as explained below. If $t > 1$, we additionally choose an interval j_2 from B_t such that its centrality $c(j_2 \mid B_t) = c_t - 1$ and $j_2 < i_t$. Note that such an interval must exist in this case, because of the centrality of the interval i_t in B_t.

Now we distinguish two cases according to the left interval endpoints of j_1 and j_2. Either the left endpoint of j_2 lies further left, which implies that the centrality $c(j_1 \mid T) \geq c(j_2 \mid T) \geq c_t - 1$, or the left endpoint of the interval j_1 lies further left, in which case the interval j_2 is contained in j_1, and thus the density $D(j_2 \mid T) \geq D(j_1 \mid T) \geq 4n_t$.

In either case there exists an interval $i \in \{j_1, j_2\}$ in T such that $D(i \mid T) \geq 4n_t$ and $c(i \mid T) \geq c_t - 1$. In order to fulfill the condition (2) we have to replace either the interval i_t or the interval i_{t-1} with the new interval i depending on the centrality $c(i \mid T)$.

- If the centrality $c(i \mid T) \geq 2$ or $t = 1$, we replace the interval i_t with the new interval i, and set the values $t' = t$, $i'_t = i$, and $n'_t = 4n_t$.

  The new block $B'_t \supset T$, since $r'_t = s_{t-1} + n'_t/2 = s_{t-1} + 2n_t = s_t + n_t > s_t$. Because of the centrality of the interval i'_t the condition (2) is clearly satisfied, and we have also that $i'_t \subset i_{t-1}$, if $t > 1$. Hence, the interval i'_t is contained in B'_t fulfilling the condition (1). To verify the condition (3) we need a short calculation. We have $n'_t = 4n_t \geq 4n_{t-1}4^{2-c_t} \geq n'_{t-1}4^{2-c'_t}$, where the last inequality holds because $c'_t \geq c_t - 1$. The condition (4) holds because the density $d'_t \geq 4n_t = n'_t$.

  Hence, the density of the interval $i'_{t'}$ with respect to $\mathcal{I}_1$ is at least $s'_{t'} = s_t + 3n_t$.

- Otherwise $c(i \mid T) = 1$ and $t > 1$. Thus the centrality $c_t = 2$ and $n_t \geq n_{t-1}$. In this case we discard the interval i_t and replace the interval i_{t-1} with the new interval i. We set the numbers $t' = t-1$, $i'_{t-1} = i$ and $n'_{t-1} = 4n_t$.

  We are left to show that the sequences $N' = N_{t-2} \circ n'_{t-1}$ and $I' = I_{t-2} \circ i'_{t-1}$ form an admissible pair (N', I').

For condition (1) observe that the new block $B'_{t-1} \supseteq B_{t-1} \cup T$, since $r'_{t-1} = s_{t-2} + n'_{t-1}/2 = s_{t-2} + 2n_t \geq s_{t-2} + n_{t-1} + n_t = s_t$. Because of the centrality of the interval i'_{t-1} we have $i_{t-1} \supset i'_{t-1}$. Thus, we can conclude that $i'_{t-1} \in B'_{t-1}$. Since the interval i'_{t-1} is contained in the interval i_{t-1}, the centrality $c'_{t-1} \geq c_{t-1}$, which establishes the condition (2), because $c_{t-1} \geq 2$, if $t > 2$. The condition (3) holds, since $n'_{t-1} = 4n_t > n_{t-1} \geq n_{t-2}4^{2-c_{t-1}} \geq n'_{t-2}4^{2-c'_{t-1}}$. Finally, the density condition (4) is fulfilled because $d'_{t-1} \geq D(i \mid T) + d_{t-1} \geq 4n_t + n_{t-1} > 4n_t = n'_{t-1}$.

This time, the density of the interval $i'_{t'}$ with respect to $\mathcal{I}_1$ is at least $s'_{t'} = s_{t-2} + 4n_t \geq s_{t-2} + n_{t-1} + n_t + 2n_t = s_t + 2n_t$.

