Algorithms for Call Control in Ring Based Networks

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of

Doctor of Sciences

presented by

R. SAI ANAND

M. E. in Computer Science & Engineering,
Indian Institute of Science, Bangalore
born 26.05.1976
citizen of India

accepted on the recommendation of
Prof. Dr. Thomas Erlebach, examiner
Dr. habil. Martin Skutella, co-examiner

2004
Communication networks lend themselves to the study of a variety of optimization problems. A fundamental problem among them that is intensively investigated is that of call admission control, or call control for short. The problem arises in the following setting. The communication network consists of nodes (e.g. terminals, routers, et cetera) which are connected to each other via links (e.g. optical cables). These links carry data between nodes. They have a certain bandwidth capacity associated with them. A call is a communication request between any two nodes of the network. It is associated with a bandwidth requirement, i.e. the rate at which data is communicated between the nodes. A call may be accepted into the network if (i) a route connecting the two nodes exists and (ii) the bandwidth requirement of the call can be reserved along all the links of one such route. Once a call is accepted a profit is accrued by the network. The optimization problem arises due to the fact that the sum of the bandwidth requirements of calls that are routed through any link of the network should not exceed its bandwidth capacity. The call control problem is to maximize the number (or profits) of calls that can be accepted into the network such that the capacity constraints on the links are not violated.

A ring topology is one in which the nodes are connected to each other forming a cycle. In all-optical networks, the ring topology is a popular configuration. Bigger networks are formed from individual rings by interconnecting them. Some top reasons that make the ring topology a favoured one are its simplicity, scalability and survivability in the presence of link failures. This thesis mainly investigates several variants of call control on ring based topologies for two reasons. Firstly, all-optical networks are in the forefront of revolutionizing communications today and the ring topology is the building block of such networks. Secondly, rings are simple topologies on which several network optimization problems have been studied in the past and there are several others which have not been resolved. The goal of our research, the outcome of which is the thesis presented here, is to address a few such problems pertaining to call control.

There are two basic variants of call control that we consider, namely on-line and off-line. In the off-line version of call control, it is assumed that
all the calls that occur in the network are given in advance. An algorithm for this version can decide which calls to accept and which to reject taking this overall picture into consideration. In the online version, calls arrive into the network in a sequence, one after the other. An algorithm for the on-line version makes a decision to accept a call that is presented to it only based on what decisions it made in the past and the current state of the network. In particular, it has no knowledge of the calls that might be presented to it in the future.

A tree of rings is a graph obtained by connecting several disjoint rings in a “tree”-like fashion, by identifying vertices in different rings. The tree of rings topology is a natural result of interconnecting rings in all-optical networks to form larger networks. We investigate on-line call control problems on trees of rings. Non-preemptive randomized algorithms with competitive ratios that are best possible up to constant factors are given.

Fixed parameter tractability is an emerging area to tackle \( \mathcal{NP} \)-hard problems. We view call control problems from this perspective. Fixed parameter tractability results are shown for variants of the off-line call control problems on arbitrary graphs, undirected and bidirected trees of rings. To the best of our knowledge, these are the first such results for the call control problem.

For two variants of off-line call control problems in rings, we present polynomial time approximation algorithms. When the route of a call can be determined by the algorithm, we present an algorithm that accepts and routes at most 3 calls fewer than what an optimal algorithm can achieve. When the routes are pre-determined we present an algorithm that achieves a profit that is at least one-half of that achieved by an optimal algorithm. For various special cases, we present optimal polynomial time algorithms or approximation schemes. We also give an indication of the difficulty of finding an optimal polynomial algorithm for the general problem.

For addressing the off-line call control problem for calls with arbitrary bandwidth requirements, a starting point would be to study it on the simple topology of a line. A line is simply a set of nodes connected to form a path. We identify several restrictions for which algorithms with “nice” ratios are achievable.
Zusammenfassung


Control anzugehen.


Man erhält einen Tree of Rings Graphen, wenn man mehrere disjunkte Ringe “baumartig” zusammenfügt, wobei schrittweise Knoten verschiedener Ringe zu einem Knoten zusammengefasst werden. Die Tree of Rings Topologie ist eine natürliche Konsequenz, wenn Ringe in rein optischen Netzwerken verbunden werden, um größere Netzwerke zu formen. Wir untersuchen on-line Call Control Probleme in Tree of Rings Graphen. Es werden nichtpreemptive, randomisierte Algorithmen angegeben, deren kompetitive Raten bis auf konstante Faktoren bestmöglich sind.


Für zwei Varianten des off-line Call Control Problems in Ringen präsentieren wir Approximationsalgorithmen, die in polynomialer Zeit laufen. Für den Fall, dass die Route eines Calls durch den Algorithmus bestimmt werden kann, geben wir einen Algorithmus an, der höchstens 3 Calls weniger annimmt (wobei eine Route für jeden angenommenen Call ausgegeben wird) als ein optimaler Algorithmus. Für den Fall, dass die Routen vorgegeben sind, präsentieren wir einen Algorithmus, der mindestens die Hälfte des Profits eines optimalen Algorithmus erzielt. Für diverse Spezialfälle geben wir optimale Polynomialzeit-Algorithmen oder Approximationsschemata an. Des weiteren weisen wir auf die Schwierigkeit hin, einen optimalen polynomialen Algorithmus für das allgemeine Problem zu finden.

Acknowledgements

My heartfelt thanks primarily and unreservedly go to my advisor Thomas Erlebach. He has been a fountainhead of inspiration and guidance for me throughout my tenure as a PhD student. The sense of confidence he infused in me, especially during those times when the problems at hand seemed intractable, has proved to be invaluable. At a personal level, he was very supportive, patient and easily accessible. Without his unlimited supply of ideas and tips for solving problems and help in formulating the problems in the first place, the present work could not have taken shape. All the results in this thesis have been influenced by him, in one way or the other. It was a pleasure and honour to work under his supervision.

Thanks to Martin Skutella for kindly agreeing to be a co-referee of my work. His careful reading and feedback has helped improve the presentation of the thesis.

Financial support for my doctoral studies came from the European Graduate Program, “Combinatorics, Geometry and Computation” (CGC). I joined the program as a pre-doc student in October 2000 and subsequently, was granted a scholarship to continue as a doctoral student. I would like to register my thanks and appreciation for the efforts of Emo Welzl, Speaker of the CGC program at ETH Zürich, in founding and running the program.

I spent the 5 months from August to December 2002 working in the research group of Rolf Möhring at TU Berlin, as part of an exchange semester. I am grateful to him for his kindness in inviting me.

Thanks are due to my colleagues in the Theory Group; Alex Hall, Matúš Mihaľák, Stamatis Stefanakos and Danica Vukadinović. They made a terrific set of work-mates whether that be at research or over a cup of coffee at Z’vieri. Thanks to them, in particular, for proof-reading parts of my thesis. I thank Udo Adamy, Christoph Ambühl, Thomas Erlebach, Alex Hall and Stamatis Stefanakos, with whom I co-authored various papers.

Life outside work would not have been half as fun but for the new friends I made in Zürich. They made it easier for me to settle down in a foreign land in addition to sharing lighter moments with me. Thanks to all of them. Special thanks to a very special friend, Anna, whose love and care has brought a new freshness into my life. I should mention my friends from back home; specifically, the BigBoys who kept constantly in touch with me and recreated the good times that we enjoyed together at IISc through group chat sessions.

Finally, I would like to express my deep sense of gratitude to my family for their love, support and encouragement.
Contents

Abstract i

Zusammenfassung iii

Acknowledgements v

1 Introduction 1
  1.1 Motivation ................................. 1
  1.2 Call control .............................. 5
  1.3 Organization of the thesis ............... 7

2 Preliminaries 9
  2.1 A quick 101 course to the thesis ......... 9
     2.1.1 The basics .......................... 9
     2.1.2 Graphs ................................ 12
     2.1.3 Parameterized complexity .......... 15
  2.2 Problem definitions ....................... 16
  2.3 Previous work ............................ 19
     2.3.1 Results relating to off-line call control .................. 19
     2.3.2 Results relating to on-line call control .......... 23

3 On-line Call Control 27
  3.1 The algorithm for on-line MEDP .......... 28
  3.2 The algorithm for on-line MEDPwPP ....... 30
     3.2.1 Terminology .......................... 31
     3.2.2 Intersecting versus crossing .......... 32
     3.2.3 The algorithm ........................ 33
     3.2.4 Analysis of the algorithm ............ 34
  3.3 A lower bound for randomized algorithms . 41
4 Call Control: FPT Results
  4.1 Off-line MEDPwPP-k on arbitrary graphs .......... 47
  4.2 Off-line MEDP-k on trees of rings ............. 48
    4.2.1 The undirected case ....................... 49
    4.2.2 The bidirected case ....................... 52

5 Routing and Call Control
  5.1 The unit profits case .......................... 61
    5.1.1 An integer linear program ................. 62
    5.1.2 Rounding scheme .......................... 63
  5.2 Arbitrary profits case .......................... 78
  5.3 Remarks ...................................... 79

6 Pre-routed Call Control
  6.1 Off-line, uniform PCC on rings ................. 82
    6.1.1 An integer linear program ................. 82
    6.1.2 A 2-approximation ........................ 82
    6.1.3 Notes on the complexity of off-line, uniform PCC .. 84
  6.2 Optimal algorithms for special cases of off-line, uniform PCC . 87
    6.2.1 Calls with routes of equal length .......... 88
    6.2.2 Calls with “proper” routes ................ 90
    6.2.3 Calls with restricted profits .............. 91
    6.2.4 The dual LP ................................ 92
  6.3 Algorithms for special cases of a different “kind” ................ 94
    6.3.1 Minimum edge capacity bounded by a constant .... 94
    6.3.2 Calls with profits proportional to length of routes ... 95
    6.3.3 Calls with bounded route lengths ............ 96

7 Call Control on Lines
  7.1 PCC on lines with constant number of vertices .... 102
  7.2 PCC on lines when calls have bounded length routes .... 104
  7.3 PCC on lines when calls have equal length routes ... 105
    7.3.1 $\min \{ \frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}} \}$ is a constant .......... 105
    7.3.2 $\min \{ \frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}} \}$ is arbitrary .......... 108

8 Conclusions and Open Problems
  8.1 Summary of results ........................... 111
  8.2 Further research ............................. 112

Bibliography .................................. 115
Contents

Index 123
Curriculum Vitae 125
1 Introduction

The first part of the chapter gives an overview of optical networks and the SONET standard. It places in perspective the importance of ring based topologies for such networks. In the second part we motivate the study of call control problems in communication networks and present the organization of the thesis.

1.1 Motivation

In 1876, then US president, Rutherford B. Hayes, is reported to have exclaimed, “An amazing invention, but who would ever want to use one?”. He was alluding to the “electrical speech machine” (what we call telephone these days) invented by Alexander G. Bell earlier that year. Little could anyone at that time imagine how this invention would change the world in the years to come. The humble beginnings in the spring of 1876 were the precursors to modern day telecommunications; ranging from the modest telephone to the “digital age” Internet. Communication networks have become so ubiquitous in the 21st century that life without them is almost unimaginable.

Optical networks. Optical fiber technology is in the forefront of revolutionizing present day communication networks. Optical fibers are essentially thin fibers of glass that have a cylindrical core surrounded by a cladding. In optical networks, light is used as the carrier of information over optical fibers. The phenomenon of total internal reflection makes it possible to send light signals through the fiber over long distances.

Today, networks based on optical fibers are fast replacing traditional copper based ones. The advantages that optical networks provide include high bandwidth capabilities and ability to carry multiple types of traffic. The speed at which these networks operate also cannot be matched by electromechanical devices. While optical technology offers bandwidths upwards of 1Gbps (up to around 40Gbps) and potentially several terabits per second, typical coaxial cable lines offer bandwidths of the order of a few 100Mbps (up to around 1Gbps). The long distances through which optical fibers can carry
data without the use of repeaters is another benefit. Unlike electric signals which are susceptible to electronic interferences from the surroundings, optical signals are immune to such interferences. Among the few disadvantages of optical networks vis-a-vis the traditional ones are the costs of producing optical fibers and the complexity of connecting fibers to each other.

A dominant technology standard that is prevalent in implementing optical networks is *Synchronous Optical Network*, SONET for short. It standardizes the bit rates, coding schemes and operational functionality of the various network elements. Traditionally, network elements have been asynchronous, each running on its own clock. This meant that coordinating communication between two different nodes required several stages. SONET, on the other hand, is a synchronous system; the average frequency of all clocks is the same. This makes multiplexing of various signals easier to handle. The standardization allows easy integration of network elements that are produced by a variety of vendors. Most asynchronous systems are capable of only point-to-point communication. With SONET it is possible to have multi-point configurations enabling optimal concurrent sharing of the SONET infrastructure by several customers.

The SONET standard specifies several network configurations [Son01]. These are *Point-to-Point*, *Point-to-Multipoint*, *Hub Network* and *Ring Architecture*. The Point-to-Point configuration is the simplest where two terminal nodes are connected to each other with a fiber. In Point-to-Multipoint, a special network element called an add/drop multiplexer (ADM) facilitates the adding and dropping of several connections at intermediate points. The Hub Network is designed for scalability of optical networks. A hub concentrates traffic at a central site and allows easy re-provisioning of the connections. In the Ring Architecture, several ADMs are connected with fibers to form a ring for either bi-directional or uni-directional traffic. This is the most popular of the SONET configurations because of its fault tolerance capability. See Figure 1.1 for the different SONET network configurations.

**Ring networks.** The ring topology is a favoured one not only in optical networks. In local area networks that use cable based technology, the Token Ring network is widely used. The Token Ring was originally developed by IBM in the 1970s. In this network, the terminals are connected to a set of Multistation Access Units (MSAU). These MSAUs themselves are connected to each other in a closed loop completing the ring network. Ethernet based LAN typically employ a bus or star topology. However, several implementations of Ethernet Virtual Private Networks use an emerging new technology called Resilient Packet Ring. This technology provides a virtual ring network connectivity, although the physical connections may not be a ring topology.

There are several advantages to the ring topology. First, it is a very simple
topology and is easy to implement. Second, in a ring all the nodes connected to it have equal access and thus the topology offers fairness. Third, it is scalable and the impact of scaling on performance is minimal. Adding new network elements is relatively easy in a ring network. Fourth, and perhaps the most important is the survivability of the network in the presence of link failures. If a single link connecting the nodes fails, existing connections routed through that link can be diverted via a different one through the ring.

The robustness of the ring topology with respect to link failures has made it the building block of optical networks. Several Ring Architectures are interconnected to each other to form bigger networks. Ring based network topologies have become commonplace and are being deployed at a fast rate. This growth also gives rise to a number of research issues that have to be tackled in these networks. For this reason, we concentrate on rings or ring based networks in this thesis.
1.1. Motivation

Network optimization. Countering the enormous capabilities of optical networks in terms of high bandwidth, speed and multiplexing of various types of traffic are the increasing sizes and complexity of these networks and rise in bandwidth hungry applications like multimedia, supercomputer networking, video-on-demand et cetera. The increasing size introduces problems relating to efficient usage of the network resources. On the other hand, high bandwidth applications competing for the limited (albeit high) bandwidths on the fiber links lead us to issues of profitability and fairness. In effect, the advancing technology of the communication networks and the increase in applications that use them entail the study of a variety of optimization problems. Two fundamental problems that arise in this context are call admission control and wavelength assignment. The former problem is common to all types of communication networks while the latter is a specific feature of optical networks.

The wavelength assignment problem arises from the Wavelength Division Multiplexing (WDM) employed in optical networks. For establishing several connections that share a fiber, the bandwidth of the fiber is partitioned into several channels. In each channel, the allotted bandwidth can be used at a unique wavelength. When two different signals use different wavelengths on the same fiber there is no interference between them. Therefore, several connections can share a fiber by sending information at different wavelengths. The typical number of channels on a fiber is in the range of 10-100 in practice. The wavelength assignment problem is due to this limit on the number of channels available for use. It has been investigated in two forms: (i) to maximize the number of connections that can be assigned a wavelength constrained by the maximum number of wavelengths allowed on each fiber and (ii) to minimize the number of wavelengths when all the connections are assigned wavelengths such that two connections that share a fiber get different wavelengths. While the second form is useful during design of optical networks based on forecasted traffic, the first is useful when the network is operational.

The call admission control, or briefly call control, problem arises from the limited supply of bandwidth on the links and the quality of service guarantees demanded by network applications. A call (variously referred to as call request or connection request) is a communication request between a pair of nodes that run a network application between them. A Quality of Service (QoS) guarantee for the application means that some pre-determined amount of network resources, like bandwidth, are reserved along the route through which the call is to be established. This is absolutely essential if the application is to be executed to completion. At any point in time, there are several call requests waiting to be admitted into the network. However, due
to the QoS requirements of the calls and the bandwidth capacity limitations on the links not all of them can be admitted simultaneously. The call control problem, essentially, then is to decide which of these requests are to be accepted and which to be rejected. Within the bandwidth limitations of the links that carry data, the optimization problem of call control is to maximize the profits accrued from call requests that can be accepted into the network.

1.2 Call control

In this thesis, we exclusively concentrate on the call control problem. As pointed out in the previous section, the growing importance of optical networks and the ring topology as a building block motivates the study of network optimization problems in ring based networks. In this section, we shall examine the problem in more detail by abstracting it into a mathematical model and then go on to describe variants to the basic problem that we shall investigate. We shall state most of the notions in an informal manner for now. Formal definitions of the different problems are presented in the following chapters.

A graph model. For our purposes it suffices to treat a communication network as an undirected capacitated graph. Network node elements like terminal nodes, ADMs and routers will be mapped as vertices of the graph and the fiber links connecting them as edges of the graph. The bandwidths on the fiber links translate to the capacities of the edges. For a call between a pair of nodes to occur, a path in the network connecting the nodes should be established. Such a path is called a route for the call. The bandwidth requirement of a call is the rate at which data is to be transmitted between the end nodes of the call. The QoS requirements of the call imply that if the call is admitted into the network then along all the edges of the route the bandwidth demand of the call has to be allocated to it. The capacity on an edge places limits on calls that can be routed through it; namely, the sum of the bandwidths of calls routed through it cannot exceed its capacity. An accepted call, usually, has an associated profit that is earned by the network if the call is accepted and completed. The call control problem is to maximize the profits of the calls that can be accepted while satisfying the capacity constraints on the edges.

Variants of call control. There are several aspects to the fundamental problem just described that yield particular variants.
(a) Pre-routed versus un-routed: When the network is well connected, there are several choices for establishing a route for a call. However, due to predetermined routing strategies or protocols implemented in networks, the
route for a call is sometimes fixed as soon as the end nodes are known. In this case, the call control algorithm has to route the call along this pre-determined route if it chooses to accept the call. This is referred to as pre-routed call control. An efficient way of using the network would be to let the call control algorithm choose the routes for individual calls by itself. Taking into account the complete set of call requests many more calls could be accepted if the routes were not fixed beforehand. This variant where the routing choice in addition to the acceptance and rejection decision is left to the call control algorithm is known as routing and call control.

(b) Uniform versus non-uniform bandwidth requirements: Typically, different network applications have differing bandwidth requirements. It is sometimes appropriate and mostly algorithmically easier to handle the problem where all the bandwidth requirements are equal. This special case is called uniform call control. The general problem where the bandwidths are unequal is called non-uniform call control. The uniform case is equivalent to all bandwidth requirements being unity after the capacities on the edges are suitably scaled.

(c) On-line versus off-line: The off-line version of call control is one in which all the call requests that occur in a network are given in advance. In the off-line variant decisions to accept and reject calls can be made taking into consideration this overall picture of the network. In contrast, for the on-line call control problem the calls that occur in the network are revealed to the algorithm in an incremental fashion, one at a time. Here, the algorithm needs to make a decision on accepting or rejecting a call as soon as it is presented. With no knowledge of future requests, an on-line algorithm has only partial information based on which it can take a decision. The decision can depend on the actions performed by it in the past. Decisions to reject calls are not revoked. The on-line variant comes in two different flavours depending on whether acceptance decisions are permanent or not. In the preemptive version, a call that was previously accepted may be removed and replaced with a call being presently requested. This decision may be based on a combination of factors; the bandwidth requirements and profits of the calls. In the preemptive version no profit is earned from preempted calls. In the non-preemptive version, a decision on acceptance is never reversed. The on-line problem models the real life situations more accurately. At the same time, off-line problems are important in their own right and also as a tool to evaluate the performance of on-line algorithms.

Time dimension. In real life, calls have life spans. The life span starts from the moment the call is requested, it continues through the duration of the call if it is admitted and ends when the call transaction completes or is rejected. This adds another dimension, of time, to the call control problem. In this thesis, we do not study this dimension. In our model, all calls are
assumed to exist simultaneously in the network once accepted. An accepted call lives in the network as long as any other accepted call exists. This simplifying assumption makes it easier to handle the already hard problem of call control.

1.3 Organization of the thesis

In Chapter 2, Preliminaries, definitions of terms and concepts used in the thesis are presented. Along with the formal definitions of the problems that we investigate, this chapter also contains a brief literature survey on topics related to the thesis.

In Chapter 3, On-line Call Control, we present results on on-line call control for the trees of rings topology. Both the pre-routed and routing versions of the call control problem are studied in the on-line and uniform variant setting. With edge capacities set to unity, the problem we investigate is the on-line version of the well known Maximum Edge Disjoint Paths (MEDP) problem. For both the pre-routed and routing versions we give logarithmic competitive ratios which are the best possible up to constant factors.

Parameterized complexity [DF99] has developed into an active area of research in the recent years. It is an interesting combination to study call control in terms of parameterized complexity. To the best of our knowledge, this is the first attempt in viewing call control problems from this perspective. In Chapter 4, Call Control: FPT Results, we consider the number of rejected calls as the parameter. For undirected and bidirected trees of rings with unit capacity edges we show fixed parameter tractability of the off-line, uniform, routing and call control problem.

Chapters 5 and 6 are devoted to studying call control problems in rings. In Chapter 5, Routing and Call Control, we study the off-line, uniform, routing and call control problem in rings. For the objective of maximizing the cardinality of accepted and routed calls we show an additive guarantee approximation algorithm that accepts and routes at most 3 fewer calls as compared to an optimal algorithm. For the objective of maximizing profit we provide a 2-approximation algorithm. Unfortunately, the computational complexities of both these problems are unknown at the time of writing.

Investigation of the pre-routed variant of the above problem is carried out in Chapter 6, Pre-routed Call Control. For the objective of maximizing cardinality, we showed a polynomial time optimal algorithm in [AAAE02]. Here, we consider the profit maximization problem and show a 2-approximation. The computational complexity of this problem is unresolved. It turns out that the problem is at least as hard as the exact matching problem in bi-
partite graphs whose complexity is not known for over 15 years. We show further that for several special cases it is possible to obtain polynomial time optimal algorithms or (randomized) approximation schemes.

We take a break from ring topologies in Chapter 7, Call Control on Lines. We consider the off-line, non-uniform call control problem on a line, which is simply an undirected path. The Knapsack problem is a special case of it and therefore the call control problem we study is \( \mathcal{NP} \)-hard. The results we obtain are (i) fully polynomial time approximation schemes (FPTAS) for constant length lines, (ii) polynomial time approximation schemes (PTAS) for calls with bounded length routes and (iii) constant and poly-logarithmic approximation ratios for special cases of calls having equal length routes.

There are several questions that stem from the research presented here in this thesis. We conclude by pointing out interesting open problems and directions for possible future research in the final chapter, Conclusions and Open Problems (Chapter 8).

**Remarks.** Many of the results presented here are joint work with other researchers and have appeared in the proceedings of various conferences or journals. The results of Chapter 3 were obtained together with T. Erlebach and presented at the LATIN 2002 conference [AE02]. Fixed parameter tractability results in Chapter 4 are part of the results presented in [AEHS02] at the SWAT 2002 conference (collaboration with T. Erlebach, A. Hall and S. Stefanakos). An extended version of this article appeared in [AEHS03]. The results on call control problems in ring networks shown in Chapters 5 and 6 appeared in the proceedings of WADS 2003 [AE03] (collaboration with T. Erlebach).
Preliminaries

Notations and/or definitions of mathematical terms and concepts that we will use throughout this thesis are presented along with formal specifications of the problems we study. This chapter also summarizes previous work related to the topics we investigate.

2.1 A quick 101 course to the thesis

2.1.1 The basics

Combinatorial optimization, which is the broad area under which we study all our problems, deals in numbers. The most basic of numbers are the set of natural numbers, we denote them by \( \mathbb{N} \). The set of whole numbers, \( \mathbb{N}_0 \), contains zero in addition to all the natural numbers. \( \mathbb{Z} \) denotes the set of all integers. The set of all positive (negative) integers is represented by \( \mathbb{Z}^+ (\mathbb{Z}^-) \), a subscript of 0 to the symbol denotes the corresponding set together with the element zero. The set of real numbers is denoted by \( \mathbb{R} \). A set of objects will be denoted by capital letters or a sequence of capital letters, i.e. \( A, B, OPT, \) et cetera. The cardinality of a set \( A \) is given by the symbol \( |A| \). All logarithms are of base 2 unless noted otherwise. As is standard the symbol \( e \) will denote the base of natural logarithms.

A decision problem is one which has two possible answers; “yes” or “no”. An algorithm to solve a decision problem is a step by step procedure that for every input produces the correct answer. The performance of an algorithm can be measured in terms of time and space that it uses to arrive at a solution. We concentrate only on time as a performance measure of an algorithm. Loosely speaking, given a problem and an input the time required by the algorithm is the number of steps executed by it. The notions of algorithms and their execution times can be formalized in the Turing machine model.

An important aspect connected to measuring the performance of an algorithm is the size of the input. The input instance is given to the algorithm using an encoding scheme. For instance, all input parameters may be repre-
presented as a string of 0s and 1s or characters from some other alphabet. The
length of the input then is the number of characters used for encoding the
whole of it. The worst case time complexity of an algorithm is the maximum
time (measured as a function of the size of the input) required by it over all
possible inputs of the same size. An algorithm is considered efficient if its
worst case time complexity is polynomially related to the input size (and we
say the algorithm runs in polynomial time). We do not present the formal
details of algorithms, Turing machines, size of the input and running time
here but point out the excellent tutorial book by Garey and Johnson [GJ79].

Decision problems can be seen as a language. The latter is a subset of
strings of an underlying alphabet set. A language is said to be in class \( \mathcal{P} \),
if there exists a deterministic Turing machine that accepts a string from the
alphabet set if and only if it belongs to the language and runs in polynomial
time. A decision problem is in \( \mathcal{P} \) if the corresponding language is in \( \mathcal{P} \).
In other words, there exists a polynomial time algorithm that always correctly
outputs “yes” or “no” for every input to the decision problem. A language
is in the class \( \mathcal{NP} \) if there is a polynomial time non-deterministic Turing
machine that recognizes the strings of the language. A decision problem
whose equivalent language is in \( \mathcal{NP} \) is also in \( \mathcal{NP} \).

\( \mathcal{NP} \)-complete is the set of those problems in \( \mathcal{NP} \) that are in some sense
the hardest to solve. A problem in \( \mathcal{NP} \) is said to be \( \mathcal{NP} \)-complete if any
problem in \( \mathcal{NP} \) can be reduced to it using a polynomial reduction. Essentially,
this implies finding a polynomial algorithm for an \( \mathcal{NP} \)-complete problem
yields an efficient algorithm to all problems in class \( \mathcal{NP} \) or mathematically,
\( \mathcal{P} = \mathcal{NP} \). It is widely believed that the existence of a polynomial time
algorithm for an \( \mathcal{NP} \)-complete problem is unlikely, i.e. \( \mathcal{P} \neq \mathcal{NP} \). A decision
problem \( \ell \) is called \( \mathcal{NP} \)-hard if every problem in \( \mathcal{NP} \) can be polynomially
reduced to it.

An optimization problem is a computational problem whose objective is
to find the best of all possible solutions. Formally, an optimization
problem is a four tuple, \( \Pi = (D_{\Pi}, S_{\Pi}, \text{obj}_{\Pi}, \text{spec}) \). \( D_{\Pi} \) is the set of instances of
the optimization problem \( \Pi \). \( S_{\Pi} \) is the family of feasible solutions for the
instances. \( \text{obj}_{\Pi} \) is a function from the set \( D_{\Pi} \times S_{\Pi} \) to the set of rational
numbers. \( \text{spec} \) is a specification whether the optimization problem \( \Pi \) is a
maximization or minimization problem. We refer the reader to [Vaz01, p.
345] for a comprehensive definition of optimization problems. Let \( OPT(I) \)
represent the objective value of an optimal solution to instance \( I \) of the
maximization (minimization) problem \( \Pi \). That is, \( OPT(I) = \text{obj}_{\Pi}(I, s^*) \) is the
objective value of a feasible solution \( s^* \) such that \( \text{obj}_{\Pi}(I, s^*) \geq \text{obj}_{\Pi}(I, s) \)
(\( \text{obj}_{\Pi}(I, s^*) \leq \text{obj}_{\Pi}(I, s) \)), for every feasible solution \( s \) of \( I \).

Decision problems can be formulated from the problem underlying an
optimization problem $\Pi$ as follows. Together with an input instance $I$ of $\Pi$ we specify a rational number $B$. If $\Pi$ is a maximization (minimization) problem we ask for a “yes” or “no” answer to the question: Is there a feasible solution $s$ of $I$ such that the objective function is greater (less) than $B$? A polynomial time algorithm for the optimization problem will answer the above question in polynomial time and hence the decision version is in class $P$. On the other hand, if the decision version is $NP$-hard then it establishes hardness for the optimization problem. We also say the optimization problem is $NP$-hard, overloading the formal definition of $NP$.

In this thesis, we will mainly encounter optimization problems that are $NP$-hard. While there is little hope of finding polynomial time optimal algorithms for them, one can search for an approximation algorithm. An approximation algorithm always produces a feasible solution to the input instance that is also guaranteed to be “near” the optimal solution. The nearness is measured using the approximation ratio. An algorithm $A$ for a maximization type optimization problem $\Pi$ is said to have an approximation ratio of $\delta > 1$ if for any instance $I$ of $\Pi$ it produces a feasible solution $t^*$ of $I$ such that $OPT(I) \leq \delta \cdot obj_\Pi(I, t^*)$ and further runs in time polynomial in the input size, $|I|$. If $\Pi$ is a minimization problem, we require the approximation algorithm to produce a feasible solution $t^*$ such that $obj_\Pi(I, t^*) \leq \delta \cdot OPT(I)$ and to run in time polynomial in the size of the input. An approximation scheme for an optimization problem $\Pi$ is an algorithm $A$ with an error parameter $\epsilon > 0$ as part of the input that has an approximation ratio of $1 + \epsilon$, for every $\epsilon$. $A$ is said to be a polynomial time approximation scheme, abbreviated PTAS, if for every fixed $\epsilon$ it runs in time polynomial in the size of the input. $A$ is called a fully polynomial time approximation scheme (FPTAS) if its running time is polynomial both in the size of the input and $\frac{1}{\epsilon}$.

In the above discussion, an algorithm was implicitly assumed to be deterministic. Given an input, such an algorithm always produces the same solution. However, there are several problems (like playing against an adversary) where such algorithms perform badly. The only way out is to introduce some kind of randomization in the algorithm. A randomized algorithm is one which can use random bits produced by some source and use them in its computation. The output of a randomized algorithm will depend on the random bits it uses and therefore may be different from one run of the algorithm to another on the same input instance. For many problems, randomized algorithms run faster than the best known deterministic algorithms of similar performance. Refer to the book [MR95] for more on randomized algorithms. The approximation ratio of a randomized algorithm for an optimization problem is defined differently. Instead of comparing the objective value of the solution produced on a single run of the algorithm we compare
the “expected” objective value. Formally, a randomized approximation algorithm has a ratio of $\delta \geq 1$ for a maximization type optimization problem if $OPT(I) \leq \delta \cdot E[\text{obj}_\Pi(I, s)]$ for all instances $I$, where the expectation is taken over the random choices of the algorithm and $s$ is the feasible solution of $I$ produced by the algorithm. For a minimization problem, a randomized algorithm has an approximation ratio $\delta \geq 1$ if $E[\text{obj}_\Pi(I, s)] \leq \delta \cdot OPT(I)$ for all instances $I$, where the expectation is taken over the random choices of the algorithm and $s$ is the feasible solution of $I$ produced by the algorithm.

We are also interested in optimization problems that are on-line in nature, in this work. In an on-line problem, the input is fed to the algorithm incrementally. The algorithm needs to process the part of the input presented to it before it can process the succeeding parts. Thus, for an on-line problem an algorithm makes a decision based only on partial knowledge of the input. Performances of algorithms for on-line problems are measured using competitive ratios. The “quality” of the solution output by the algorithm is compared to that of an optimal off-line algorithm that knows the whole of the input in advance. Assume that the input of the on-line optimization problem II with an objective of maximization is fed to the on-line algorithm $\mathcal{A}$ in $m$ parts, say $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$. Let the solution computed by the on-line algorithm be $s_\sigma$. The value of the solution is computed similar to a normal optimization problem as $\text{obj}_\Pi(\sigma, s_\sigma)$. Let $OPT(\sigma)$ represent the value of the optimal off-line solution. The algorithm $\mathcal{A}$ is said to have a competitive ratio of $\rho \geq 1$, or simply to be $\rho$-competitive, if $OPT(\sigma) \leq \rho \cdot \text{obj}_\Pi(\sigma, s_\sigma)$, for every possible input $\sigma$. For minimization problems, the above inequality is replaced by $\text{obj}_\Pi(\sigma, s_\sigma) \leq \rho \cdot OPT(\sigma)$.

### 2.1.2 Graphs

Recall from Chapter 1 that we model communication networks as graphs. In this section, we shall explain graph theoretic concepts that we will use in the rest of this thesis.

A graph $G$ is given by an ordered pair $(V, E)$. $V = \{v_1, v_2, \ldots\}$ is a set of vertices of the graph and $E \subseteq \{\{u, v\} : u, v \in V\}$ is a set of unordered pairs of vertices called edges. We will only be concerned with finite graphs, which are graphs with a finite vertex set. Figure 2.1 shows an example of a graph, pictorially.

An edge $e = \{u, v\} \in E$ is said to be incident on the vertices $u$ and $v$, we also say that the vertices are incident on the edge. Two edges are said to be incident on each other if they share an incident vertex. Two vertices are adjacent to each other if there is an edge incident on both of them. A neighbour of a vertex is any of the vertices adjacent to it. The degree of a
Figure 2.1: A graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_5\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_5\}, \{v_3, v_4\}, \{v_4, v_5\}\}$.

A path between the vertices $v_i$ and $v_{n+1}$ in a graph is an alternating sequence of vertices and edges, $v_1, e_1, v_2, e_2, \ldots, v_n, e_n, v_{n+1}$ such that $e_i$ is incident on the vertices $v_i$ and $v_{i+1}$, $i = 1, 2, \ldots, n$. The path is said to pass through the edges that appear in its sequence. The length of a path is the number of edges in it. A path is simple if every vertex appears in it at most once. A simple path between the vertices $u$ and $v$ is said to be a shortest path if the length of no other path between the two vertices is strictly smaller than its length. The diameter of a graph is the maximum of the lengths of shortest paths between every pair of vertices in the graph. A path is a cycle if the first and last vertices are identical, do not appear elsewhere in the path and all other vertices in the path occur exactly once in it. A graph is connected if there is a path between any pair of vertices in it. A component of a graph is a maximal connected subgraph in the graph. A cut is a set of edges whose removal results in an increase in the number of components of the graph.

A chain or line is a graph that consists of a simple path. A ring is a graph that is a cycle. A tree is a connected graph that has no cycles in it. A tree of rings is a class of graphs that is recursively defined as: (a) a ring is a tree
A tree can be rooted at an arbitrary vertex, called its root. The vertices adjacent to the root vertex are its children and the root is the parent of its children. This parent-child relationship can be propagated throughout the tree: The vertices adjacent to the child of a root (other than the root itself) are its children, the vertices adjacent to these children (other than the children of the root) are their children and so on. Rooting a tree at a vertex assigns a unique parent to every vertex except the root itself. The ancestors
of a vertex are all the vertices in the path between it and the root, excluding itself. If \( u \) is an ancestor of \( v \) then \( v \) is a descendant of \( u \). The level of the root is 0 and the level of any other vertex is 1 more than the level of its parent. For two vertices \( u \) and \( v \), we define the least common ancestor (lca\((u, v)\)) as the largest level vertex that lies on the path from \( u \) to the root and on the path from \( v \) to the root. We shall carry over these definitions to trees of rings by considering the “underlying” tree and talk about the root ring, the parent ring of a ring, the least common ancestor ring of two rings and so on.

A directed graph or digraph \( G = (V, E) \) is defined similar to a graph except that for the edges the order of the incident pair of vertices is important. The first vertex in the order is the start vertex and the second is the end vertex. An edge in a digraph is visualized as a directed arc from the start vertex to the end vertex. A digraph is bidirected if \((u, v) \in E \implies (v, u) \in E\).

A particularly useful enhancement to graphs as defined above is the notion of edge capacitated graphs. Besides the vertex and edge sets, capacitated graphs are specified with a capacity function on the edge set \( c : E \rightarrow \mathbb{N}_0 \). This is useful when modelling problems as graph problems. The capacity function could be used to model lengths, weights, demands, et cetera that occur in the problem. Of course, one may think of capacity functions on the vertex set too. For our purposes here, capacity functions on the edge set suffice. Unless otherwise specified our discussions will center around (undirected) edge capacitated graphs. An edge capacitated graph \( G = (V, E) \) with the capacity function \( c \) on the edges will be written compactly as \( G = (V, E, c) \) from now on.

### 2.1.3 Parameterized complexity

Parameterized complexity is emerging as a prominent research topic for dealing with \( \mathcal{NP} \)-hard problems. The basic idea behind it is to find algorithms for \( \mathcal{NP} \)-hard problems whose running time can be restricted to be exponential in a parameter to the problem.

The favourite example to illustrate the notion is to consider the \( \mathcal{NP} \)-hard problem \textsc{MinVertexCover}. The object of \textsc{MinVertexCover} is, given a graph \( G = (V, E) \) to find a minimum cardinality vertex set \( V' \subseteq V \) that “covers” the edges, i.e. for every edge \( e \in E \) there is a vertex \( v \in V' \) such that \( e \) is incident on \( v \). A naive approach is to try all possible subsets of \( V \) and answer the question. From the classical viewpoint of computational complexity all known algorithms for this problem are not much better than this. In the parameterized setting, one asks the following question: Given a parameter \( k \leq |V| \), does there exist an algorithm that finds a vertex cover of size at most \( k \) (if such a vertex cover exists) or answers “no” if no such
vertex cover exists, in time $O(f(k) \cdot poly(|V|, |E|))$. The function $f$ may be any arbitrary function. The answer to this particular question is yes [DF99]. Problems like the parameterized version of MINVERTEXCOVER for which it is possible to design algorithms that run in time $O(f(k)poly(|I|))$ ($|I|$ is the size of the input and $k$ the parameter) are said to be fixed parameter tractable (FPT).

As one might imagine, not all problems are FPT. Parameterized complexity gives rise to a new hierarchy of problem classes analogous to $\mathcal{P}$ and $\mathcal{NP}$ in the classical view. They are instructive, as problems that were classified to be in the same class of $\mathcal{NP}$-hard previously, can be distinguished in terms of their tractability or otherwise in the parameterized view. We do not need much heavy machinery from parameterized complexity other than the above description for our investigations. The interested reader is referred to the book by Downey and Fellows [DF99] on the subject.

2.2 Problem definitions

Now, we are in a position to formally define the call control problem and its several variants. The communication network is modelled as an edge capacitated graph. The nodes of the network (like terminal nodes, routers, et cetera) form the vertex set of the graph and the physical links (like cables, fibers) connecting the nodes its edge set. For the capacities on edges, we consider the bandwidth capacities of the links. We assume the capacities are whole numbers.

A call $r$ in the communication network is identified by 5 parameters, $r = (u_r, v_r, b_r, t_{start,r}, t_{end,r})$. $u_r$ and $v_r$ are the nodes of the network between which communication occurs for the call, also called its end vertices. $b_r \in \mathbb{N}$ is the bandwidth requirement of the call. This has to be reserved along all the edges of a route used for establishing the call (Recall that a route for a call is a path connecting its end vertices). $t_{start,r}$ and $t_{end,r}$ respectively are the times at which the call starts and ends. In the off-line setting, the start and end times of all calls are the same and without loss of generality we assume, $t_{start,r} = 0$ and $t_{end,r} = 1$, for all calls $r$ in the network. In the on-line setting, while the start times differ the end times are all the same. Here, a call once accepted into the network will remain as long as any other accepted call is in the network. Without loss of generality, we let $t_{end,r} = \infty$, for all calls $r$. A call $r$ also has a profit, $p(r) \in \mathbb{N}$, associated with it. If not explicitly specified calls are assumed to have a profit of 1. The profit of a set of calls is the sum of the profits of individual calls in it.

Given a communication network represented as a graph $G = (V, E, c)$, a
set of calls $S$ and a set $R$ specifying a route for each call in $S$, $R$ is said to be feasible if for every edge $e \in E$ the sum of the bandwidth requirements of calls whose routes in $R$ pass through $e$ is at most $c(e)$. A set of calls $S$ is said to be feasible for the graph $G$ if (i) every call can be assigned a route and (ii) the set of assigned routes is feasible. In general, if calls have varying start and end times, feasibility is verified for the routes of calls that exist at each time instant $t$. For the on-line setting by our assumption that all calls have the same end times, it is enough to verify that feasibility holds for all of the accepted calls.