Altogether, this proves the induction step and therewith the lemma. $\square$

Chapter 7

Conclusion

The work presented in this thesis addresses two optimization problems that are encountered in the operation of communication networks, namely call admission control and fixed-path coloring. The call admission control problem is concerned with the decision which connection requests to accept and which to reject in order to maximize some objective function such as the number of accepted calls, for example. The fixed path coloring problem asks for an assignment of wavelengths (or colors) to a set of paths in the network such that paths that share a link are assigned different colors. The goal of the fixed path coloring problem is to use as few colors as possible.

The goal of this thesis was to study the complexity of several variants of these optimization problems, and to provide polynomial-time algorithms that either solve the problems optimally or compute provably good approximations for problems that are difficult to solve optimally.

We obtained the following results.

- A linear-time optimal algorithm for call admission control in chain networks with arbitrary edge capacities in the case that all calls have unit profit and require one unit of bandwidth.
- A polynomial-time optimal algorithm for call admission control in rings with arbitrary edge capacities in the case that the calls are given as pre-determined paths, have unit profit, and require one unit of bandwidth. This algorithm can be implemented to have a running-time of $O(\ell n c_{min} \log \ell)$, where n is the number of nodes in the ring, ℓ is the number of given paths, and c_{min} denotes the minimum edge capacity.

- An 18-approximation algorithm for call admission control in star networks with arbitrary edge capacities in the case that the calls are associated with a starting time, a duration, a positive bandwidth requirement, and an arbitrary profit. This resolves an open problem due to Erlebach [35].

- A 24-approximation algorithm for the same call admission control problem in stars as above, if additionally call alternatives are allowed.

- An $\mathcal{APX}$-hardness proof for call admission control in star networks with unit edge capacities in the case that each call has one out of three possible starting times, unit duration, unit bandwidth, and unit profit. This implies $\mathcal{APX}$-hardness for all call admission control variants that are generalizations of this restricted variant. We also present an alternative $\mathcal{NP}$-hardness proof.

- A 2-approximation algorithm for call admission control in star networks with unit edge capacities in the case that each call is associated with an arbitrary starting time and an arbitrary duration, and has unit bandwidth and unit profit. This problem generalizes the restricted variant that is proved to be $\mathcal{APX}$-hard.

- A fixed-parameter tractable algorithm for a parameterized version of the above problem in star networks, if the number of rejected calls is taken as the parameter.

- A 3-approximation algorithm for call admission control in star networks with arbitrary edge capacities in the case that each call is associated with a starting time, a duration, and an arbitrary profit, and requires unit bandwidth.

- A 195-competitive on-line algorithm for the problem of coloring intervals with bandwidth requirements in $(0, 1]$.

A general goal of future research is of course to find better approximation algorithms for various call admission control problems. As far as my personal interests are concerned, I find the open call admission control problems in the realm of ring networks most challenging. In particular, these involve the inquiry of the complexities of the two problems <Ring, det $|\, b_i = 1$> and <Ring $|\, b_i = 1,\, w_i = 1$>, which are both variations of the problem <Ring, det $|\, b_i = 1,\, w_i = 1$>. In Section 4.2 we showed how to solve the latter problem in polynomial time, although, at first sight, it appears to be close to the maximum k-colorable subgraph problem for circular-arc graphs, which is $\mathcal{NP}$-hard.

For both problems, 2-approximations can be obtained rather easily by the edge-removal technique. An LP-based approach that yields a 2-approximation for <Ring, det $|\, b_i = 1$> is presented in [9]. Recently, Anand and Erlebach observed that a result by Hochbaum and Levin [55] shows that solving the problem optimally is at least as difficult as the bipartite exact matching problem, which is known to have a randomized polynomial-time algorithm (even in the non-bipartite case) [74], but to date its complexity has not been established to be in $\mathcal{P}$ or $\mathcal{NP}$-hard according to [55]. If this problem turns out to be hard to solve optimally, an improved approximation algorithm (with a ratio better than 2) would be desirable.