The two variants of call control, un-routed and pre-routed, described in the first chapter give us the two general problems of RoutingAndCallControl (RCC) and PreRoutedCallControl (PCC). The formal definitions of these problems follow. When we talk about a set of calls (or their routes) we allow the set to be a multi-set, i.e. an element may occur in it with any multiplicity. When the call control problem is on-line in nature the input set of calls should be seen as a sequence of calls that arrive in the order of their start times. As we noted while describing the on-line variant in the previous chapter, the calls are either accepted or rejected as soon as they arrive.

<table>
<thead>
<tr>
<th>Problem RoutingAndCallControl (RCC)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An edge capacitated graph $G = (V, E, c)$ and a set of calls $S$. A profit function on the set of calls $p : S \rightarrow \mathbb{N}$.</td>
</tr>
<tr>
<td><strong>A feasible solution:</strong> A feasible subset of calls $S' \subseteq S$ together with a feasible set of routes specified for calls in $S'$.</td>
</tr>
<tr>
<td><strong>Objective:</strong> A maximum profit feasible solution.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem PreRoutedCallControl (PCC)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An edge capacitated graph $G = (V, E, c)$, a set of calls $S$ and a set of routes $R$ (one for each call in $S$). A profit function on the set of calls $p : S \rightarrow \mathbb{N}$.</td>
</tr>
<tr>
<td><strong>A feasible solution:</strong> A subset of calls $S' \subseteq S$ such that $R_{S'} \subseteq R$ is feasible, where $R_{S'}$ is the set of routes of calls in $S'$.</td>
</tr>
<tr>
<td><strong>Objective:</strong> A maximum profit feasible solution.</td>
</tr>
</tbody>
</table>

The distinction between the two problems disappears when we consider $G$ to be a chain or a tree. For any two vertices in these topologies there is
only one possible route connecting them. Hence, we shall just talk about the call control problem for them without qualifying it with the terms un-routed or pre-routed.

The special case of call control with uniform bandwidth requirements and all edge capacities unity is also of theoretical interest. This is the well known maximum edge disjoint paths problem. The two general problems above have their own versions of it which we call \textsc{MaximumEdgeDisjointPaths} (MEDP, for short) and \textsc{MaximumEdgeDisjointPathswithPrespecifiedPaths} (MEDPwPP, for short) problems. Two paths in a graph are said to be \textemdash edge disjoint if there is no edge through which both of them pass.

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\textbf{Problem} \textsc{MaximumEdgeDisjointPaths} (MEDP) \\
\textbf{Input:} A graph $G = (V, E)$ and a set of calls $S$ with unit bandwidth requirements. \\
\textbf{A feasible solution:} A subset of calls $S' \subseteq S$ such that (i) every call in $S'$ is assigned a route and (ii) the assigned routes for any pair of calls in $S'$ are edge disjoint. \\
\textbf{Objective:} A maximum cardinality feasible solution. \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\textbf{Problem} \textsc{MaximumEdgeDisjointPathswithPrespecifiedPaths} (MEDPwPP) \\
\textbf{Input:} A graph $G = (V, E)$, a set of calls $S$ with unit bandwidth requirements and a set of routes $R$ (one for each call in $S$). \\
\textbf{A feasible solution:} A subset of calls $S' \subseteq S$ such that the routes in $R$ of any pair of calls in $S'$ are edge disjoint. \\
\textbf{Objective:} A maximum cardinality feasible solution. \\
\hline
\end{tabular}
\end{table}

Essentially, MEDP is RCC with the input graph having unit edge capacities, the calls having unit bandwidth requirements and profits of unity. MEDPwPP is the corresponding analogue of PCC.

We also investigate the parameterized versions of call control. From an optimality point of view, accepting a maximum cardinality feasible call set is equivalent to rejecting a minimum cardinality call set such that the remaining set of calls is feasible. In practice, we would expect that the network is provisioned with enough bandwidth so that very few calls are rejected. Hence, we take the number of rejected calls as the parameter and formulate the parameterized problems \textsc{RoutingAndCallControl-$k$} and \textsc{PreRoutedCallControl-$k$}. For these parameterized problems we are interested in finding FPT algorithms. Note that, unlike the RCC and PCC problems, their parameterized versions do not consider profits of calls.
Chapter 2. Preliminaries

Problem **RoutingAndCallControl-\(k\)** (RCC-\(k\))

**Input:** An edge capacitated graph \(G = (V, E, c)\), a set of calls \(S\) and an integer \(k \geq 0\).

**Objective:** A subset of calls \(S' \subseteq S\) with \(|S'| \leq k\) such that \(S \setminus S'\) is feasible and to give a feasible set of routes for calls in \(S \setminus S'\) or declare no such subset \(S'\) exists.

Problem **PreRoutedCallControl-\(k\)** (PCC-\(k\))

**Input:** An edge capacitated graph \(G = (V, E, c)\), a set of calls \(S\) and a set of routes \(R\) (one for each call in \(S\)) and an integer \(k \geq 0\).

**Objective:** A subset of calls \(S' \subseteq S\) with \(|S'| \leq k\) such that \(R_{S \setminus S'} \subseteq R\) is feasible, where \(R_{S \setminus S'}\) is the set of routes of calls in \(S \setminus S'\) or declare no such subset \(S'\) exists.

If the capacities on the edges and the bandwidth requirements of calls are all equal to unity, the RCC-\(k\) and PCC-\(k\) problems will be referred to as MEDP-\(k\) and MEDPwPP-\(k\), respectively.

From here on, we shall refer to individual problems by their acronyms. Each of the succeeding chapters presents results on one or more of these problems when the input graph is restricted to a specific topology. We shall not explicitly specify calls by the 5 parameter model described above. Instead, we use descriptions like off-line, uniform bandwidths, et cetera to specify the parameters that are common to all the calls in the set. In any case, it should be clear from the context how one can formulate the problems in the above formal setup.

### 2.3 Previous work

The optimization problems described in the previous section were abstracted out of the call control problem occurring in communication networks. Even so, the underlying graph problem is related to a variety of other well known and studied problems in the literature ranging from interval graph colouring to multi-commodity flows to routing problems in VLSI layout design.

#### 2.3.1 Results relating to off-line call control

The simplest version of the off-line call control problem occurs when the graph is restricted to be just an edge between 2 vertices. The non-uniform
call control problem is then the **Knapsack** problem: Given a knapsack of a certain size and a set of items with varying sizes and weights, pick a subset of maximum weight such that the sum of the sizes of the items in the subset is at most that of the knapsack. **Knapsack** is a weakly \( \mathcal{NP} \)-hard problem and an FPTAS is known for it [IK75]. One can also solve this problem optimally in pseudo-polynomial time using dynamic programming [Dan57].

For chains with unit capacity edges, the call control problem on uniform bandwidth calls is equivalent to finding the maximum weight independent set in an interval graph. The maximum size independent set problem on interval graphs can be solved in polynomial time [Gol80, p. 174]. When all edges on the chain have equal capacity the uniform call control problem is related to the maximum \( k \)-colourable induced subgraph of an interval graph. Carlisle and Lloyd [CL95] gave a linear time algorithm to solve this problem. They also show how to solve the weighted version optimally in polynomial time. This solves the uniform call control problem on chains with equal capacity edges optimally.

A variant of the off-line RCC problem on rings was studied by Wilfong and Winkler [WW98]. They consider a bidirected ring with a set of calls input to it. The objective is to route all of the calls such that the maximum number of routes through an edge is minimized. For this they showed a polynomial time optimal algorithm. For the undirected case, polynomial time solvability of the same problem was shown by Frank [Fra85].

A graph is a **circular-arc graph** if it is the intersection graph of a set of arcs on a circle. Linear time recognition of circular-arc graphs was shown by McConnell [McC01]. Polynomial time algorithms to find maximum size cliques and maximum size independent sets of a circular-arc graph were given by Gavril [Gav74]. The off-line MEDPwPP problem on rings is the maximum size independent set problem on circular-arc graphs. In [AAAE02], the off-line, uniform PCC problem on rings with arbitrary edge capacities is shown to be polynomially solvable when the profits are equal.

**The edge disjoint paths problem.** The MEDP and MEDPwPP problems in the off-line setting have been of interest to the algorithm community independent of their relation to call control. The decision version of off-line MEDP, where the question is whether the call set is feasible, is an \( \mathcal{NP} \)-complete problem for general graphs. \( \mathcal{NP} \)-completeness for meshes was shown in [KvL84] and for planar graphs with maximum degree 3 in [MP93]. Menger’s theorem [Men27] shows a min-max relationship between the number of edge disjoint paths between a pair of vertices in a graph and the sizes of cuts separating them. It states that the maximum number of edge disjoint paths between a pair of vertices is equal to the minimum size of a cut that separates the two vertices. Such optimal edge disjoint paths can be found in
polynomial time using maximum flow algorithms.

For off-line MEDP on directed graphs, Guruswami et al. [GKR+99] showed a lower bound of $\Omega(m^{1/2-\epsilon})$ on the approximation ratio, for any $\epsilon > 0$, where $m$ is the number of edges in the directed graph. Recently, Chekuri and Khanna [CK03] observed that this lower bound is derived for sparse graphs and thus the lower bound when expressed in terms of $n$, the number of vertices, is only $\Omega(n^{1/2-\epsilon})$, for any $\epsilon > 0$. Currently, the best known ratios for off-line MEDP on directed and undirected graphs are $O(\min\{n^{4/5}, \sqrt{m}\})$ and $O(\min\{n^{2/3}, \sqrt{m}\})$, respectively [CK03].

There are special graph classes for which off-line MEDP and MEDPwPP are polynomially solvable; chains (equivalent to the maximum independent set in an interval graph), rings [WL98] and trees [GVY97]. If the number of calls is bounded by a constant then graph minor series results yield a polynomial time algorithm for off-line MEDP on general graphs [RS95].

For off-line MEDP in bidirected trees Erlebach and Jansen [EJ01] have shown a $(\frac{5}{3} + \epsilon)$-approximation, for every fixed $\epsilon > 0$. They also show that the problem is $\mathcal{APX}$-hard for bidirected trees with arbitrary degree. Both off-line MEDP and MEDPwPP for undirected and bidirected trees of rings are constant factor approximable [Erl01] and are also $\mathcal{APX}$-hard.

**Multi-commodity flows.** The RCC problem with uniform bandwidth requirements has connections to multi-commodity flow problems. In the multi-commodity flow problem, there are several commodities (say $k$ of them) to be transported across a network. Associated with each commodity is a source and sink pair between which flow paths have to be established to meet a specified demand. The edges of the network have limited capacities so that the amount of flow of various commodities through an edge should not exceed its capacity. There can be several objectives to the problem. The decision version is to answer whether all the demands can be simultaneously satisfied. The optimization version, where demands may or may not be specified for commodities, is to maximize the sum of the routed flows. If the flows are allowed to be fractional then linear programming can solve the decision and optimization versions. Tardos [Tar86] gave a strongly polynomial time algorithm by showing that any linear program with $\{0, \pm 1\}$ constraint matrix can be solved in strongly polynomial time.

In the following discussion, the underlying graph is assumed to be undirected. When we insist that the flows in a multi-commodity flow be all integers the problem is called integer multi-commodity flow. The special case of integer multi-commodity flow with only one commodity is the maximum flow problem with integer edge capacities and demand. This can be solved in strongly polynomial time, a result that can be obtained by combining those of [FF57] and [EK72]. The problems get harder with increase in
2.3. Previous work

the number of commodities. Already for 2 commodities the decision version is \( \mathcal{NP} \)-complete as was shown by Even, Itai and Shamir [EIS76]. However, the problems can be nicely characterized when additional information on the so-called "Euler condition" is known. The demand across a cut is the sum of the demands of commodities whose source and sink vertices lie on either side of the cut. The capacity of a cut is the sum of the capacities of the edges in the cut. The Euler condition is satisfied when for each vertex the sum of the capacities of edges incident on it plus the sum of demands of commodities having that vertex as its source or sink is even. For the decision version of integer 2-commodity flow with integer edge capacities, if it is known that the Euler condition is satisfied then integer flow paths satisfying the demands exist if and only if the cut condition is satisfied namely, for all cuts, the demand across a cut is at most its capacity [RW66]. Half-integer flows exist for the 2-commodity problem with integer edge capacities even if the Euler condition is dropped [Hu63].

The results for 2 commodities were generalized to three or more commodities by Lomonosov [Lom76, Lom85] and Seymour [Sey80]. A theorem that extends all these results is given in [Sch03, p. 1267]. The demand graph of a multi-commodity flow problem is one with the vertex set consisting of the sources and sinks and the edge set consisting of the source-sink pairs for the commodities. The generalization [Sch03, p. 1267] states that existence of integer flows satisfying all demands is equivalent to the cut condition being satisfied if the demand graph is either a complete graph on 4 vertices or a cycle on 5 vertices or a union of two stars in conjunction with the Euler condition being satisfied.

The decision version of the multi-commodity flow problem on planar graphs is of interest in VLSI design. We point out some important results for this class of graphs and for more related results refer the reader to the survey by Frank [Fra90]. If the source and sink vertices all lie on the outer infinite face of the planar graph then the cut condition is necessary and sufficient for existence of integer flows satisfying demands if the Euler condition is satisfied [OS81]. If the planar graph is a cycle then the cut condition is necessary and sufficient for existence of integer flows satisfying demands if for any cut the difference between the capacity of the cut and the demand across it is even [FNS+92]. Seymour [Sey81] gave the equivalence of the cut condition and existence of integer flows satisfying demands if (i) the Euler condition is satisfied and (ii) the graph obtained from adding edges of the demand graph to the input graph is planar.

The maximization versions of multi-commodity flow problems are generally harder than the decision versions. For 2 commodities, [Raj94] shows that if the capacities on the edges are integers and the sum of the capacities
of edges incident on a vertex that is neither a source nor a sink is even then maximum integer flows equal to the capacity of the minimum cut separating the source and sink pairs exists. If the demand graph is complete and the sum of the capacities of edges incident on a non-source, non-sink vertex is even then the maximum integer flow is equal to the minimum capacity cut separating each source and sink pair [Lov76]. Such a maximum integer flow can be found in polynomial time [IKN98]. A more general result is given by Mader [Mad78] when we restrict flows only along paths whose internal vertices are not sources or sinks.

Several interesting results are known if the topologies of the underlying graph in a multi-commodity flow problem are restricted to lines, stars or trees. For maximizing the sum of the routed flows, the integer multi-commodity flow problem on chains is solvable in polynomial time. When formulated as a linear program the constraint matrix is easily seen to be totally unimodular. For stars with arbitrary capacity edges, the problem is equivalent to finding maximum 6-matchings [GVY97]. In the same article they also show that for trees with unit capacity edges, the maximum integer flow problem is polynomially solvable. However, the problem becomes \( \mathcal{NP} \)-hard for trees with edge capacities 1 and 2. Garg, Vazirani and Yannakakis [GVY97] show a primal dual based 2-approximation for maximizing the sum of the routed integral flows for trees with arbitrary edge capacities. Chekuri, Mydlarz and Shepherd [CMS03] consider the optimization version of the multi-commodity flow problem on trees with demands and profits associated with each commodity. They call it the demand flow problem. Here, the algorithm is not allowed to route a fraction of the demand. It can either route the whole demand (gaining a profit) or none at all (gaining no profit). They show a 4-approximation for unit demands and arbitrary profits. A 48-approximation is also shown for arbitrary demands under the assumption that the maximum demand of a commodity is at most the minimum edge capacity (called bottleneck assumption). For lines and rings [CMS03] give a \((2 + \epsilon)\)-approximation for every \( \epsilon > 0 \), under the bottleneck assumption.

### 2.3.2 Results relating to on-line call control

Early work on on-line call control concentrated on developing algorithms with good competitive ratio for topologies like lines and trees. Garay, Gopal, Kutten, Mansour and Yung [GGK+97] considered the on-line uniform call control on lines with unit edge capacities. Depending on the profit model they obtain the best possible competitive ratios up to constant factors. The algorithms are preemptive and deterministic. If the profit of a call is proportional to the length of its route they show a \((2g + 1)\)-competitive algorithm,
2.3. Previous work

where \( g = \frac{1 + \sqrt{5}}{2} \) is the golden ratio. If all profits are equal (equivalent to on-line MEDP) they present an \( O(\log n) \)-competitive algorithm, where \( n \) is the number of vertices on the line. \([GGK+97]\) also consider the uniform call control problem on a single edge with unit capacity. Unlike the standard on-line assumption we make that all calls have the same end times, they consider the case when calls may have different end times. For this, they give a 4-competitive deterministic algorithm when the profits are proportional to the duration of the call.

Adler and Azar \([AA99]\) investigated randomized and preemptive algorithms for the on-line RCC problem on lines. For on-line MEDP on lines they provide a 16-competitive algorithm. This should be contrasted with the \( \Omega(\log n) \) lower bound for deterministic algorithms. For on-line call control on uniform edge capacity lines they showed an \( O(1) \)-competitive randomized preemptive algorithm when the profits of calls are proportional to their bandwidth requirements.

Results for on-line MEDP on trees are given in \([ABFR94]\) and \([AGLR94]\). Using the “classify-and-randomly” select paradigm Awerbuch et al. \([ABFR94]\) show an \( O(\log n) \)-competitive randomized, non-preemptive algorithm for on-line MEDP on a tree with \( n \) nodes. \([ABFR94]\) also extend their basic randomized algorithm to on-line RCC on trees to yield logarithmic competitive ratio algorithms. The competitive ratio gets an additional multiplicative factor of \( O(\log M) \), where \( M \) is the ratio between the largest and smallest bandwidth requirements. The \( O(\log n) \) ratio was improved to \( O(\log d) \) for on-line MEDP on trees with diameter \( d \) in \([AGLR94]\). This competitive ratio is optimal up to constant factors. We adapt this result for obtaining an on-line algorithm for the tree of rings topology in Chapter 3.

Awerbuch et al. \([AGLR94]\) show results on mesh based topologies for on-line call control problems. A tree of meshes, roughly speaking, is a collection of meshes at various “levels” with interconnection between nodes at consecutive levels which has an underlying tree structure. For a tree of meshes with \( 2\log n \) levels and \( n^2 \) vertices on each level they show an \( O(\log \log n) \)-competitive algorithm for on-line MEDP. They use this to obtain an \( O(\log n \log \log n) \)-competitive algorithm for on-line MEDP on a \( n \times n \) mesh. An \( \Omega(\log n) \) lower bound on the competitive ratio is also provided. Kleinberg and Tardos \([KT95]\) give an \( O(\log n) \)-competitive randomized algorithm for on-line MEDP on the \( n \times n \) mesh.

For on-line call control problems in general graphs it is difficult to obtain poly-logarithmic competitive ratios. If the bandwidth requirement of individual calls can be restricted to be below a fraction of the minimum available bandwidth on the edges then Awerbuch, Azar and Plotkin \([AAP93]\) derive
logarithmic competitive ratios for the on-line RCC problem. Since their analysis makes use of the value of the longest duration of a call, assume that the end times of calls are finite in value and the duration of the longest call is $T$. Let the profit $p(i)$ of a call $i$ be proportional to both its bandwidth requirement $b_i$ and its duration $T_i = t_{\text{end,}i} - t_{\text{start,}i}$. Let the profits satisfy the condition that $n \leq \frac{p(i)}{b_i T_i} \leq Fn$, for some suitable $F$. Denote by $\mu$ the value $2nTF + 1$. An algorithm with competitive ratio $2 \log(2\mu)$ exists when the bandwidths satisfy the requirement $b_i \leq \frac{\min_{c(e) \in c(e)}}{\log \mu}$. [AAP93] also show that if the bandwidth requirements are above a fixed fraction of the minimum capacity then poly-logarithmic competitive ratios are not achievable.

Goel, Meyerson and Plotkin [GMP01] consider the on-line RCC problem in a distributed setting. In their model, there are $k$ routers at which call requests arrive. Each router decides independent of the others how calls will be routed. Restrictions similar to [AAP93] are used to derive $O(\log \mu)$-competitive algorithms. In particular, they show that there is a deterministic algorithm achieving competitive ratio $O(\log \mu)$ if the bandwidth requirements are at most an $O(k \log \mu)$ fraction of the minimum capacity. For randomized algorithms the condition is that the bandwidth requirements be at most an $O(k + \log \mu)$ fraction of the minimum edge capacity.
3 \hspace{1em} \textbf{On-line Call Control}

The on-line MEDP and MEDPwPP problems on trees of rings are studied in this chapter. We give logarithmic competitive ratio algorithms which are optimal up to constant factors.

Trees are the simplest of connected graphs after chains. Consider a set of nodes connected to form a chain and the same set of vertices connected to form an arbitrary tree. They both use the same number of edges. However, speaking in an average sense, over various possible sets of pairs of vertices far more vertex pairs can be routed along edge disjoint paths on the tree network as compared to the chain.