In the second problem (<Ring $|\, b_i = 1, w_i = 1$>) only the endnodes of the calls are given and the algorithm must determine a path for each of the accepted requests. Anand and Erlebach presented an approximation algorithm for this problem that always computes feasible solutions containing at least *OPT* $- 3$ accepted calls, where *OPT* denotes the number of calls in the optimal solution [9]. We implemented an algorithm that first computes the routing of all calls that minimizes the maximum load of a link. This can be done efficiently and optimal in our setting [78], and leaves us with an instance in which the routing of every call is determined. Then the algorithm computes a maximum cardinality subset of these pre-determined calls that does not violate any capacity constraints using our algorithm described in Section 4.2. In the experiments we performed on randomly generated instances this combined algorithm accepted never less than *OPT* $- 1$ calls. We believe that this problem is polynomial-time solvable, but although there is some evidence for this belief, we did not yet discover the truth. Any continuative result (validation or disproof of our belief) on this problem would be very interesting future research.

Bibliography

[1] U. Adamy, C. Ambuehl, R. S. Anand, and T. Erlebach. Call control in rings. In *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 2380, pages 788–799. Springer-Verlag, 2002.

[2] U. Adamy and T. Erlebach. Online coloring intervals with bandwidth. In *Proceedings of the 1st Workshop on Approximation and Online Algorithms (WAOA)*, LNCS, 2003. To appear.

[3] U. Adamy, T. Erlebach, D. Mitsche, I. Schurr, B. Speckmann, and E. Welzl. Call admission control with advance reservations in star networks. Manuscript, September 2003.

[4] 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 (SODA)*, pages 1–10, 1999.

[5] P. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(3–4):209–218, 1998.

[6] S. Albers. Competitive online algorithms. *Optima*, 54:1–8, 1997. Feature article in the newsletter of the Mathematical Programming Society.

[7] S. Albers. Online algorithms: a survey. *Mathematical Programming Series B*, 97(1–2):3–26, 2003.

[8] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, New York, Second edition, 1992.

[9] R. S. Anand and T. Erlebach. Routing and call control algorithms for ring networks. In *Proceedings of the 8th International Workshop on Algorithms and Data Structures (WADS)*, LNCS 2748, pages 186–197. Springer-Verlag, 2003.

[10] R. S. Anand, T. Erlebach, A. Hall, and S. Stefanakos. Call control with k rejections. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT)*, LNCS 2368, pages 308–317. Springer-Verlag, 2002.

[11] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties*. Springer-Verlag, Berlin, 1999.

[12] Y. Azar and O. Regev. Strongly polynomial algorithms for the unsplittable flow problem. In *Proceedings of the 8th Integer Programming and Combinatorial Optimization Conference (IPCO)*, LNCS 2081, pages 15–29. Springer-Verlag, 2001.

[13] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001.

[14] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[15] Y. Bartal and S. Leonardi. On-line routing in all-optical networks. *Theoretical Computer Science*, 221(1–2):19–39, 1999.

[16] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Perennes, and U. Vaccaro. Graph problems arising from wavelength-routing in all-optical networks. In *Proceedings of IPPS, 2nd Workshop on Optics and Computer Science (WOCS)*, 1997. Published on CD-ROM.

[17] P. Berman, B. DasGupta, S. Muthukrishnan, and S. Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 427–436, 2001.

[18] A. Blum, A. Kalai, and J. Kleinberg. Admission control to minimize rejections. In *Proceedings of the 7th International Workshop on Al-*

gorithms and Data Structures (WADS), LNCS 2125, pages 155–164. Springer-Verlag, 2001.

[19] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.

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

[21] G. Calinescu, A. Chakrabarti, H. Karloff, and Y. Rabani. Improved approximation algorithms for resource allocation. In *Prodeedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, LNCS 2337, pages 401–414. Springer-Verlag, 2002.

[22] M. C. Carlisle and E. L. Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59:225–235, 1995.

[23] P. Carmi, T. Erlebach, and Y. Okamoto. Greedy edge-disjoint paths in complete graphs. In *Proceedings of the 29th International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, 2003. To appear.

[24] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, LNCS 2462, pages 51–66. Springer-Verlag, 2002.

[25] C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity demand flow in a tree. In *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 2719, pages 410–425. Springer-Verlag, 2003.

[26] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. In *Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, LNCS 1665, pages 313–324. Springer-Verlag, 1999.