A tree of rings topology is a natural consequence of the above observation: Ring Architectures which form the building blocks of optical networks are connected to each other forming a tree among them. Trees of rings retain the simplicity and better throughput connectivity (as compared to chains) of trees. They inherit the robustness of rings in that removing a few edges (not from the same ring, though) does not disconnect the nodes from each other. Trees, on the other hand, do not have this advantage. Removing even a single edge destroys their connectivity.

From a practical point of view, it is thus clear that the topology of tree of rings is an advantageous one. They are about as easy to implement as trees (on the same set of nodes), at the cost of a few extra links. The extra cost is more than made up by the fault tolerance of the topology in the presence of link failures. For us as theorists, the tree of rings topology offers benefits too. The tree structure at the rings level gives us a handle to exploit the graph properties of trees in the design and analysis of algorithms. For instance, given any pair of vertices in a tree of rings any path connecting them has to pass through the same set of rings in it. On each individual ring we can use the graph properties of rings while solving problems on the whole graph.

We consider the easiest of on-line call control problems on trees of rings in this chapter. The setting is as follows. All edges of the input tree of rings $T$ have a capacity of 1. The calls, which arrive one after the other according to their start times, have a bandwidth requirement of 1 and a profit of 1. All calls have the same end times, which is our standard assumption for on-line
call control problems. We will specify the set of calls (say \( m \) of them) as a sequence \( S = (r_1, r_2, \ldots, r_m) \), where each \( r_i \) is the pair of end vertices of the call (omitting the other common parameters of the calls) and the order in which they appear is the increasing order of their start times. The on-line algorithm should decide to accept (and route, in the case of MEDP) or reject calls as they arrive. The call control problems are, essentially, the on-line MEDP and MEDP\(_{wPP} \) problems on the given tree of rings. Compacting notation, for MEDP\(_{wPP} \) we assume that the \( r_i \)'s are the paths themselves instead of being only the pairs of end vertices of calls. In Section 3.1, we describe an algorithm for the on-line MEDP problem on trees of rings. In Section 3.2, we present the results for on-line MEDP\(_{wPP} \) on trees of rings. Both these algorithms are randomized and non-preemptive in nature. We conclude the chapter by showing lower bounds on the competitive ratios achievable by randomized algorithms for these problems.

### 3.1 The algorithm for on-line MEDP

Recall from the problem definition of MEDP in Chapter 2 that the on-line algorithm while accepting a call must also assign a route to it. If a path between the two end vertices of a call in a tree of rings has to traverse through \( k \) rings there are \( 2^k \) potential routes for the call. We eliminate this large search space for the algorithm by taking advantage of the tree structure. We identify one edge from each of the rings in the tree of rings and prohibit any route from passing through that edge. By doing so, we have reduced the topology of tree of rings to a tree. Now, routing is no longer a problem as there is exactly one possible choice for a call. We need to make sure, however, that by restricting the routes for calls we are not forced to reject too many calls as compared to an optimal off-line algorithm. The following theorem, which has been proved in [Erl01], implies that we lose only a constant factor by removing the edges.

**Theorem 3.1.1** Consider the off-line MEDP problem on a tree of rings \( T \) with the input call set \( S \). Suppose that \( S \) is feasible in \( T \). If an arbitrary edge is removed from every ring in \( T \) then there exists a call set \( S' \subseteq S \) with \( |S'| \geq |S|/3 \), which is feasible in the resulting tree.

**Proof:** Consider a feasible edge disjoint routing \( R \) of \( S \) in \( T \). Let us reroute the paths (while not changing the end points of the paths) in \( R \) such that for any ring \( C \) in \( T \) the rerouted paths do not pass through the edge removed from \( C \). These rerouted paths now lie on a tree obtained by removing the arbitrary edges (one from each ring in \( T \)) in \( T \). Since \( R \) was pairwise edge
disjoint the reroutings increased the number of paths through any edge to at most 2. Raghavan and Upfal, in [RU94], have shown that given a set of paths in an undirected tree such that through any edge at most \( L \) of these paths pass, there is an algorithm that can colour the paths with at most \( 3L/2 \) colours such that any two incident paths get different colours. Using this result, we can colour the set of rerouted paths using at most 3 colours such that paths that are incident on each other get different colours. Among the (at most) 3 sets of paths with the same colour, the one with maximum cardinality has at least one-third of the rerouted paths. The set of pairs of end vertices of the paths in this set is precisely the call set \( S' \) we require.

\[ \Box \]

For on-line MEDP in trees with diameter \( d \), \( O(\log d) \)-competitive randomized algorithms are given by Borodin and El-Yaniv in [BEY98, Sect. 13.5.2] and Awerbuch et al. in [AGLR94]. Here, we review the algorithm of Borodin and El-Yaniv, which they call TREE-AAP. The other algorithm will be discussed in the context of MEDPwPP in the next section. TREE-AAP itself is based on the algorithm by Awerbuch, Azar and Plotkin [AAP93].

**Algorithm TREE-AAP:** Let \( d \) be the diameter of the tree and \( \mu = 4d \). For each edge \( e \) in the tree, let \( L_0(e) = 0 \) and \( c_0(e) = (2 + \log d)(\mu L_0(e) - 1) \). Initially \( i = 1 \). Let the call \( r_i, i \in \{1, 2, \ldots, m\} \), with the unique path \( P(r_i) \) in the tree be presented. If \( \sum_{e \in P(r_i)} c_{i-1}(e) \leq d \), \( r_i \) becomes a candidate. If \( r_i \) is a candidate and \( P(r_i) \) does not intersect the route of a previously accepted call, accept it with probability \( \frac{1}{4(1 + \log d)} \). Otherwise, reject it. If \( r_i \) is not a candidate, set \( L_i(e) = L_{i-1}(e) + \frac{1}{2(1 + \log d)} \), for all \( e \in P(r_i) \) and \( L_i(e) = L_{i-1}(e) \), for all \( e \not\in P(r_i) \). Calculate \( c_i(e) = (2 + \log d)(\mu L_i(e) - 1) \), for all edges \( e \). Process the next call, \( r_{i+1} \).

We combine the Theorem 3.1.1 and an algorithm for on-line MEDP on trees to obtain an algorithm for on-line MEDP on trees of rings. The “best possible” tree after the edges are removed from the tree of rings is one that has the minimum possible diameter. A spanning tree with minimum diameter can be computed efficiently, even in arbitrary graphs [HT95].
3.2. The algorithm for on-line MEDPwPP

In the case of MEDPwPP, we cannot use the cut-one-link heuristic as we did for MEDP because in the worst case all pre-specified paths for the calls might use one of the deleted links. In this case, we cannot hope for a good competitive ratio with the above heuristic. Therefore, we need a different approach.

In order to motivate our algorithm, we briefly explain why randomization is necessary and why placing “roadblocks” on the edges of a rejected path is helpful. It is easy to observe that no deterministic algorithm for MEDPwPP can achieve a good competitive ratio even in chain networks: Consider a chain with $n$ nodes. Label the vertices $1, 2, \ldots, n$. Let the request sequence, specified as the end vertices of the requested paths, be $\{1, n\}, \{1, 2\}, \{2, 3\}, \ldots, \{n-1, n\}$. Any deterministic algorithm has to accept the first path presented to it. Otherwise, it will achieve a competitive ratio of $\infty$ on a request sequence that consists of this single path only (keep in mind that in the on-line scenario, the algorithm has no knowledge of future
Chapter 3. On-line Call Control

requests). But accepting the first path in the above instance precludes the other $n-1$ paths from being accepted, giving a competitive ratio of at least $n-1$ for any deterministic algorithm.

From the above, it is clear that only a randomized algorithm can hope to achieve a good competitive ratio for MEDPwPP. The basic idea of a randomized algorithm is to decide to accept or reject a path presented to it in a probabilistic manner. If the probability of accepting that path is positive and depends only on the paths that have been accepted previously (but not on previously rejected paths), an adversary could force the algorithm to accept the path by presenting it sufficiently many times. Therefore, the randomized algorithm needs to remember (in some way) a path that it has rejected once. In the MEDP algorithm for trees given in [AGLR94], this is achieved by placing so-called roadblocks on some of the edges of a rejected path (cf. Subsection 3.2.3). The roadblock on an edge causes any future path passing through it to be rejected immediately. While we use this idea for the tree of rings structure for the same reason, it has to be adapted in ways different from the one used for trees. Also, the topology of tree of rings presents various features in the analysis which are not encountered in trees. For example, it seems crucial to introduce the notion of crossing paths (see below) in order to carry out the analysis.

3.2.1 Terminology

In the following, we shall use calls and paths (pre-specified for calls) interchangeably. A path passes a ring if it contains at least one edge of the ring. A path touches a ring if it contains at least one vertex of the ring. We say that two paths cross if they share at least 2 vertices or if they pass through the same two consecutive rings. Note that any two paths that intersect also cross, but not vice versa (cf. Figure 3.1). If a path $p$ crosses a path $q$ that was presented earlier, the region of overlap of $p$ with $q$ is defined as the sub-path of $q$ between the extreme shared vertices of $p$ and $q$. If $p$ and $q$ cross and share only one vertex, then the region of overlap is just that shared vertex. Note that the region of overlap of $p$ with $q$ is not symmetric w.r.t. $p$ and $q$.

When we use terms like parent ring and ancestor ring, we refer to the underlying tree structure of the tree of rings $T$, rooted at an arbitrary ring. For a path $r_i$, define its least common ancestor ring (abbreviated lca-ring), $\text{lca}_\text{ring}(r_i)$, to be the ring $C$ such that $C$ contains at least one vertex of $r_i$ and no ancestor ring of $C$ contains a vertex of $r_i$. If a path $r_i$ passes more than one ring and the end points of $r_i$ are $u_i$ and $v_i$, then call the sub-path from $u_i$ to $\text{lca}_\text{ring}(r_i)$ the left side of $r_i$ and the sub-path from $v_i$ to $\text{lca}_\text{ring}(r_i)$ the right side of $r_i$. If $u_i$ or $v_i$ lie in the $\text{lca}_\text{ring}(r_i)$ then the corresponding
3.2. The algorithm for on-line MEDPwPP

3.2.2 Intersecting versus crossing

Let OPT represent an optimal set of pairwise edge disjoint paths among the given sequence of paths (calls). The optimality is meant in an off-line sense.

Lemma 3.2.1 The paths in OPT can be coloured with at most 3 colours such that no two paths with the same colour cross.

Proof: Consider the paths in OPT according to the descending order of the lca-rings of the paths. That is, paths that have the root ring as their lca-rings are considered first, then the paths that have children of the root ring as their lca-rings and so on. The order among paths that have the same ring as their lca-rings is as follows: Those that have at least one edge in the lca-ring come before those that share only a vertex with the lca-ring. For paths whose lca-rings are siblings, we order from left to right. If an order for a pair of paths cannot be deciphered from these rules then between them they can be ordered arbitrarily. We prove the claim by induction on the number of paths in OPT when arranged in the above order.

Basis Step: The first path in the above order does not cross any other coloured path. Thus, it can be coloured with one colour. Therefore, the claim holds for the basis step.

Induction Step: Consider a path $r_k$ in OPT. Assume that all paths preceding it in the above order can be coloured with at most 3 colours such that two paths that cross get different colours. Consider such a colouring. We will show how to extend this colouring such that all paths up to $r_k$ are coloured with at most 3 colours such that no two paths with the same colour cross. We consider the following cases:

Case 1: $r_k$ has edges only in $\text{lca}_\text{ring}(r_k)$.

$r_k$ can cross at most one of the paths preceding it, namely the one which contains all of the edges in the $\text{lca}_\text{ring}(r_k)$ not in $r_k$. In this case, $r_k$ can be
coloured with any of the two colours other than the one used to colour this crossing path.

Case 2: \( r_k \) has edges outside \( lca_{ring}(r_k) \).

If \( r_k \) crosses any of the paths preceding it then these paths necessarily have to touch \( lca_{ring}(r_k) \). The lca-ring of such a preceding path is either \( lca_{ring}(r_k) \) or its ancestor. Thus, \( r_k \) crosses these paths in one of two possible ways: either, along the \( lca_{ring}(r_k) \) and the left side of \( r_k \), or along the \( lca_{ring}(r_k) \) and the right side of \( r_k \). Of course, it is possible that just one of the preceding paths crosses both the left and right sides of \( r_k \). Since we consider an edge disjoint optimal set, at most two of the preceding paths could have crossed \( r_k \). Thus, \( r_k \) can be coloured with the third colour not used by these at most two crossing paths.

This covers all the cases for \( r_k \), and the induction step is proved.

Our on-line algorithm for MEDPwPP, presented in the next subsection, will in fact accept a set of pairwise non-crossing paths. Lemma 3.2.1 implies that there exists a set of pairwise non-crossing paths that contains at least one third of the paths in \( OPT \), so we lose at most a factor of 3 by considering non-crossing paths instead of edge disjoint paths.

### 3.2.3 The algorithm

When the algorithm rejects a path, it will sometimes place roadblocks on certain edges of the path, and two extra roadblocks above. If there is a roadblock on an edge, no path using that edge will be accepted later on. The sub-path of a rejected path between two consecutive roadblocks is called a segment.

The algorithm, which is an adaptation of the on-line algorithm for trees in [AGLR94], is defined as follows. Let \( r_i \) be the next path (call). If \( r_i \) crosses a previously accepted path, reject it. If it crosses a previously rejected path such that the region of overlap of \( r_i \) with the rejected path contains a roadblock, or if it passes through an extra roadblock, reject it as well. If \( r_i \) is not rejected by the above conditions, it becomes a candidate. Accept a candidate with probability 1/2. If a candidate is not accepted, reject it and place roadblocks on it as follows: Number the edges from one end of the candidate to the other 1, 2, 3, ... and so on. Choose an integer \( t \) (called level) uniformly at random from \([0, \lfloor \log T \rfloor + 1]\), where \( T \) is the maximum path length\(^1\) in \( T \). On each edge numbered \( i \cdot 2^{t_i} \), \( i = 1, 2, \ldots \), place a roadblock.

---

\( ^1 \)In arbitrary undirected graphs, it is \( \text{NP} \)-hard to find the length of a longest path. In
Figure 3.2: Roadblocks placed by a rejected candidate for \( l = 1 \).

Also, on the lca\(\text{ring}(r_i)\), place roadblocks on the edges that are incident on the extreme vertices of the candidate path in that ring but not contained in the candidate path. If the lca-ring has only one vertex of the candidate, then the roadblocks are placed on the two edges incident on this vertex in the ring. Call these additional roadblocks extra roadblocks. The roadblocks placed by a rejected candidate path if \( l \) is chosen to be 1 are illustrated in Figure 3.2.

### 3.2.4 Analysis of the algorithm

Consider any sequence of calls \( r_1, r_2, \ldots, r_m \). By Lemma 3.2.1, any optimal set of edge disjoint paths can be coloured with at most three colours such that no two paths with the same colour cross. Thus, there exists a set of paths with the same colour which has cardinality at least \( \frac{|OPT|}{3} \). Call this set \( OPT' \). We shall denote the paths in this set as “optimal calls” or “optimal paths” from now on. For the analysis, we distribute tokens to some of the optimal calls when a candidate is rejected. We maintain a subset of the optimal paths which is updated whenever a new call is processed. Initially,
we set $C_0 = OPT'$. When call $r_i$ is processed, we compute $C_i$ from $C_{i-1}$, $i = 1, 2, \ldots, m$, as follows:

Case 1: $r_i$ is not a candidate.
Set $C_i = C_{i-1}$.

Case 2: $r_i$ is an accepted candidate.
Remove from $C_{i-1}$ all optimal calls that cross $r_i$. Also, if there are optimal calls in $C_{i-1}$ that cross a previously rejected candidate $r_j$ in the same segment as $r_i$ crosses $r_j$ and that received their last token from $r_j$, remove them from $C_{i-1}$. Take the resulting set as $C_i$.

Case 3: $r_i$ is a rejected candidate.
The algorithm places roadblocks on some of the edges of $r_i$. These roadblocks divide the path into segments. In each segment, we distribute a token to at most one optimal call that crosses $r_i$ in that segment. Define the level of an optimal call in $C_{i-1}$ that crosses $r_i$ as $j$ if the region of overlap of it with $r_i$ does not contain a roadblock when the roadblocks are placed $2^j$ apart, but contains one when placed $2^{j-1}$ apart. If the optimal call crosses $r_i$ and shares only one vertex with it, then its level is defined as 0. Let $l$ be the level randomly chosen for $r_i$. Then it is easy to see that there are at most two optimal calls in each segment of $r_i$ that have level $l$. If there is only one optimal call, give a token to it. If there are two, give a token to either of the optimal calls with probability $1/2$. Define $C_i$ as in Case 2. In addition, return to $C_i$ all optimal calls that received a token from $r_i$. If any of the optimal calls pass through the extra roadblocks placed by $r_i$, remove them from $C_i$.

Let $a^* = |OPT'|$. Let $a$ and $c$ be the number of accepted candidates and candidates respectively. Let $t$, $b$ and $f$, respectively, represent the number of optimal calls that receive a token, the number of optimal calls that were removed by extra roadblocks, and the number of optimal calls that neither crossed any previous candidate nor were removed by an extra roadblock. $a, c, t, b$ and $f$ are random variables.

**Lemma 3.2.2** $E[a] = \frac{E[c]}{2}$

**Proof:** A candidate is accepted with probability $1/2$. Therefore,

$$E[a] = \sum_{i=1}^{m} Pr(r_i \text{ is a candidate}) \cdot Pr(r_i \text{ is accepted} | r_i \text{ is a candidate})$$

$$= \frac{1}{2} \cdot \sum_{i=1}^{m} Pr(r_i \text{ is a candidate}) = \frac{E[c]}{2}.$$

$\square$
Lemma 3.2.3 \( E[t + b + f] = \Omega\left(\frac{|OPT|}{\log \ell_T}\right) \)

**Proof:** Consider any optimal call \( \rho \). Consider the event that it was not blocked by an extra roadblock, but crossed other candidates. Let \( r_i \) be the first candidate that crossed \( \rho \). \( \rho \) would receive a token from this candidate if (a) \( r_i \) is rejected, (b) the level assigned to \( r_i \) is its level, and (c) if there were another optimal call at the same level that crosses \( r_i \) in the same segment, it is not given a token. The probability of event (a) is \( 1/2 \), of event (b) is \( 1/(\log \ell_T + 2) \) and of event (c) is \( 1/2 \). Thus, the probability that \( \rho \) receives a token is at least \( \frac{1}{4(\log \ell_T + 2)} \). If \( \rho \) is blocked by an extra roadblock or if it does not cross any candidate, then it contributes 1 to the sum \( b + f \). Now,

\[
E[t + b + f] \geq \sum_{\rho: \rho \in OPT} \frac{1}{4(\log \ell_T + 2)} = \frac{\alpha^*}{4(\log \ell_T + 2)} \geq \frac{|OPT|}{12(\log \ell_T + 2)} = \Omega\left(\frac{|OPT|}{\log \ell_T}\right)
\]

Hence, the lemma follows.

\( \square \)

Lemma 3.2.4 A rejected candidate that gives a token to an optimal path \( \rho \) that is not removed by the extra roadblocks placed by it must touch \( \text{lca}_{\text{ring}}(\rho) \).

If a candidate \( r_i \) crosses a previously rejected candidate then the previously rejected candidate must touch \( \text{lca}_{\text{ring}}(r_i) \).

**Proof:** Since a rejected candidate, call it \( r_j \), places two extra roadblocks in \( \text{lca}_{\text{ring}}(r_j) \), neither an optimal path \( \rho \) that received a token from it and was not removed by the extra roadblocks placed by it nor a later candidate \( r_i \) that crosses it can pass through any ancestor ring of \( \text{lca}_{\text{ring}}(r_j) \). Now, consider \( r_i \). (The proof for \( \rho \) is analogous.) The crossing between \( r_j \) and \( r_i \) occurs somewhere in the sub-tree rooted at \( \text{lca}_{\text{ring}}(r_j) \). Therefore, \( \text{lca}_{\text{ring}}(r_i) \) is either \( \text{lca}_{\text{ring}}(r_j) \) or its descendant. In the former case, we are done. In the latter case, recall that the graph \( T \) in which the paths lie is a tree of rings. Hence, any path that originates from the sub-tree rooted at \( \text{lca}_{\text{ring}}(r_i) \) and passes through to \( \text{lca}_{\text{ring}}(r_j) \) has to pass via \( \text{lca}_{\text{ring}}(r_i) \). Thus, \( r_j \) has to touch \( \text{lca}_{\text{ring}}(r_i) \) after crossing \( r_i \), as claimed in the lemma.

\( \square \)

Now we want to bound the number of paths with tokens that are removed from \( C_{i-1} \) by a candidate \( r_i \).

Let \( r_i \) be a candidate. If \( r_i \) uses edges in \( \text{lca}_{\text{ring}}(r_i) \), then let \( A \) and \( B \) be the two edges in \( \text{lca}_{\text{ring}}(r_i) \) that are incident to vertices of \( r_i \) but that are not contained in \( r_i \), and let \( C \) and \( D \) be the edges in \( \text{lca}_{\text{ring}}(r_i) \) that are on \( r_i \) and share an endpoint with \( A \) or \( B \) (see Figure 3.3, left-hand side). If \( r_i \) contains
only one vertex $v$ in $\text{lca}_\text{ring}(r_i)$, then let $E$ and $F$ be the edges incident to $v$ that are used by $r_i$, and let $G$ and $H$ be the other two edges incident to $v$ that are contained in rings through which $r_i$ passes (see Figure 3.3, right-hand side).

**Lemma 3.2.5** Let $r_i$ be a candidate. If $r_i$ uses edges in $\text{lca}_\text{ring}(r_i)$, then any previously rejected candidate $r_j$ that crosses $r_i$ contains $A$, or contains $B$, or contains $C$ and $D$. If $r_i$ uses only one vertex in $\text{lca}_\text{ring}(r_i)$, then any previously rejected candidate $r_j$ that crosses $r_i$ contains $E$, or contains $F$, or contains $G$, or contains $H$. Furthermore, the edge in $A, B, C, D$ or $E, F, G, H$ contained in $r_j$ lies in the segment of $r_j$ where $r_i$ crosses $r_j$.

**Proof:** Case 1: $\text{lca}_\text{ring}(r_i)$ contains at least one edge of $r_i$.

By Lemma 3.2.4, any previously rejected candidate $r_j$ that crosses $r_i$ touches $\text{lc}_\text{ring}(r_i)$. In fact, $r_j$ must use edges in $\text{lc}_\text{ring}(r_i)$, because otherwise its extra roadblocks would have led to the immediate rejection of $r_i$. If $r_j$ does not use $A$ or $B$, it must use all edges on the sub-path of $r_i$ from $C$ to $D$, because otherwise again its extra roadblocks would have led to the rejection of $r_i$.

Case 2: $\text{lc}_\text{ring}(r_i)$ has no edges of $r_i$ in it.

Let $v$ be the vertex in $\text{lc}_\text{ring}(r_i)$ that is contained in $r_i$. Any previously rejected candidate $r_j$ that crosses $r_i$ must touch $v$ (by Lemma 3.2.4) and must use edges in one of the two rings containing $v$ through which $r_i$ passes. But then $r_j$ must reach $v$ through one of the edges $E, F, G,$ or $H$.

In both cases, it is easy to see that the respective edge must lie in the segment of $r_j$ where $r_i$ crosses $r_j$. Otherwise, the region of overlap of $r_i$ with $r_j$ would contain a roadblock.
Lemma 3.2.5 shows that the set of all previously rejected candidates that cross the current candidate $r_i$ can be grouped into at most four different classes such that if $r_j, r_k$ are in the same class then $r_j$ crosses $r_k$ in the same segment of $r_k$ as $r_i$ crosses $r_k$. For each class, there can be at most one optimal path in $C_{i-1}$ that has received its last token from that segment of a previously rejected candidate in that class. Therefore, we have the following lemma.

**Lemma 3.2.6** At most 4 optimal paths must be removed from $C_{i-1}$ because each of them received its last token from a previously rejected candidate and crossed the rejected candidate in the same segment as the one crossed by $r_i$.

If candidate $r_i$ itself is an optimal call with token then there can be no other calls in $C_{i-1}$ with tokens that cross it. Let us assume that $r_i$ is not a call with a token. The calls with tokens that do not satisfy the conditions of Lemma 3.2.6 and are removed by $r_i$ are those that cross it. Therefore, it remains to bound the following types of optimal paths in $C_{i-1}$ with tokens that cross $r_i$:

(type I) optimal path $p$ received its last token from a rejected candidate $r_j$ but $r_i$ does not cross $r_j$ at all.

(type II) optimal path $p$ received its last token from a rejected candidate $r_j$ but $r_i$ crosses $r_j$ in a segment different from the one in which $p$ crosses $r_j$.

**Lemma 3.2.7** There can be at most 2 optimal paths in $C_{i-1}$ of type I or II that use edges in $lca_{ring}(r_i)$.

**Proof:** If $r_i$ has no edges in $lca_{ring}(r_i)$ there can be at most 2 optimal paths that can cross it and use edges in $lca_{ring}(r_i)$; namely those paths that use the two edges incident on the vertex of $r_i$ in $lca_{ring}(r_i)$. Let $r_i$ use edges in $lca_{ring}(r_i)$. Consider the edges $A$ and $B$. Any optimal path of type I or II that uses edges in $lca_{ring}(r_i)$ can be associated to one of these edges as follows. Any such optimal path either uses at least one of the edges $A$ or $B$, say $A$ or the rejected candidate that gave it the last token uses at least one of the edges $A$ or $B$, say $A$. Now, associate the optimal path to edge $A$. In the first case, there can be at most one optimal path using edge $A$. In the second case, the rejected candidate gives token to at most one optimal path in the segment containing the edge $A$. The lemma now easily follows. 

**Lemma 3.2.8** There can be at most 4 optimal paths of type I or II in $C_{i-1}$ that cross $r_i$ either on its left or on its right side.
Proof: Consider optimal paths of type I or II for which the region of overlap with $r_i$ lies either on the left or right side of $r_i$. We bound the number of type I or II paths on each side to be at most 2. We prove the bound for paths crossing the left side of $r_i$. A symmetric argument holds for the right side.

Let $\rho$ be an optimal path of type I or II that crosses $r_i$ on its left side and whose lca-ring is the furthest from the root among all such optimal calls. Let $r_j$ be the previously rejected candidate that gave the last token to $\rho$.

Case 1: $\text{lca}_\text{ring}(\rho)$ is a descendant of $\text{lca}_\text{ring}(r_i)$.

In this case, $r_i$ crosses $r_j$ and $\rho$ is a type II path. Since the segment of $r_j$ in which $\rho$ crosses $r_j$ is different from the segment of $r_j$ in which $r_j$ reaches $\text{lca}_\text{ring}(r_i)$, there must be at least one roadblock on $r_j$ separating these two segments. Furthermore, this roadblock must in fact be on an edge in $\text{lca}_\text{ring}(\rho)$, so the situation must be exactly as shown in Figure 3.4: $r_i$ begins at $w$ or enters $\text{lca}_\text{ring}(\rho)$ at $w$ and leaves $\text{lca}_\text{ring}(\rho)$ at $u$; $r_j$ meets $r_i$ for the first time at vertex $u$; $\rho$ crosses $r_j$, meets $r_i$ for the first time at vertex $w$, and either uses the edge incident to $w$ in $\text{lca}_\text{ring}(\rho)$ that is contained in $r_i$ (Figure 3.4, left-hand side), or uses edges in the ring below $\text{lca}_\text{ring}(\rho)$ where $r_i$ comes from (Figure 3.4, right-hand side); and there is at least one roadblock on $r_j$ somewhere between the crossing with $\rho$ and the vertex $u$.

We note that there cannot be any other type II path whose lca-ring is equal to $\text{lca}_\text{ring}(\rho)$, because any such path would also have to contain the edge in $\text{lca}_\text{ring}(\rho)$ that is not in $r_i$ and that is incident on the vertex labelled $w$ in Figure 3.4. So assume that there is a type I or II optimal path $\rho'$ crossing the left side of $r_i$ and whose lca-ring is an ancestor of $\text{lca}_\text{ring}(\rho)$. Let $u'$ be the vertex of $\text{lca}_\text{ring}(\rho')$ where $r_i$ enters that ring from below, and let $v'$ be the
Figure 3.5: The two possibilities for $p'$ together with the paths traced by candidates $r_i$ and $r_j$. 

vertex where $r_i$ leaves that ring going up. Then $r_j$ must also be incident on both $u'$ and $v'$. Furthermore, $p'$ must either contain both edges incident to $u'$ in $\text{lca}_{\text{ring}}(p')$ or it must use edges in $\text{lca}_{\text{ring}}(p')$ and the child ring of $\text{lca}_{\text{ring}}(p')$ through which $r_i$ passes. See Figure 3.5. In both these cases, $p'$ crosses $r_j$ in the same segment in which $r_j$ contains one of the edges named $A$, $B$, etcetera. At the time when $r_j$ was processed by the algorithm, at most one optimal path with tokens crossing that segment of $r_j$ was not removed from $C_{i-1}$. Therefore, at most one optimal path of type I or II that crosses $r_i$ on the left side (in addition to the optimal path $p$) can still be in $C_{i-1}$ when $r_i$ is processed.

Case 2: $r_i$ contains at least one vertex in $\text{lca}_{\text{ring}}(r_i)$.

There can be at most two edge disjoint paths that cross $r_i$ on its left side and contain a vertex of $\text{lca}_{\text{ring}}(r_i)$.

Thus, the number of optimal paths in $C_{i-1}$ of type I or II that cross $r_i$ on its left side is at most 2. Similarly, the number of optimal paths in $C_{i-1}$ of type I or II that cross $r_i$ on its right side is at most 2. This gives us the bound of 4 as claimed in the lemma.

\[ E[c] \geq \frac{E|q|}{12}. \]

\textbf{Proof:} We show that any candidate $r_i$ removes at most 12 paths with tokens. If $r_i$ itself is an optimal path with token then no other optimal path can cross it. Therefore, assume that $r_i$ is not an optimal path with token. By
Lemma 3.2.6, at most 4 paths with tokens are removed because they received their last token from previously rejected candidates and crossed the rejected candidates in the same segments in which \(r_i\) crossed them. Any other path with token removed by \(r_i\) must fall into one of the two types: type I or II. Lemma 3.2.7 bounds the number of type I or II calls that use edges in \(\text{lca}_{\text{ring}}(r_i)\) to be 2. The only other calls with tokens that will be removed are those of type I or II that cross \(r_i\) either on its left or on its right side (and do not use edges of \(\text{lca}_{\text{ring}}(r_i)\)). Lemma 3.2.8 shows that there are at most 4 such calls. Finally, if \(r_i\) is rejected, at most 2 paths with token are removed due to the extra roadblocks. In total, at most 12 paths with tokens are removed.

\[\Box\]

**Theorem 3.2.10** For on-line MEDPwPP on trees of rings whose simple paths have a maximum length of \(t_T\), there exists a randomized algorithm that achieves a competitive ratio of \(O(\log t_T)\).

**Proof:** Every rejected candidate places two extra roadblocks. Hence, \(c \geq \frac{b}{2}\). Also, every optimal call that was not removed by an extra roadblock and did not cross any candidate must have been a candidate itself. Therefore, \(c \geq f\).

From this and the above lemmas we get:

\[E[a] = \frac{E[c]}{2} \geq \frac{1}{2} \max\{\frac{E[d]}{12}, \frac{E[f]}{2}\} = \Omega(E[t + b + f]) = \Omega(\frac{\text{OPT}}{\log t_T})\]

\[\Box\]

### 3.3 A lower bound for randomized algorithms

To see that the randomized algorithms presented above are optimal up to constant factors, we present lower bounds on randomized algorithms for these problems. We follow the method illustrated in [AGLR94]. We restrict ourselves to the non-preemptive setting where a call once accepted cannot be rejected in favour of a later call. First we show that on a chain of \(n\) nodes, no randomized algorithm can achieve a competitive ratio better than \(O(\log n)\) for the MEDP problem (Recall that MEDP and MEDPwPP are the same problems for chains). This proves the lower bound for MEDPwPP on trees of rings by considering the longest path in it and requesting calls with routes lying only on this path. The method for chains can be extended to rings on \(n\) nodes for the MEDP problem on it. The basic principle behind showing the lower bound is the one due to von Neumann [MR95, pp. 31-35]: Give a
probability distribution over the input such that any deterministic algorithm cannot beat the lower bound.

Accordingly, we show a distribution on a set of pre-defined calls on a chain with \( n+1 \) nodes, where \( n \) is a power of 2. Then, prove that any deterministic algorithm that has this set of calls input in an on-line manner according to the distribution, can accept at most an \( O(\log n) \) fraction of the number of calls accepted by an optimal off-line algorithm. Let the chain be \( L = (V, E) \) with \( V = \{1, 2, \ldots, n+1\} \) and \( E = \{(i, i+1) : i = 1, 2, \ldots, n\} \). The input sequence of calls is \( S_0, S_1, \ldots, S_{\log n} \), where \( S_0 = \{(1, n+1)\}, S_1 = \{(1, n/2+1), (n/2+1, n+1)\}, \ldots, S_i = \{(1, n/2^i+1), (n/2^i+1, n/2^{i-1}+1), \ldots, (n^{2i-1}+1, n+1)\}, \ldots, S_{\log n} = \{(1, 2), (2, 3), \ldots, (n, n+1)\} \). The distribution on the request sequence is defined as follows: With probability \( 2^{-i}, i = 0, 1, \ldots, \log n \), the sequence of calls presented is \( S_0, S_1, \ldots, S_i \). With probability \( \frac{1}{2n} \) the sequence is empty. Given the sequence \( S_0, S_1, \ldots, S_i, i = 0, 1, \ldots, \log n \), an optimal off-line algorithm maximizes the number of accepted calls by accepting all calls in \( S_i \), i.e. \( 2^n \) calls. Thus, the expected number of calls accepted by the off-line algorithm is \( \sum_{i=0}^{\log n} 2^{-i-1} 2^n = \frac{\log n + 1}{2} \).

Next, we consider how a deterministic algorithm \( A \) performs on the input sequence. Assume that on some input sequence, \( A \) has already decided upon calls up to (and excluding) those in \( S_i, i = 0, 1, \ldots, \log n \). The calls in \( S_i \) all have lengths of \( n/2^i \). Thus, if there are \( k \) edges not used up by \( A \) on the chain then at most \( k/(n/2^i) \) calls from \( S_i \) can be accepted. Denote by \( A(i, k) \) the maximum expected number of calls accepted from \( S_i, S_{i+1}, \ldots \), given that the sequence is terminated somewhere after \( S_i \) and at most \( k \) edges are not used up by calls in \( S_0, S_1, \ldots, S_{i-1} \). Here, the maximum is taken over all possible ways of accepting calls presented before the calls in \( S_i \) are presented. Note that \( A(\log n, k) \leq k \). We can write down the following recurrence relation,

\[
A(i, k) \leq \max_{\frac{k}{n/2^i} \leq i} \left\{ \frac{l + \frac{1}{2} A(i + 1, k - l \cdot (n/2^i))}{l} \right\}
\]

It is not too hard to see why the recurrence holds. The left hand side is the maximum expected number of calls accepted from \( S_i, S_{i+1}, \ldots \) as defined above. The right hand side groups the accepted calls from these sets into two types; those accepted from \( S_i \) and those accepted from \( S_{i+1}, S_{i+2}, \ldots \). The probability that calls from \( S_{i+1} \) and later are presented given that calls from \( S_i \) are presented is 1/2. The first term on the right hand side is the number of calls that are accepted from \( S_i \). For each of the different possibilities, the second term is the maximum expected number of calls that can be accepted from \( S_{i+1}, S_{i+2}, \ldots \) multiplied by the probability that they are presented to the algorithm. An induction argument (see [AGLR94]) shows that \( A(i, k) \leq \)
$k/(n/2^i)$, for all $i, k$. In particular, this gives us that the maximum expected number of calls that $A$ accepts is at most 1 ($\geq A(0, n)$). As noted above the expected number of calls accepted by an optimal off-line algorithm is $(\log n + 1)/2$. Therefore, we have the following theorem from [AGLR94].

**Theorem 3.3.1** No randomized algorithm for MEDP (or MEDPwPP) on chains with $n$ nodes can have a competitive ratio better than $O(\log n)$.

For rings, arguments along lines similar to the above go through. We specify only the input sequence and the probability distribution on it. Consider the ring $C = (V, E)$ with $V = \{0, 1, ..., n - 1\}$ and $E = \{\{i, i + 1(\text{mod } n)\} : i = 0, 1, ..., n - 1\}$, where $n$ is a power of 2. The input request sequence is $S_1, S_2, ..., S_{\log n}$, where $S_i$ is defined as $S_i = \{\{0, n/2^i\}, \{n/2^i, n/2^{i-1}\}, ..., \{n^{\lfloor \log n \rfloor - 1}, 0\}\}$. The distribution on the input is to present all calls from $S_1, ..., S_i$ with probability $2^{-i}, i = 1, 2, ..., \log n$ and to present no calls at all with probability $1/n$. The difference between the problems on chain and ring lies in the fact that in rings calls can be routed in 2 different ways. Even so, for the above sequence of calls the minimum of the lengths of the two possible routes of any call in $S_i$ is at least $n/2^i$. The recurrence for chains is, thus, valid for rings as well. Calculations, like the ones above, confirm that the expected number of calls accepted by an optimal off-line algorithm is $\log n$ and the maximum expected number of calls accepted by a deterministic algorithm $A$ on the input distribution is at most 2 ($\geq A(1, n)$). Thus we have,

**Theorem 3.3.2** No randomized algorithm for MEDP on rings with $n$ nodes can have a competitive ratio better than $O(\log n)$. 
Call Control: FPT Results

The main results of this chapter are fixed parameter tractable algorithms for the parameterized version of off-line MEDP problems on undirected and bidirected trees of rings. We show that when the paths are pre-specified the problem is fixed parameter tractable even for arbitrary graphs.

From an optimization point of view the following two problems are equivalent: (1) find a maximum number of feasible calls that can be accepted and (2) find a minimum number of calls that should be rejected so that the remaining set of calls is feasible. For approximation algorithms, however, it makes a difference. Consider approximation algorithms for problems (1) and (2) on some topology that achieve an approximation ratio of 2, say. If we run the algorithms on an instance with 10 calls where the optimal algorithm accepts 8 calls, the approximation algorithm for problem (1) will accept 4 in the worst case and the algorithm for problem (2) will accept 6 in the worst case. Thus, if we know in advance that a lot more calls will be accepted than rejected then it makes sense to formulate the problem as in (2) instead of (1).

Typically, network resources are over-provisioned so that rejections of calls occur infrequently. Blum et al. [BKK01], arguing as above, consider the objective of minimizing rejections for on-line, preemptive, uniform call control problems. The routes for calls are fixed in the input. They obtain algorithms with good competitive ratios; (i) for chains and rings with arbitrary capacities they show 2-competitive algorithms, (ii) for general graphs with bounded edge capacities of c, they provide \((c + 1)\)-competitive algorithms, and (iii) for general graphs on \(m\) edges with arbitrary capacities, they showed an \(O(\sqrt{m})\)-competitive algorithm. For the off-line, pre-routed, uniform call control problem on an arbitrary graph with \(m\) edges, an \(O(\log m)\)-ratio randomized algorithm is given in [BKK01] for the objective of minimizing rejections.

When an \(\mathcal{NP}\)-hard problem is known to have a “parameter” that always takes small values, a natural line of investigation that opens up is that of parameterized complexity. Given that the number of calls that are rejected is small, we take it as a parameter to the call control problem and ask the
question: Is there an algorithm that given a parameter $k$ and an input instance of a call control problem (of size $|I|$) decides whether there exists a feasible call set obtained by rejecting at most $k$ of the input calls in time $O(f(k)\text{poly}(|I|))$. Such an algorithm, if it exists, is called a fixed parameter tractable (FPT) algorithm.

In Section 4.1 we study the off-line MEDPwPP-$k$ problem on arbitrary (undirected) graphs. In Section 4.2, we offer results for the off-line MEDP-$k$ problem on bidirected and undirected trees of rings. For all the above problems, we show FPT algorithms.

A standard technique for deriving FPT algorithms is the method of bounded search trees [DF99]. Abstractly, it works as follows. The given parameterized problem is the root of a search tree. Every other node in the search tree corresponds to a “smaller” instance of the original parameterized problem (for instance, the parameter at a node is smaller than that of its parent). The search tree satisfies the condition that given “solutions” to all the children of a node, the node’s “solution” itself can be computed in polynomial time in the size of the original input. Further, “solutions” to the leaf nodes in the tree can be found in polynomial time. An FPT algorithm for the parameterized problem can then be formulated if the size of the tree is bounded by a (arbitrary) function of the parameter.