[27] C. Chhabra, T. Erlebach, B. Stiller, and D. Vukadinović. Price-based call admission control in a single DiffServ domain. Technical Report TIK-Report 135, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, June 2002. Available electronically at ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report135.pdf.

[28] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: An approach to high bandwidth optical WAN's. *IEEE Transactions on Communications*, 40(7):1171–1182, 1992.

[29] M. Chrobak and M. Ślusarek. On some packing problems related to dynamic storage allocation. *RAIRO Informatique Théoretique et Applications*, 22:487–499, 1988.

[30] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[31] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1997.

[32] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. Resource management with hoses: point-to-cloud services for virtual private networks. *IEEE/ACM Transactions on Networking (TON)*, 10(5):679–692, 2002.

[33] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[34] T. Erlebach. Approximation algorithms and complexity results for path problems in trees of rings. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, LNCS 2136, pages 351–362. Springer-Verlag, 2001.

[35] T. Erlebach. Call admission control for advance reservation requests with alternatives. In *Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE)*, Proceedings in Informatics, pages 51–64. Carleton Scientific, 2002.

[36] T. Erlebach and K. Jansen. Call scheduling in trees, rings and meshes. In *Proceedings of the 30th Hawaii International Conference on System Sciences (HICSS)*, volume 1, pages 221–222. IEEE Computer Society Press, 1997.

[37] T. Erlebach and K. Jansen. Off-line and on-line call-scheduling in stars and trees. In *Proceedings of the 23rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, LNCS 1335, pages 199–213. Springer-Verlag, 1997.

[38] T. Erlebach and K. Jansen. Scheduling of virtual connections in fast networks. In *Proceedings of the 4th Parallel Systems and Algorithms Workshop (PASA)*, pages 13–32. World Scientific Publishing, 1997.

[39] T. Erlebach and K. Jansen. The maximum edge-disjoint paths problem in bidirected trees. *SIAM Journal on Discrete Mathematics*, 14(3):326–355, 2001.

[40] T. Erlebach, K. Jansen, C. Kaklamanis, and P. Persiano. An optimal greedy algorithm for wavelength allocation in directed tree networks. In *Proceedings of the DIMACS Workshop on Network Design: Connectivity and Facilities Location*, volume 40 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 117–129. AMS, 1998.

[41] T. Erlebach and D. Vukadinović. New results for path problems in generalized stars, complete graphs, and brick wall graphs. In *Proceedings of the 13th International Symposium on Fundamentals of Computation Theory (FCT)*, LNCS 2138, pages 483–494. Springer-Verlag, 2001.

[42] E. M. Eschen and J. P. Spinrad. An $O(n^2)$ algorithm for circular-arc graph recognition. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 128–137, 1993.

[43] A. Feldmann. On-line call admission for high-speed networks (Ph.D. Thesis). Technical Report CMU-CS-95-201, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1995.

[44] H. Gabow and R. Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30(2):209–221, 1985.

[45] J. A. Garay, I. S. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. *Journal of Algorithms*, 23:180–194, 1997.

[46] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York-San Francisco, 1979.

[47] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic and Discrete Methods*, 1(2):216–227, 1980.

[48] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1:180–187, 1972.

[49] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[50] M. C. Golumbic and R. E. Jamison. The edge intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 38(1):8–22, 1985.

[51] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[52] R. A. Guérin and A. Orda. Networks with advance reservations: The routing perspective. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 118–127, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[53] 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 Annual ACM Symposium on Theory of Computing (STOC)*, pages 19–28, 1999.

[54] A. Gyárfás and J. Lehel. Online and First-Fit colorings of graphs. *Journal of Graph Theory*, 12:217–227, 1988.

[55] D. S. Hochbaum and A. Levin. Cyclical scheduling and multi-shift scheduling: complexity and approximation algorithms. Manuscript, June 2003.

[56] K. Jansen. Approximation results for wavelength routing in directed trees. In *Proceedings of IPPS, 2nd Workshop on Optics and Computer Science (WOCS)*, 1997. Published on CD-ROM.