More concretely, for our call control problems we do the following: Let the call control problem on a topology $G$ with input set of calls $S$ and the parameter $k$ be given. Either, we show that $S$ is feasible in $G$ or we identify a set $S_{\text{rej}} \subseteq S$ of calls (in polynomial time). The set $S_{\text{rej}}$ is constructed such that if a feasible solution exists that rejects a set of at most $k$ calls, there is a feasible solution that rejects a set of at most $k$ calls and contains at least one of the calls in $S_{\text{rej}}$. If $k = 0$, we are done at this point. Otherwise, we branch on the search tree for each call $r \in S_{\text{rej}}$ (labeling the edge with $r$) and recursively ask whether for the calls in $S \setminus \{r\}$ there exists a feasible solution that rejects at most $k - 1$ calls. If the cardinality of $S_{\text{rej}}$ is bounded by a function of $k$ (say $f(k)$) then we have an FPT algorithm. If at any node, whose level is at most $k$, we have a feasible solution, then we have a solution for our original problem; reject those calls that appear as labels on the path from that node to the root. Otherwise, we can declare that there is no feasible solution to the problem at the root. The number of nodes we will have to inspect is at most $O(f(k)^k)$. Since, the work at each node is polynomial in the size of the input, the algorithm that exhaustively searches the search tree up to level $k$ runs in time $O(f(k)^k\text{poly}(|I|))$. We, therefore, have an FPT algorithm for our problem. The function $f(k)$ will be a small constant in all our cases.
4.1 Off-line MEDPwPP-$k$ on arbitrary graphs

The result of this section is exceedingly simple but is helpful in understanding the FPT algorithms of the following sections. The input to the off-line MEDPwPP-$k$ problem, we consider, is (i) an arbitrary graph $G = (V, E)$ whose edges have unit edge capacities, (ii) a set of calls $S = \{r_1, r_2, \ldots, r_m\}$ with unit bandwidth requirements and a unique path for each call and (iii) an integer parameter $k \geq 0$. The goal is to find a subset $S' \subseteq S$ such that $|S'| \leq k$ and $S \setminus S'$ is feasible in $G$ for the given set of routes, if it exists, or declare that no such set $S'$ exists.

Instead of talking about calls we will talk about their paths. It is trivial to observe that if two paths intersect at an edge then any solution to the problem must reject one of these two paths. The FPT algorithm is now straightforward. If there is no edge through which two paths pass the given set is feasible and we stop (This check can be performed in $O(|E|)$ time. This can be seen as follows. If the total number of edges (counting multiplicities) in the paths is at most $|E|$ then the running time is evident. Else, by the pigeon hole principle we will find an edge through which 2 paths pass at least by the time we examine the $(|E| + 1)$-th edge in the paths). Otherwise, we find an edge through which two paths pass, say $r_{j1}, r_{j2}$. If $k = 0$, we can declare that $S$ is not feasible and stop. If $k \geq 1$, we recurse for each path $r_{jl}, l \in \{1, 2\}$ and ask if there is a subset $S'_l \subseteq S \setminus \{r_{jl}\}$ such that $|S'_l| \leq k - 1$ and $S \setminus (S'_l \cup \{r_{jl}\})$ is feasible. The search tree so constructed is a binary tree. At each node we do a polynomial amount of work to check its feasibility or to find its two children. If at any node, with level at most $k$, we have feasibility, we can output the set $S'$ to be the set of paths on the recursive branches from that node to the root. If no node with level at most $k$ has a feasible solution then we can declare that no feasible solution exists by rejecting at most $k$ paths. The running time of the FPT algorithm is $O(2^k|E|)$. The running time can be significantly improved by noting that the parameterized problem above is equivalent to finding a vertex cover of size at most $k$ in the conflict graph of the given set of paths (The conflict graph of a set of paths is one whose vertex set will find an edge through which 2 paths pass at least by the time we examine the $(|E| + 1)$-th edge in the paths). Otherwise, we find an edge through which two paths pass, say $r_{j1}, r_{j2}$. If $k = 0$, we can declare that $S$ is not feasible and stop. If $k \geq 1$, we recurse for each path $r_{jl}, l \in \{1, 2\}$ and ask if there is a subset $S'_l \subseteq S \setminus \{r_{jl}\}$ such that $|S'_l| \leq k - 1$ and $S \setminus (S'_l \cup \{r_{jl}\})$ is feasible. The search tree so constructed is a binary tree. At each node we do a polynomial amount of work to check its feasibility or to find its two children. If at any node, with level at most $k$, we have feasibility, we can output the set $S'$ to be the set of paths on the recursive branches from that node to the root. If no node with level at most $k$ has a feasible solution then we can declare that no feasible solution exists by rejecting at most $k$ paths. The running time of the FPT algorithm is $O(2^k|E|)$. The running time can be significantly improved by noting that the parameterized problem above is equivalent to finding a vertex cover of size at most $k$ in the conflict graph of the given set of paths (The conflict graph of a set of paths is one whose vertex set is the set of paths and the edge set is the set of pairs of paths that share an edge). The fastest known FPT algorithm for the parameterized vertex cover problem on a graph $G' = (V', E')$ and parameter $k$ runs in time $O(|V'|k + 1.2852k^2)$ [CKJ01]. The conflict graph for a given set of paths $R$ on a graph $G = (V, E)$ can be built in $O(|E| + |R|^2|V|)$ time.

**Theorem 4.1.1** Consider an instance of off-line MEDPwPP-$k$ given by an arbitrary graph $G = (V, E)$ with unit edge capacities, a set of calls $S$ with pre-specified paths and unit bandwidth requirements and a parameter $k \geq$
There exists a fixed parameter tractable algorithm that solves it in time $O(|S|k + 1.2852^k + |S|^2|V| + |E|)$.

In [AEHS03], we present a generalization of the above FPT result when the edge capacities are bounded by a constant. However, if the edge capacities are arbitrarily large then the problem is proved to be $W[2]$-hard. $W[t]$-hardness, with $t > 1$, for a problem is considered to be strong evidence that there exists no FPT algorithm for it [DF99]. We conclude this section by stating some of the parameterized tractability and intractability results for arbitrary graphs in [AEHS03].

**Theorem 4.1.2** Consider an instance of off-line PCC-$k$ given by an arbitrary (edge capacitated, directed or undirected) graph $G = (V, E, c)$, a set of calls $S$ with pre-specified paths and unit bandwidth requirements and a parameter $k \geq 0$.

(a) If the edge capacities are bounded by a constant $c$ then there exists a fixed parameter tractable algorithm that solves the off-line PCC-$k$ problem in time $O((c + 1)^k|E|)$.

(b) If the edge capacities are arbitrarily large then the off-line PCC-$k$ problem is $W[2]$-hard even when $G$ is restricted to be a series-parallel graph.

### 4.2 Off-line MEDP-$k$ on trees of rings

This is the only section of the thesis that considers the call control problem on directed graphs in addition to undirected graphs. We note here how the call control problem changes in a directed graph. A call $r$ in a directed graph $G$ is given by an ordered pair $(u, v)$ of its end vertices $u, v$, in addition to the other parameters. A route for the call $r$ is a directed path in $G$ from $u$ to $v$. Feasibility of a routing is verified by checking that the capacity constraints are obeyed at every directed edge in the graph. The off-line MEDP-$k$ problems on trees of rings are given by a (undirected or bidirected) tree of rings $T = (V, E)$ with unit capacity edges, a set of calls $S = \{r_1, r_2, \ldots, r_m\}$ with unit bandwidth requirements and an integer parameter $k$. The objective is to find a subset of calls $S' \subseteq S$ of cardinality $|S'| \leq k$ such that the remaining call set $S \setminus S'$ is feasible in $T$ and to give a feasible routing for this set or declare no such set $S'$ exists. This problem is known to be $\mathcal{NP}$-complete for both undirected and bidirected trees of rings [Erl01]. We shall refer to a call as a pair of its end vertices, as all other parameters are common to the set of calls. In the undirected case, the order is immaterial. We use the terms edges and arcs interchangeably for directed graphs, in the discussion below.
4.2.1 The undirected case

Let an instance of off-line MEDP-$k$ on an undirected tree of rings $T$ with unit edge capacities be given. We employ the technique of bounded search trees. Before we present the details of the algorithm, we discuss some important properties about calls and their routes in trees of rings.

For any call $(u,v)$ in a tree of rings, all undirected paths from $u$ to $v$ contain edges of the same rings. For each ring that a path from $u$ to $v$ passes through (i.e., contains an edge of that ring), the vertex at which the path enters the ring (or begins) and the vertex at which the path leaves the ring (or terminates) is uniquely determined by the endpoints of the path. Thus, a call $(u,v)$ in a tree of rings can be viewed as a combination of sub-calls in all rings that a path from $u$ to $v$ passes through. Consequently, a set of calls can be routed along edge disjoint paths if and only if all sub-calls of the calls can be routed along edge disjoint paths in the individual rings of the tree of rings. Therefore, before we tackle the problem in trees of rings, we need to investigate conditions for a set of calls in a ring being routable along edge disjoint paths.

Let $C$ be a ring. Imagine $C$ drawn as a circle in the plane, with its vertices distributed at equal distance along the circle. A (sub-)call $(u,v)$ between two vertices in $C$ can be represented as a straight line segment joining $u$ and $v$. We call these line segments chords and use the terms chord and call interchangeably if no confusion can arise.

Two chords are said to be parallel if they do not intersect, if they intersect at a vertex in $C$, or if they are identical (see Figure 4.1). Note that if two chords are parallel then we can assign edge disjoint paths to the corresponding calls, and that these paths are uniquely determined.

A 2-cut in a ring $C$ is a pair of edges in the ring. A call crosses a 2-cut if each of the two possible paths connecting the endpoints of the call contains exactly one of the edges in it.

If $p$ is a path in $C$ then the path formed by the set of remaining edges in $C$ (i.e., by the edges in $C \setminus p$) is called its complementary path.
4.2. Off-line MEDP-fc on trees of rings

The FPT algorithm

The following lemma will be the main ingredient in deriving the FPT algorithm for trees of rings.

**Lemma 4.2.1** Given a ring $C$ and a set $S$ of calls in $C$, the calls in $S$ can be routed along edge disjoint paths if and only if (i) the chords of the calls in $S$ are pairwise parallel and (ii) no 2-cut is crossed by three calls.

**Proof:** If $|S| = 1$, the lemma is trivially true. So assume that $|S| \geq 2$.

Assume that it is possible to route $S$ along edge disjoint paths. Consider a feasible edge disjoint routing of $S$. Let $(u, v)$ and $(w, x)$ be two arbitrary calls in $S$. Let $p_1$ and $p_2$ be the paths assigned to them in the edge disjoint routing. Since $p_1$ is edge disjoint from $p_2$, $p_2$ lies in the complementary path of $p_1$. Clearly, the chords of the calls $(u, v)$ and $(w, x)$ must be parallel. This proves (i). Any call that crosses a 2-cut must be routed through one of the two edges of the 2-cut. If there were three calls crossing the same 2-cut, one of the two edges of the 2-cut would be used by at least two paths. Therefore, (ii) must hold as well.

Conversely, let $S$ satisfy (i) and (ii). We show that there is an edge disjoint routing for the calls in $S$. Consider any two calls $(u, v)$ and $(w, x)$ in $S$. By (i), they are parallel and can be assigned edge disjoint paths, say, $p_1$ and $p_2$. Furthermore, the paths $p_1$ and $p_2$ are uniquely determined. If $|S| = 2$, we are done. Otherwise, let $(y, z)$ be an arbitrary third call in $S$. Note that the non-trivial connected components of $G \setminus (p_1 \cup p_2)$ are chains. We claim that $y$ and $z$ lie in the same non-trivial connected component of $C \setminus (p_1 \cup p_2)$.

Assume to the contrary that $y$ or $z$ is an internal vertex of $p_1$ or $p_2$, say, that $y$ is an internal vertex of $p_1$. If $z$ is also a vertex of $p_1$, three calls cross the 2-cut consisting of an edge between $y$ and $z$ on $p_1$ and an arbitrary edge of $p_2$, contradicting (ii). If $z$ is not a vertex of $p_1$, the call $(y, z)$ is not parallel to the call routed along $p_1$, contradicting (i). Thus, $y$ and $z$ cannot be internal vertices of $p_1$ or $p_2$.

It remains to show that $y$ and $z$ cannot lie in different non-trivial connected components of $C \setminus (p_1 \cup p_2)$. If $y$ and $z$ were to lie in two different chains of $C \setminus (p_1 \cup p_2)$, a 2-cut consisting of arbitrary edges from $p_1$ and $p_2$ would be crossed by three calls, contradicting (ii).

Having excluded all other possibilities, we can conclude that $y$ and $z$ lie in the same chain of $C \setminus (p_1 \cup p_2)$, as claimed. This implies that $(y, z)$ can be routed in a unique way such that its path is edge disjoint from $p_1$ and $p_2$.

For each remaining call in $S$, we can apply the same reasoning as above: We show that both of its endpoints must be contained in one of the chains
remaining after all assigned paths are removed from $C$, and we assign the unique path in this chain to it. This completes the proof of the converse.

With the result of Lemma 4.2.1 at our disposal, we are ready to obtain the FPT algorithm for off-line MEDP-$k$ in trees of rings. If $k = 0$, the algorithm checks if the conditions of Lemma 4.2.1 hold for each ring of the tree of rings. If this is the case, an edge disjoint routing can be computed efficiently (the proof of Lemma 4.2.1 is constructive) and the algorithm outputs the feasible routing. Otherwise, the algorithm declares there is no edge disjoint routing for all calls in $S$.

Now assume that $k > 0$. If the condition of Lemma 4.2.1 holds for the sub-calls in each ring of the tree of rings, an edge disjoint routing for all calls is obtained. Otherwise, there are either two (sub-)calls in a ring that are not parallel, so that at least one of the two must be rejected, or there are three (sub-)calls crossing a 2-cut in some ring, so that at least one of the three must be rejected. In the former case, we get a set of two rejection candidates, in the latter case, we have three rejection candidates. For each call $r$ in the set of rejection candidates, we check recursively whether there exists a solution rejecting at most $k - 1$ calls from $S \setminus \{r\}$. The degree of any node in the search tree is at most 3. Since the depth of the search tree is bounded by $k$, the size of the search tree is $O(3^k)$. As the conditions of Lemma 4.2.1 can be checked easily in polynomial time at each node of the search tree, we obtain an FPT algorithm with running time $O(3^k \text{poly}(|S|))$.

**Theorem 4.2.2** There is an FPT algorithm for off-line MEDP-$k$ in undirected trees of rings.

The above discussion shows also that off-line MEDP-$k$ in undirected trees of rings can be seen as an instance of the problem HittingSet-$k$\textsuperscript{1} in which each set in the family has cardinality at most 3: The ground set $U$ consists of the calls $S$, and the family $\mathcal{S}$ of subsets of $U$ consists of all sets of two calls whose sub-calls in some ring are not parallel and all sets of three calls whose sub-calls cross a 2-cut in some ring.

An efficient FPT algorithm for HittingSet-$k$ with sets of size at most 3, called 3-HittingSet-$k$, has been shown by Niedermeier and Rossmanith in [NR03]. The algorithm combines the techniques of bounded search trees and reduction to problem kernel (see [DF99]) with the technique from [NR00]

---

\[\text{1The HittingSet-$k$ problem is defined as follows. Given a ground set } U, \text{ a family } \mathcal{S} \text{ of subsets of } U \text{ and a positive integer } k \text{ does there exist a subset } U' \subseteq U \text{ such that } |U'| \leq k \text{ and } U' \text{ intersects every set in } \mathcal{S}?\]
4.2. Off-line MEDP-\(k\) on trees of rings

Figure 4.2: (a) a directed chain, (b) a bidirected chain, (c) a directed chord crossing a 2-cut (consisting of the arcs shown dashed), (d) a pair of crossing chords, (e) a minimal set of routes for a pair of chords and (f) a non-minimal set of routes for the same pair of chords. For bidirected rings we show the bidirected arcs between two vertices as an edge that is directed both ways.

in order to obtain running time \(O(2.270^k + n)\), where \(n\) is the size of the input. Thus, by transforming a given instance of off-line MEDP-\(k\) into an instance of 3-HITTINGSET-\(k\) as outlined above, we obtain an FPT algorithm for off-line MEDP-\(k\) in undirected trees of rings that runs in time \(O(2.270^k + \text{poly}(|I|))\).

4.2.2 The bidirected case

We turn to bidirected trees of rings. Recall that each accepted call \((u, v)\) must now be assigned a directed path from \(u\) to \(v\). As in the undirected case, a set of calls is feasible if and only if all sub-calls are feasible in the individual rings. Thus, we consider an individual bidirected ring first.

Before proceeding further, let us adapt the terminology to the bidirected case. A directed chain is a directed graph obtained from an undirected chain...
by replacing every edge \( \{u, v\} \) in it by at least one of the two arcs \((u, v)\) or \((v, u)\). Note that every (sub-) call in a directed chain either has a unique route or has no route at all. A bidirected chain is a directed chain such that if an arc is in the graph then the oppositely directed arc is also in it. In a bidirected ring, we shall visualize a (sub-) call \((u, v)\) by drawing a directed chord from \(u\) to \(v\). A 2-cut in a bidirected ring is a set of 4 arcs such that if an arc is in the 2-cut then the oppositely directed arc is also in the 2-cut. Removing the arcs in a 2-cut from a bidirected ring leaves behind two components of the graph, each of which are bidirected chains (one of them could be a lone vertex). With respect to a planar embedding we can visualize these bidirected chains as lying on either side of the 2-cut and we say a vertex lies on a particular side of the cut. A call (or its corresponding chord) is said to cross a 2-cut if its end vertices lie on either side of the cut. Otherwise, it lies on one of the two possible sides of the 2-cut. In a bidirected ring, every call has two possible routes, which are directed paths from the start vertex to the end vertex. If a call crosses a 2-cut then all its routes must use exactly one of the arcs of the 2-cut. The routes for two calls are said to be edge disjoint if they do not share an arc. A set of routes \(R\) for a given set of calls is said to be an edge disjoint routing if routes in \(R\) are pairwise edge disjoint. A set of calls can be edge disjointly routed if there exists an edge disjoint routing for it. A set of routes \(R\) for a given set of calls is said to be minimal if there exists no other set of routes \(R'\) for the calls such that \(R'\) uses only a proper subset of arcs in \(R\). A pair of calls (or their chords) cross if their chords intersect at exactly one point but which is not one of their
end points. Figure 4.2 clarifies these notions.

The following simple lemmas capture some properties which will be essential in deriving the FPT algorithm for bidirected trees of rings.

**Lemma 4.2.3** Let $S$ be a set of calls in a bidirected ring $C$ such that every 2-cut in it is crossed by at most 2 chords corresponding to the calls in $S$. An edge disjoint routing of $S$ in $C$ exists and can be found in linear time.

**Proof:** If $|S| \leq 2$ the lemma is trivial. Assume $|S| \geq 3$. If a pair of chords cross then we can find a 2-cut crossed by these two chords and at least one other chord in $S$. This is more easily explained in figures (see Figure 4.3) than in words. If $|S| \geq 3$ then no two chords can cross pairwise. Now, on a clockwise traversal of the ring the end vertices of any chord are encountered consecutively. The edge disjoint routing of the chords is also found after the traversal. If the end vertices of call $r = (u, v)$ are encountered with $v$ following $u$, then its route is the set of arcs pointing in the direction of the traversal and encountered in the traversal between $u$ and $v$. Otherwise, it is the set of arcs encountered in the traversal between $u$ and $v$ and pointing in the reverse direction to the traversal. The fact that the calls are pairwise non-crossing guarantees that this routing is indeed edge disjoint. The time taken to find the edge disjoint routing is linear because (i) the traversal of the ring (and identification of the end vertices of calls during the traversal) takes time linear in the number of vertices of the ring and the number of calls and (ii) the routes for the calls are found by examining the traversal a second time.

**Lemma 4.2.4** Let a set of 3 chords in a bidirected ring $C$ cross a 2-cut. The maximum number of minimal edge disjoint routings of the 3 chords in $C$ is 2.

**Proof:** Let the 3 chords be $s_1, s_2, s_3$ and the 2-cut that they cross be $E'$. By the pigeon hole principle, at least two of the chords, without loss of generality say $s_1, s_2$, cross $E'$ such that both their end vertices lie on one side of $E'$ and both their start vertices lie on the other side of $E'$. If the start and end vertices of $s_3$ also lie on the same sides of $E'$ as those, respectively, of $s_1$ or $s_2$ there can be no edge disjoint routing for all of them. This is easy to see; there are only two arcs in $E'$ that point in the same “direction” as the 3 chords. The routes of the 3 chords must use at least one of these 2 arcs (Notice that when a chord crosses a 2-cut all its routes must use one of the arcs of the 2-cut). Thus, if all 3 chords must be edge disjointly routed,
Chapter 4. Call Control: FPT Results

the start and end vertices of $s_3$ must lie in those sides of $E'$ where the end and start vertices, respectively, of $s_1, s_2$ lie. Now, a case analysis gives the bound of 2 on the number of minimal edge disjoint routings of $s_1, s_2, s_3$. The Figure 4.4 shows two minimal edge disjoint routings for the cases discussed below.

Case 1: $s_1$ and $s_2$ cross
There are two minimal edge disjoint routings for $s_1$ and $s_2$. For each of these 2 routings there exists an arc such that it and its oppositely directed arc are used in the routing. This leaves at most one route for $s_3$ for each of the 2 edge disjoint routings of $s_1$ and $s_2$.

Case 2: $s_1$ and $s_2$ do not cross
There is exactly one minimal edge disjoint routing for $s_1$ and $s_2$. In a bidirected ring there are 2 routes for any chord. Thus, there are at most 2 minimal edge disjoint routings for all the 3 chords, $s_1, s_2, s_3$.

Figure 4.4: Two possible minimal edge disjoint routings for 3 chords. a(i) and a(ii) when the 2 chords shown in solid do not cross. b(i) and b(ii) when the 2 chords shown in solid cross.
Corollary 4.2.5 For a set of 4 chords in a bidirected ring $C$ that cross a 2-cut there are at most 2 minimal edge disjoint routings.

Proof: Fix 3 of these 4 chords. By Lemma 4.2.4 there are at most 2 minimal edge disjoint routings for these three chords. For each of these routings there is at most one way to edge disjointly route the fourth chord. Note that, given an edge disjoint routing of the 3 chords that is not minimal, routing the fourth chord edge disjointly does not yield a minimal edge disjoint routing for the four chords.

In a directed chain, two chords that have (unique) routes cannot be edge disjointly routed if their routes share an arc.

Lemma 4.2.6 Given a set of calls $S$ on a bidirected ring $C$ such that 4 calls $r_1, r_2, r_3, r_4 \in S$ cross a 2-cut, either (i) $S$ can be edge disjointly routed, or (ii) we can find a set $S_{\text{ref}} \subseteq S$ such that $|S_{\text{ref}}| \leq 5$ and $S_{\text{ref}}$ cannot be edge disjointly routed in $C$.

Proof: Assume that $S$ cannot be edge disjointly routed. We will then construct $S_{\text{ref}}$ as in condition (ii) of the statement of the lemma. Since
Chapter 4. Call Control: FPT Results

$r_1, r_2, r_3, r_4$ cross a 2-cut, say $E'$, there are at most two minimal edge disjoint routings for them in $C$ by Corollary 4.2.5.

Case 1: $r_1, r_2, r_3, r_4$ have no edge disjoint routings in $C$.
Set $S_{rej} = \{r_1, r_2, r_3, r_4\}$ and we are done.

Case 2: $r_1, r_2, r_3, r_4$ have only one minimal edge disjoint routing $R$ in $C$.
Fix this routing $R$ and remove from $C$ all arcs in $R$. The resulting graph, call it $L (= C \setminus R)$, is a collection of directed chains. Since $S$ cannot be edge disjointly routed in $C$, $S \setminus \{r_1, r_2, r_3, r_4\}$ cannot be edge disjointly routed in $L$. If a chord $r_k \in S \setminus \{r_1, r_2, r_3, r_4\}$ does not have a route in $L$ then setting $S_{rej} = \{r_1, r_2, r_3, r_4, r_k\}$ will satisfy the lemma. Otherwise, we may assume, by the observation before the statement of the lemma, that there are two chords $r_j, r_k \in S \setminus \{r_1, r_2, r_3, r_4\}$ whose unique routes in $L$ intersect at an arc. Since, $r_j, r_k$ both have routes in $L$ they must lie on one side of the 2-cut $E'$. Any edge disjoint routing of $r_j$ and $r_k$ will use two of the edges in $E'$. Thus, we can take $S_{rej} = \{r_1, r_2, r_3, r_j, r_k\}$. Any edge disjoint routing of chords in $S_{rej}$ will need 5 arcs in $E'$, which is impossible.

Case 3: $r_1, r_2, r_3, r_4$ have two minimal edge disjoint routings in $C$.
At least two of them with their start (end) vertices on the same side of $E'$ must cross each other, say $r_1, r_2$. Let $\{a, b, c, d\}$ be the set of end vertices of $r_1$ and $r_2$ as shown in Figure 4.5. In the figure we show only 4 vertices other than $a, b, c$ and $d$, but there could be many others.

Case 3a: There is a chord $r_j \in S \setminus \{r_1, r_2\}$ one of whose end vertices lies between $a$ and $b$ while going clockwise around the ring.
If $r_1$ and $r_2$ are routed as shown in Figure 4.5(i), $r_j$ has no route among the remaining arcs of $C$. We fix the routing of $r_1$ and $r_2$ to be as shown in Figure 4.5(ii). Now, the routes of all chords in $S \setminus \{r_1, r_2\}$ are unique or do not exist. If no route exists for $r_j$ then we set $S_{rej} = \{r_1, r_2, r_j\}$. Otherwise, we fix the unique route for $r_j$ among the remaining set of arcs. By assumption $S$ cannot be edge disjointly routed in $C$. Therefore, we can either find a chord $r_k$ which has no route after the routes for $r_1, r_2, r_j$ have been fixed or we find two chords $r_k, r_l$ whose unique routes intersect. We set $S_{rej} = \{r_1, r_2, r_j, r_l\}$ in the former case and $S_{rej} = \{r_1, r_2, r_j, r_k, r_l\}$ in the latter case. It can now be verified that $S_{rej}$ satisfies the conditions of the lemma in all these cases.

Case 3b: There is a chord $r_j \in S \setminus \{r_1, r_2\}$ one of whose end vertices lies between $c$ and $d$ while going anti-clockwise around the ring.
Arguments similar to Case 3a go through by observing that $r_j$ has no route when $r_1$ and $r_2$ are routed as shown in Figure 4.5(ii).

Case 3c: There is a chord $r_j \in S \setminus \{r_1, r_2\}$ one of whose end vertices lies between $a$ and $c$ while going anti-clockwise around the ring.
Let this end vertex be $f$. Notice that all arcs of the clockwise path from $c$ to $a$ are used up in both the possible routings of $r_1$ and $r_2$ (See Figure 4.5).
There is at most one way of routing \( r_3 \) using the remaining set of arcs. If no such route exists, then we set \( S_{rej} = \{ r_1, r_2, r_j \} \). If a route for \( r_j \) exists, fix this route. Let the vertex adjacent to \( f \) and contained in the route of \( r_j \) be \( g \). If \( r_1, r_2 \) and \( r_j \) are to be edge disjointly routed then both the arcs \((f,g)\) and \((g,f)\) would be used. Consider the graph \( L \) obtained from \( C \) by removing the arcs \((f,g)\) and \((g,f)\) and the (remaining) arcs in the route fixed for \( r_j \). The routes for chords in \( S \setminus \{ r_1, r_2, r_j \} \) are unique or do not exist in \( L \). If there is a chord \( r_k \) which has no route in \( L \) then we can set \( S_{rej} = \{ r_1, r_2, r_j, r_k \} \). If the unique routes in \( L \) for two of the chords, say \( r_k, r_l \), intersect then we can set \( S_{rej} = \{ r_1, r_2, r_j, r_k, r_l \} \). Otherwise, \( S \setminus \{ r_1, r_2, r_j \} \) has an edge disjoint routing in \( L \). Now, we can identify a chord \( r_{j1} \in S \setminus \{ r_1, r_2, r_j \} \) (respectively, \( r_{j2} \in S \setminus \{ r_1, r_2, r_j \} \)) such that its unique route in \( L \) intersects either the route of \( r_1 \) or the route of \( r_2 \) or both, when \( r_1, r_2 \) are routed as shown in Figure 4.5(i) (respectively, Figure 4.5(ii)). This statement follows because we assumed that \( S \) has no edge disjoint routing in \( C \). In this case, we set \( S_{rej} = \{ r_1, r_2, r_j, r_{j1}, r_{j2} \} \). It is possible that \( r_{j1} = r_{j2} \). One can verify that \( S_{rej} \), in each of these cases, satisfies the conditions of the lemma.

Case 3d: There is a chord \( r_j \in S \setminus \{ r_1, r_2 \} \) one of whose end vertices lies between \( b \) and \( d \) while going clockwise around the ring.

Arguments similar to those in Case 3c go through by observing that all the arcs in the anti-clockwise path from \( d \) to \( b \) are used by both the possible routings of \( r_1 \) and \( r_2 \).

If none of the above case distinctions hold for chords in \( S \setminus \{ r_1, r_2 \} \) then the set of start and end vertices of the chords is a subset of \{a, b, c, d\}. For the arguments below, keep in mind that we are still in Case 3. If \( r_3 \) and \( r_4 \) cross then without loss of generality \( r_3 = (a, d) \) and \( r_4 = (b, c) \). Any edge disjoint routing of \( r_1, r_2, r_3, r_4 \) uses all the arcs of \( C \). Thus, \( S_{rej} = \{ r_1, r_2, r_3, r_4, r_j \} \), where \( r_j \) is any chord in \( S \setminus \{ r_1, r_2, r_3, r_4 \} \), will satisfy the conditions of the lemma. If \( r_3 \) and \( r_4 \) do not cross then without loss of generality \( r_3 = (a, c) \) and \( r_4 = (b, d) \). We can now argue similarly as in Case 3c by observing that both the arcs \((a,g)\) and \((g,a)\) will be used by any edge disjoint routing of \( r_1, r_2, r_3 \), where \( g \) is the vertex adjacent to \( a \) and lies between \( a \) and \( c \) while going anti-clockwise.

The case analysis, under Case 3, considers all the possibilities for the chords in \( S \setminus \{ r_1, r_2 \} \). For each case, we have shown that when \( S \) cannot be edge disjointly routed we can identify a set \( S_{rej} \) satisfying the conditions of the lemma.

Lemma 4.2.7 Given a set of calls \( S \) on a bidirected ring \( C \) such that 3 calls \( r_1, r_2, r_3 \in S \) cross a 2-cut,
either (i) $S$ can be edge disjointly routed, or (ii) we can find a set $S_{rej} \subseteq S$ such that $|S_{rej}| \leq 5$ and $S_{rej}$ cannot be edge disjointly routed in $C$.

**Proof:** We will again prove (ii) assuming (i) does not hold. Let $r_1, r_2, r_3$ cross the 2-cut $E'$.

Case 1: There is a chord $r_4 \in S \setminus \{r_1, r_2, r_3\}$ that crosses $E'$.

Proof of Lemma 4.2.6 shows (ii).

Case 2: No chord in $S \setminus \{r_1, r_2, r_3\}$ crosses $E'$.

By Lemma 4.2.4 there are at most two minimal edge disjoint routings of $r_1, r_2, r_3$.

Case 2a: $r_1, r_2, r_3$ have no edge disjoint routings in $C$.

Set $S_{rej} = \{r_1, r_2, r_3\}$.

Case 2b: $r_1, r_2, r_3$ have only one minimal edge disjoint routing $R$ in $C$.

As in Case 2 of proof of Lemma 4.2.6, we can either find a chord $r_j$ which has no route in $C \setminus R$ or we can find a pair of chords $r_j, r_k$ which have routes in $C \setminus R$ but cannot be edge disjointly routed. We let $S_{rej} = \{r_1, r_2, r_3, r_j\}$ or $S_{rej} = \{r_1, r_2, r_3, r_j, r_k\}$ according to the two possibilities.

Case 2c: $r_1, r_2, r_3$ have two minimal edge disjoint routings $R_1, R_2$ in $C$.

All chords other than $r_1, r_2, r_3$ lie on one side of $E'$. Suppose for one of the routings, say $R_1$, there are two distinct chords $r_j, r_k$ which have (unique) routes in $C \setminus R_1$ but their routes intersect at an arc. $S_{rej} = \{r_1, r_2, r_3, r_j, r_k\}$ follows from the same arguments as in Case 2 of Lemma 4.2.6. Otherwise, as $S$ is not edge disjointly routable there must exist chords $r_j, r_k$ (not necessarily distinct) that have no routes in $C \setminus R_1, C \setminus R_2$, respectively. Now, $S_{rej} = \{r_1, r_2, r_3, r_j, r_k\}$.

In each of the cases $S_{rej}$ satisfies condition (ii) of the statement of the lemma.

Lemma 4.2.7 is the core ingredient of our FPT algorithm for MEDP-$k$ on tree of rings $T$ given a set of calls $S$. Note that checking whether a set of (sub-)calls in a bidirected ring can be edge disjointly routed can be done efficiently. First, test if there is a 2-cut crossed by more than 2 calls. If the test is negative, Lemma 4.2.3 even gives an edge disjoint routing for the calls. If the test is positive, we have a 2-cut crossed by (at least) 3 chords. By routing them along the (at most) two different minimal edge disjoint routings, we can test whether all the calls are edge disjointly routable. This check also gives an edge disjoint routing for the calls, if it exists.

If $k = 0$ in MEDP-$k$ on tree of rings $T$ with input set of calls $S$, then for each bidirected ring $C$ in $T$ we test whether the (sub-)calls can be edge
disjointly routed. If the test fails for one of the rings, we answer no and stop. Otherwise, we have an edge disjoint routing of all calls in $S$. If $k \geq 1$ and if the above test fails for one of the rings, by Lemma 4.2.7 we have a set $S_{rej}$ with at most 5 calls such that at least one call in it must be rejected to find an edge disjoint routing for the remaining calls. Recursively, we query MEDP-$(k - 1)$ on $T$ with calls $S \setminus \{r\}, r \in S_{rej}$. An FPT algorithm can be devised from the search tree so built. The search tree has a maximum degree of 5 at each node. If at any node, with level at most $k$, we have a feasible solution then we have the set $S'$ for the original MEDP-$k$ problem given by the calls rejected along the path from that node to the root of the search tree. Otherwise, we can declare that no set $S'$ exists.

**Theorem 4.2.8** There is an FPT algorithm for off-line MEDP-$k$ in bidirected trees of rings that runs $O(5^k \text{poly}(|I|))$ time, where $|I|$ is the size of the input to the MEDP-$k$ problem.

In [NR03], an FPT algorithm is presented for the $d$-HITTINGSET-$k$ problem, the HITTINGSET-$k$ problem in which the cardinality of the sets in the family is at most $d$, which runs in time $O(c^k + n)$, where $c = d - 1 + O(d^{-1})$ and $n$ is the size of the input. For $d = 5$, the constant $c = 4.23$ [NR03]. Similar to the undirected case, we can view an instance of the MEDP-$k$ problem on bidirected trees of rings as an instance of HITTINGSET-$k$ where the cardinality of the sets in the family is at most 5: The ground set $U$ of the instance would be the set of calls $S$, the family $S$ of subsets of $U$ would be all subsets of $S$ of cardinality at most 5 such that their sub-calls in some ring of the tree of rings cannot be edge disjointly routed. We consider only subsets with cardinality up to 5 since we have shown above that for any set of (sub-)calls that cannot be edge disjointly routed we can identify a subset with (at most) 5 of them that cannot be so routed. Thus, we can obtain an FPT algorithm for MEDP-$k$ in bidirected trees of rings that runs in time $O(4.23^k + \text{poly}(|I|))$, where $|I|$ is the size of the input to the MEDP-$k$ problem.
Chapter 5  Routing and Call Control

This chapter considers the off-line, uniform RCC problem on rings. For the objective of maximizing the number of accepted calls, a polynomial time algorithm with an additive guarantee of 3 is devised. For the objective of maximizing the profits of accepted calls, a 2-approximation is given.

The problems studied in this chapter have their origins in the analysis carried out in Chapter 4. There, in each individual ring we required to answer a “yes-or-no” question: can the set of (sub-)calls in a ring be routed such that the routes are pairwise edge disjoint? For the purposes of that chapter, we were not interested in the following question. Can we find a maximum cardinality subset of (sub-)calls in a ring that can be routed such that the routes are pairwise edge disjoint, if all the (sub-)calls in that ring cannot be so routed. We tackle this question here but in a more generalized setting. The underlying topology in this generalized setting is a ring. The edges have arbitrary capacities instead of unit capacities. Individual calls still have bandwidth requirements of 1 as before. When we want to maximize the number of calls that can be accepted and routed, we provide an algorithm that accepts at most 3 fewer calls as compared to an optimal algorithm. It also gives the routes for the accepted calls. The details can be found in Section 5.1. If calls have arbitrary profits associated with them and the objective is to maximize the profits of accepted calls, we give a 2-approximation. The details are presented in Section 5.2.

In terms of the formal definitions of Chapter 2, the problems addressed are the off-line RCC problems on rings when (i) all calls have bandwidth requirements and profits of 1 and (ii) all calls have bandwidth requirements of 1 and arbitrary profits.

5.1 The unit profits case

Let us begin by formally specifying the off-line RCC problem on rings we consider in this section. An input instance consists of an edge capacitated
5.1. The unit profits case

Figure 5.1: Ring with 8 edges. Call $i$ and its two indicator variables

ring $C = (V, E, c)$, a set of calls $S = \{\{u_i, v_i\} : u_i, v_i \in V, i \in \{1, 2, ..., m\}\}$ all with unit bandwidth requirements (omitting the other common parameters of the calls) and a profit function on calls which assigns a profit of 1 for all calls. A feasible solution is a feasible subset $S' \subseteq S$ in $C$ together with a specification of the routes of calls in $S'$. The objective is a feasible solution with maximum possible profit or equivalently, a feasible solution with the maximum cardinality.

Our approach to solving the off-line RCC problem is to formulate it as an integer linear program and to round an optimal fractional solution of the relaxed program to a feasible solution. We shall show that the feasible solution so generated is very close to an optimal solution.

5.1.1 An integer linear program

The formulation of an integer linear program (ILP) for the above off-line RCC problem is a natural one. Let the vertex set of $C$ be $V = \{1, 2, ..., n\}$ and $E = \{\{j, j + 1 \mod n\} : j = 1, 2, ..., n\}$ (0 is identified with $n$). We shall refer to the edge $\{j, j + 1 \mod n\}$ as the edge $j, j \in \{1, 2, ..., n\}$. We shall refer to the call $\{u_i, v_i\}, i \in \{1, 2, ..., m\}$ by its index $i$. Further, we consider a fixed embedding of the ring on a plane and assign a clockwise direction. For each call $i$, introduce two indicator variables $x_{i1}$ and $x_{i2}$ corresponding to the two possible routes. The first of them corresponds to the path containing edge 1 and the other to the path that does not. See Figure 5.1 for an illustration. For edge $j$, let $S_j = \{x_{ik} : \text{route } x_{ik} \text{ contains edge } j, i = 1, 2, ..., m, k \in \{1, 2\}\}$. Now, the ILP looks as follows:
Chapter 5. Routing and Call Control

max \sum_{i=1}^{m}(x_{i1} + x_{i2})

subject to
\sum_{x_{ik} \in S_j} x_{ik} \leq c(j), j = 1, 2, ..., n
x_{i1} + x_{i2} \leq 1, i = 1, 2, ..., m
x_{ik} \in \{0, 1\}, i = 1, 2, ..., m, k = 1, 2

Relaxing the above ILP changes the last of the constraints by admitting all fractional values between 0 and 1. The relaxed ILP can be solved in time polynomial in \(n, m\) and \(\log c(.)\). Denote the fractional optimal solution vector as \(x^* = (x_{11}^*, x_{12}^*, x_{21}^*, x_{22}^*, ..., x_{i1}^*, x_{i2}^*, ..., x_{m1}^*, x_{m2}^*)\) and the objective value by \(OPT^*\). It will be helpful to think of the vector \(x\) as a function on the set of routes of the calls into the real interval \([0, 1]\). Hence, we shall refer to \(x\) as a route function and as a \(\{0, 1\}\)-route function, if the components of \(x\) are either 0 or 1. The import is that \(x_{i1}, x_{i2}\) are the values of \(x(i_1), x(i_2)\) respectively. Note that for a route function, \(x_{i1} + x_{i2} \leq 1\), for all calls \(i\). Moreover, we shall talk about the route function independent of the edge capacities of the ring.

5.1.2 Rounding scheme

Before describing the rounding scheme it is useful to distinguish a relation between pairs of calls. We have already encountered this relation in the previous chapter. We define it here again, for convenience. Two calls \(i = \{u_i, v_i\}\) and \(j = \{u_j, v_j\}\) are said to be parallel if either their end points appear as \(u_i, u_j, v_j, v_i\) while traversing the ring in clockwise fashion or they share a common end point. Observe that since the pair of vertices in a call are unordered the order in which vertices of each call, namely \(u_i, v_i\), themselves appear is immaterial. If two calls are not parallel then they are said to be crossing. Equivalently, a clockwise traversal encounters these end points in

Figure 5.2: Parallel and crossing calls
the order \(u_i, u_j, v_i, v_j\). A simple observation is that when two calls are parallel one of the routes of the first call is totally contained in a route of the second and vice-versa. Parallel and crossing calls are illustrated in Figure 5.2.

The rounding scheme starts off by doing a preliminary set of transformations on parallel and crossing calls so that the components of the fractional optimal vector \(x^*\) are in a particular “canonical” form; namely, for a pair of parallel calls at least one of the \(x^*\) values is zero and for a pair of crossing calls either at least one of the \(x^*\) values is zero or the sum of the \(x^*\) values of one of the calls is 1. It should be remarked that while we change the values of the components of \(x^*\) we do not affect either the feasibility of the resultant vector or the objective value. We proceed to describe them below.