[57] C. Kaklamanis and P. Persiano. Efficient wavelength routing on directed fiber trees. In *Proceedings of the 4th Annual European Symposium on Algorithms (ESA)*, LNCS 1136, pages 460–470. Springer-Verlag, 1996.

[58] C. Kaklamanis, P. Persiano, T. Erlebach, and K. Jansen. Constrained bipartite edge coloring with applications to wavelength routing. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS 1256, pages 493–504. Springer-Verlag, 1997.

[59] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27–35, 1991.

[60] I. A. Karapetian. On the coloring of circular arc graphs. *Journal of the Armenian Academy of Sciences*, 70(5):306–311, 1980. (in Russian).

[61] R. M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Proceedings of the Workshop on Complexity of Computations (Yorktown Heights)*, pages 85–103. Plenum Press, 1972.

[62] H. A. Kierstead. The linearity of First-Fit coloring of interval graphs. *SIAM Journal on Discrete Mathematics*, 1(4):526–530, 1988.

[63] H. A. Kierstead. Coloring graphs on-line. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, LNCS 1442. Springer-Verlag, Berlin, 1998.

[64] H. A. Kierstead and J. Qin. Coloring interval graphs with First-Fit. *SIAM Journal on Discrete Mathematics*, 8:47–57, 1995.

[65] H. A. Kierstead and W. T. Trotter, Jr. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.

[66] R. Klasing. Methods and problems of wavelength-routing in all-optical networks. Technical Report CS-RR-348, Department of Computer Science, University of Warwick, September 1998.

[67] J. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[68] V. Kumar and E. J. Schwabe. Improved access to optical bandwidth in trees. In *Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 437–444, 1997.

[69] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys. Sequencing and Scheduling: Algorithms and Complexity. In S. Graves, A. R. Kan, and P. Zipkin, editors, *Handbooks in Operations Research and Management*

Science: Logistics of Production and Inventory, volume 4, pages 445–522. North-Holland, Amsterdam, 1993.

[70] S. Leonardi. On-line network routing. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, LNCS 1442. Springer-Verlag, Berlin, 1998.

[71] L. Lewin-Eytan, J. S. Naor, and A. Orda. Routing and admission control in networks with advance reservations. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, LNCS 2462, pages 215–228. Springer-Verlag, 2002.

[72] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matchings in general graphs. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

[73] M. Mihail, C. Kaklamanis, and S. Rao. Efficient access to optical bandwidth. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 548–557, 1995.

[74] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[75] T. Nishizeki and K. Kashiwagi. On the 1.1 edge-coloring of multigraphs. *SIAM Journal on Discrete Mathematics*, 3(3):391–410, 1990.

[76] S. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE Journal of Selected Areas in Communications*, 13(6):1128–1136, 1995.

[77] P. Raghavan and E. Upfal. Efficient routing in all-optical networks. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 134–143, 1994.

[78] A. Schrijver, P. Seymour, and P. Winkler. The ring loading problem. *SIAM Journal on Discrete Mathematics*, 11(1):1–14, 1998.

[79] W.-K. Shih and W.-L. Hsu. An approximation algorithm for coloring circular-arc graphs. In *SIAM Conference on Discrete Mathematics*, 1990.

[80] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[81] J. J. Sylvester. Thoughts on inverse orthogonal matrices, simultaneous sign-successions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 34(232):461–475, 1867.

[82] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232, 1985.

[83] G. Wilfong and P. Winkler. Ring routing and wavelength translation. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 333–341, 1998.

[84] H. S. Witsenhausen. On Woodall's interval problem. *Journal of Combinatorial Theory (Series A)*, 21:222–229, 1976.

[85] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987.

Curriculum Vitae

Udo Adamy

born on November 22, 1973 in Bernkastel–Kues, Germany

1984–1992	**high school in Bernkastel–Kues, Germany** Degree: Abitur
1992–1993	**civil service in Bernkastel–Kues, Germany**
1993–1998	**studies at University of Saarbrücken, Germany** major: Computer Science minor: Mathematics Degree: Diplom (Masters)
1995–1996	**Exchange student at University of Edinburgh, Scotland, U.K.** Computer Science
1999–2003	**Ph.D. student at ETH Zürich, Switzerland** Theoretical Computer Science