Transformations on parallel calls: Let \(i\) and \(j\) be two parallel calls with the path \(x_{i1}(x_{j2})\) contained in the path \(x_{j1}(x_{i2})\), say, and \(x_{i1}^*, x_{i2}^*, x_{j1}^*, x_{j2}^* > 0\). The goal behind this transformation is to set at least one of the fractional values \(x_{i1}^*, x_{i2}^*, x_{j1}^*, x_{j2}^*\) to zero. Let \(y = \min\{x_{i1}^*, x_{i2}^*, x_{j1}^*, x_{j2}^*\}\). We set

\[
\begin{align*}
x_{i1}^* &\leftarrow x_{i1}^* + y; \quad x_{i2}^* &\leftarrow x_{i2}^* - y; \quad x_{j1}^* &\leftarrow x_{j1}^* - y; \quad x_{j2}^* &\leftarrow x_{j2}^* + y.
\end{align*}
\]

Transformations on crossing calls: Consider two crossing calls \(i\) and \(j\) with \(x_{i1}^*, x_{i2}^*, x_{j1}^*, x_{j2}^* > 0\) and neither of \(x_{i1}^* + x_{i2}^*, x_{j1}^* + x_{j2}^*\) are unity. The aim of this transformation is to either set at least one of the variables to zero or make one of the sums \(x_{i1}^* + x_{i2}^*, x_{j1}^* + x_{j2}^*\) equal unity. This is achieved in a slightly more involved transformation shown below:

Set \(c_i = 1 - (x_{i1}^* + x_{i2}^*)\); \(c_j = 1 - (x_{j1}^* + x_{j2}^*)\)

and \(y = \min\{\frac{c_i}{2}, \frac{c_j}{2}, x_{i1}^*, x_{i2}^*, x_{j1}^*, x_{j2}^*\}\).

Case 1: if \(y = \frac{c_i}{2}\),
\[
\begin{align*}
x_{i1}^* &\leftarrow x_{i1}^* + y; \quad x_{i2}^* &\leftarrow x_{i2}^* - y; \quad x_{j1}^* &\leftarrow x_{j1}^* - y; \quad x_{j2}^* &\leftarrow x_{j2}^* - y
\end{align*}
\]
\(\implies x_{i1}^* + x_{i2}^* = 1\).

Case 2: if \(y = \frac{c_j}{2}\),
\[
\begin{align*}
x_{i1}^* &\leftarrow x_{i1}^* - y; \quad x_{i2}^* &\leftarrow x_{i2}^* - y; \quad x_{j1}^* &\leftarrow x_{j1}^* + y; \quad x_{j2}^* &\leftarrow x_{j2}^* + y
\end{align*}
\]
\(\implies x_{j1}^* + x_{j2}^* = 1\).

Case 3: if \(y = x_{i1}^*\) or \(x_{i2}^*\),
\[
\begin{align*}
x_{i1}^* &\leftarrow x_{i1}^* - y; \quad x_{i2}^* &\leftarrow x_{i2}^* - y; \quad x_{j1}^* &\leftarrow x_{j1}^* + y; \quad x_{j2}^* &\leftarrow x_{j2}^* + y
\end{align*}
\]
\(\implies x_{i1}^* = 0\) or \(x_{i2}^* = 0\).
Case 4: if \( y = x^*_1 \) or \( x^*_2 \),
\[
x^*_1 \leftarrow x^*_1 + y; x^*_2 \leftarrow x^*_2 + y; x^*_1 \leftarrow x^*_1 - y; x^*_2 \leftarrow x^*_2 - y
\]
\[\implies x^*_{j1} = 0 \text{ or } x^*_{j2} = 0.\]

If \( x^* \) is a basic feasible solution to the linear program it can be proved that the \( x^* \) values of pairs of crossing calls are already in the canonical form that we require. Indeed, consider a pair of crossing calls \( i \) and \( j \) that have \( x^* \) values not in the canonical form. Compute two new feasible solutions from \( x^* \) as follows (we use \( y(\neq 0) \) computed in the description above): (i) add (subtract) \( y/2 \) to (from) both components of \( x^* \) values of call \( i \) (\( j \)), the other components of \( x^* \) remain the same and (ii) subtract (add) \( y/2 \) from (to) both components of \( x^* \) values of call \( i \) (\( j \)), the other components of \( x^* \) remain the same. Clearly, the original \( x^* \) can be expressed as a convex combination of these two new vectors. Thus, if the \( x^* \) values of \( i \) and \( j \) were not in canonical form it is not basic. Conversely, if \( x^* \) were basic then for the crossing calls \( i \) and \( j \), the corresponding values are in the canonical form. For parallel calls, however, a basic solution need not be in the canonical form. Consider a ring on 4 vertices with edge capacities 1 and two parallel calls \( i \) and \( j \) in this ring. The feasible solution \( x_{i1} = x_{i2} = x_{j1} = x_{j2} = 0.5 \) is basic but is not in the canonical form we require for parallel calls.

In any case, the above described transformations convert any feasible solution of the LP into the canonical form we want. These transformations performed on every pair of calls partition the call set into four categories according to the values of their corresponding indicator variables in the optimal solution vector \( x^* \):

A) Calls for which both the corresponding indicator variables are set to zero. Let the set be denoted by \( S(a) \) and the sum of their \( x^* \) values by \( x^*(S(a)) = 0. \)

B) Calls for which exactly one of the corresponding indicator variables is non-zero. Let the set be denoted by \( S(b) \) and the sum of their \( x^* \) values by \( x^*(S(b)) \).

C) Calls which are pairwise crossing but the sum of their (non-zero) indicator variables equals unity. Let the set be denoted by \( S(c) \) and the sum of their \( x^* \) values by \( x^*(S(c)) \).

D) At most one call for which the sum of the (non-zero) indicator variables is less than one. Let the call be \( D \) and the sum of its \( x^* \) values be \( x^*_D + x^*_D < 1 \) with \( 0.5 > x^*_D \leq x^*_D, \) say.

Clearly, the rounding scheme needs to handle the class B, C and D calls by setting the indicator variables to either 0 or 1. The different categories of calls require different rounding mechanisms. We shall describe these round-
ing schemes in a sequence of lemmas below. These lemmas prove that the solution obtained through the rounding maintains feasibility and remains “close” to an optimal solution at the same time. While the calls are partitioned into different classes based on the route function $x^*$ values, for ease of presentation a set of calls will be said to be of a particular class if their $x^*$ values satisfy the conditions for that class.

**Rounding of class B calls**

Since calls in class B have one of their two indicator variables set to zero, the route function $x$ can be restricted to be defined on the unique route for each call that received a non-zero value. Instead of calls, we need only concentrate on the unique path for each call in class B. Accordingly, we show the rounding on a set of paths.

**Lemma 5.1.1 (Rounding on the line)** Let $R$ be a set of paths on an edge capacitated line $L = (V,E,c)$. Let $x : R \rightarrow [0,1]$ be a function that assigns fractional values to the set of paths and $x(R) = \sum_{s \in R} x(s)$. Further, let $x(.)$ be such that the sum of $x$ values of paths through an edge $e$ on the line is at most $c(e)$. There exists a function $x' : R \rightarrow \{0,1\}$ such that $\forall e : \sum_{s \in R:e \text{ contains } e} x'(s) \leq c(e)$ and $x'(R) = \sum_{s \in R} x'(s) \geq \lfloor x(R) \rfloor$.

**Proof:** Let $R = \{s_1, s_2, ..., s_m\}$ be a set of $m$ paths on the line such that the right end point of $s_{i+1}$ lies to the right (possibly even coinciding) of the right end point of $s_i$, $i = 1, 2, ..., m - 1$. We prove the lemma by induction on the cardinality of $R$.

**Basis Step:** If $m = 1$, then define $x'$ as:

$$x'(s_1) = \begin{cases} 0, & \text{if } x(s_1) = 0 \\ 1, & \text{if } x(s_1) > 0 \end{cases}$$

**Induction Step:** Assume the lemma holds true for $R$ with cardinality less than $m(> 1)$. If $x(s_1) = 0$ or 1, we can invoke the induction hypothesis on $R \setminus \{s_1\}$ and prove the lemma for $R$. Assume $0 < x(s_1) < 1$. We define $x'(.)$ as follows. Set $x'(s_1) = 1$ and $\epsilon = 1 - x(s_1)$. If $x'(s_1)$ together with $x(s_2), x(s_3), ..., x(s_m)$ does not violate any of the capacities of edges in $s_1$, then this is similar to the case $x(s_1) = 1$. Otherwise, let $e$ be the leftmost edge on $s_1$ that was violated by rounding $x'(s_1)$ to 1. Let $s_{e_1}, s_{e_2}, ..., s_{e_k}$ be the set of paths (other than $s_1$) that pass through $e$ and appear in increasing order of their right end points. If $\sum_{j=1}^{l} x(s_{e_j}) \geq \epsilon$ then let $e_i$ be the least index such that $\sum_{j=1}^{i} x(s_{e_j}) \geq \epsilon$. Reassign $x(s_{e_j}) = 0$, $j = 1, 2, ..., l - 1$
Chapter 5. Routing and Call Control

and $x(s_{e_1}) = \sum_{j=1}^{k} \alpha_{old}(s_{e_j}) - \epsilon$ ($\alpha_{old}(\cdot)$ denotes the $\alpha(\cdot)$ values before the reassignment). Now, $x'(s_1)$ together with the new $x(s_{e_1}), x(s_{e_2}), \ldots, x(s_{e_k})$ add up to at most $c(e)$. If $\sum_{j=1}^{k} x(s_{e_j}) < \epsilon$, reassign $x(s_{e_j}) = 0, j = 1, 2, \ldots, k$. Edge $e$ is not violated anymore and edges to the right of $e$ and in $s_1$ can have an excess of at most $\epsilon - \sum_{j=1}^{k} \alpha_{old}(s_{e_j})$. We can repeat the above procedure for the next leftmost violated edge after $e$ and continue till no edge in $s_1$ is violated. Observe that the new $x$ values are such that $x(R \setminus \{s_1\}) \geq \alpha_{old}(R \setminus \{s_1\}) - \epsilon$. Further, the new $x$ values of paths (other than $s_1$) through an edge $e$ contained in path $s_1$ sum up to at most $c(e) - 1$. We can now apply the induction hypothesis on the set $R \setminus \{s_1\}$ with the new $x$ values to obtain a function $x' : R \setminus \{s_1\} \to \{0, 1\}$ that satisfies the conditions of the lemma. This $x'$ together with $x'(s_1) = 1$ is the required $x'$ for $R$, since at any edge $e$ the sum of the $x'$ values of paths through $e$ add up to at most $c(e)$ and,

\[
x'(R) = x'(s_1) + x'(R \setminus \{s_1\}) \\
\geq 1 + [\alpha_{old}(R \setminus \{s_1\}) - \epsilon] \\
= [\alpha_{old}(R \setminus \{s_1\}) - \epsilon - 1] \\
= [\alpha_{old}(R)], \\
x_{old}(s_1) = 1 - \epsilon
\]

Hence, the lemma.

An alternative proof of the lemma can be formulated using the theory of interval graphs. The above rounding lemma for the line serves as a starting step to round the values for paths on the ring. The next lemma captures this.

Lemma 5.1.2 (Rounding on the ring) Let $R$ be a set of paths on an edge capacitated ring $C = (V, E, c)$. Let $x : R \to [0, 1]$ be a function that assigns fractional values to the set of paths and $x(R) = \sum_{s \in R} x(s)$. Further, let $x(.)$ be such that the sum of $x$ values of paths through an edge $e$ on the ring is at most $c(e)$ and for some edge $e_{sat}$ the sum is exactly $c(e_{sat})$. There exists a function $x' : R \to \{0, 1\}$ such that $\forall e, \sum_{s \in R : e \text{ contains } s} x'(s) \leq c(e)$ and $x'(R) = \sum_{s \in R} x'(s) \geq \lfloor x(R) \rfloor - 1$.

Proof: Consider the edge $e_{sat}$ in the ring and the set of paths $R_{e_{sat}} \subseteq R$ that pass through it. If there were two paths $s_{e_1}, s_{e_2}$ through $e_{sat}$ such that the former is contained in the latter then consider the following reassignment of their $x$ values; $x(s_{e_1}) \leftarrow \min\{1, x(s_{e_1}) + x(s_{e_2})\}, x(s_{e_2}) \leftarrow x(s_{e_2}) + \alpha_{old}(s_{e_1}) - x(s_{e_1})$, where $\alpha_{old}(s_{e_1})$ is the value of $x(s_{e_1})$ before the reassignment. Obviously, with this reassignment the new $x$ values add up to their old $x$ values. It
is also easy to observe that this reassignment either makes the $x$ value of the shorter path 1 or the $x$ value of the longer path 0. Thus, without loss of generality we can assume that all paths through $e_{sat}$ which have $x$ values in $(0,1)$ are not fully contained in each other. Call these paths $R_{e_{sat}}^{(1)} = \{s_1, s_2, ..., s_k\}$ where the order in which they appear is according to the increasing order of their clockwise end points. Note that $x(R_{e_{sat}}^{(1)})$ is an integer since $x(e_{sat}) = c_{e_{sat}}$ is an integer. Let $e_j$ be the smallest index such that $\sum_{i=1}^{j} x(s_i) \geq j, j = 1, 2, ..., x(R_{e_{sat}}^{(1)}) - 1$. Define $x'(s_j) = 1, j = 1, 2, ..., x(R_{e_{sat}}^{(1)}) - 1$ and $x'(s_i) = 0$, for $s_i \in R_{e_{sat}}^{(1)} \setminus \{s_1, s_2, ..., s_{x(R_{e_{sat}}^{(1)}) - 1}\}$.

Also, set $x'(s) = x(s)$, for $s \in R_{e_{sat}} \setminus R_{e_{sat}}^{(1)}$. Recall that for the paths in $R_{e_{sat}} \setminus R_{e_{sat}}^{(1)}$ the $x$ values are either 0 or 1.

**Argument:** For any edge $e'$ in the ring, the sum of the $x'$ values of paths in $R_{e_{sat}}^{(1)}$ that pass through it is at most the rounded down sum of their $x$ values.

**Proof:** Consider an edge $e'$ in the ring.

Case 1: $s_1$ and $s_k$ pass through $e'$.

If all paths in $R_{e_{sat}}^{(1)}$ pass through $e'$ then the sum of their $x'$ values is $x(R_{e_{sat}}^{(1)}) - 1$ which is one less than the sum of their $x$ values, $x(R_{e_{sat}}^{(1)})$ as required. Otherwise, because paths in $R_{e_{sat}}^{(1)}$ are not contained in each other, it should be the case that paths $s_i, s_{i+1}, ..., s_{i+l},$ for some $i, l, 2 \leq i \leq i + l \leq k - 1$, do not pass through $e'$. The sum of the $x$ values of paths through $e'$ then would be $\sum_{j=1}^{i-1} x(s_j) + \sum_{j=i+l+1}^{k} x(s_j)$. The sum of the $x'$ values of paths through $e'$ would be $\sum_{j=1}^{i-1} x'(s_j) + \sum_{j=i+l+1}^{k} x'(s_j)$. By the definition of $x'$, $\sum_{j=1}^{i-1} x'(s_j) = \left[\sum_{j=1}^{i-1} x(s_j)\right]$. Also,

\[
\sum_{j=i+l+1}^{k} x'(s_j) = \sum_{j=1}^{k} x'(s_j) - \sum_{j=1}^{i+l} x'(s_j) = x(R_{e_{sat}}^{(1)}) - 1 - \left[\sum_{j=1}^{i+l} x(s_j)\right] < x(R_{e_{sat}}^{(1)}) - \sum_{j=1}^{i+l} x(s_j), \quad m < [m] + 1, \forall m \in \mathbb{R}
\]

But, since $x'$ values are either 0 or 1, $\sum_{j=i+l+1}^{k} x'(s_j)$ must be an integer. Therefore, $\sum_{j=i+l+1}^{k} x'(s_j) \leq \left[\sum_{j=i+l+1}^{k} x(s_j)\right]$. Thus, the sum of $x'$ values of paths through $e'$ would be at most $\left[\sum_{j=1}^{k} x(s_j) + \sum_{j=i+l+1}^{k} x(s_j)\right]$, which satisfies the conditions of the argument.
Case 2: exactly one of \( s_1 \) and \( s_k \) contains \( e' \).

Say, \( s_1 \) contains \( e' \), the other case being symmetric. Again by non-containment property, there is an \( i, 1 \leq i \leq k - 1 \), such that all paths with index from 1 through \( i \) pass through \( e' \). The sum of the \( x \) values of these paths is \( \sum_{j=1}^{i} x(s_j) \) which is at least \( \lceil \sum_{j=1}^{i} x(s_j) \rceil = \sum_{j=1}^{i} x'(s_j) \). This completes the proof of the argument.

Now, consider all paths that do not pass through \( e_{sat} \), they lie on the line obtained by removing the edge \( e_{sat} \) from the ring. Therefore, we can invoke Lemma 5.1.1, to obtain an \( x' \) function on them which satisfies the condition that the sum of \( x' \) values passing through any edge is at most the rounded up value of the sum of their \( x \) values. The \( x' \) values of paths in \( R_{esat} \setminus R^{(1)}_{esat} \) are the same as their \( x \) values. The facts in the two previous sentences, combined with the statement of the above argument implies that the \( x' \) values of paths in \( R \) that pass through any edge \( e \) of the ring sum up to at most the capacity of that edge, \( c(e) \). Further, we have

\[
x'(R) = x'(R_{esat}) + x'(R \setminus R_{esat})
\geq x(R_{esat} \setminus R^{(1)}_{esat}) + x(R_{esat}) - 1 + \lceil x(R \setminus R_{esat}) \rceil, \quad \text{by Lemma 5.1.1}
= x(R_{esat}) - 1 + \lceil x(R) - x(R_{esat}) \rceil
= \lceil x(R) \rceil - 1.
\]

Lemma 5.1.2 immediately suggests a rounding scheme for class B calls such that rounded values at any edge sum up to at most the rounded up value of the sum of their \( x' \) values and at the same time lose at most one from their cumulative sum. We note that if none of the \( x' \) values at an edge sum exactly to the rounded up value at an edge then we can increase at least one of the \( x' \) values to satisfy the condition or make all \( x' \) values equal 1. This is summarized in the corollary below.

**Corollary 5.1.3 (Rounding class B calls)** Consider a set of class B calls \( S_{(b)} \) with a corresponding route function \( x^* \). There exists a \( \{0, 1\} \)-route function \( x' \) such that

(i) at every edge the sum of the \( x' \) values through it is at most the rounded up value of the sum of their \( x^* \) values.

(ii) \( x'(S_{(b)}) = \sum_{i \in S_{(b)}} (x'_{1i} + x'_{2i}) \geq \lceil x^*(S_{(b)}) \rceil - 1 \).
5.1. The unit profits case

Figure 5.3 shows an example of rounding of class B calls.

Rounding of class C calls

Our next step is to describe a rounding for the class C calls. The general idea behind the rounding is that we can reassign the $x^*$ values corresponding to a call to be either 0, 0.5 or 1 without losing on their contribution to the objective value or feasibility. These $x^*$ values can then be rounded to 0 or 1. However, to maintain feasibility we will need to throw away a constant number of calls, bounded by 2 from above. We start with a lemma that does the rounding when the two variables corresponding to a call are exactly 0.5 each. A definition is in order before we state the lemma, two edges in a ring are said to be diametrically opposite if the set of vertices on which they are incident is exactly the set of the end points of two crossing calls. We assume that every vertex in the ring is an end point of a call. If not, we could merge such a vertex with one of its neighbours without losing generality.

Lemma 5.1.4 (Rounding an even number of pairwise crossing calls)

Consider a set of $2m$ mutually crossing calls in a ring and a route function $x$ such that $x_{i1} = x_{i2} = 0.5$, $i = 1, 2, ..., 2m$. There exists a $\{0, 1\}$-route function $x'$ and a call $j$, $1 \leq j \leq 2m$ such that

(i) $x'_{j1} = x'_{j2} = 0$;
(ii) $x'_{i1} + x'_{i2} = 1$, $i \neq j$;
and (iii) the sum of the $x'$ values at any edge is at most the sum of the $x$ values.

Further to the point, the sums of the $x'$ values at diametrically opposite edges in the ring are exactly $m$ and $m-1$, respectively.

Proof: Proof by induction on $m$. For $m = 1$, there are exactly two crossing calls in the ring, say 1 and 2. Assign $x'_{11} = 1$ and $x'_{12} = x'_{21} = x'_{22} = 0$. It is
easy to verify the conditions of the lemma for this function $x'$.

Let the lemma hold for $m - 1 \geq 1$, we shall prove the statement holds for $m$. Let $2m$ pairwise calls be given. Remove two of these calls which have end points at edge 1 in the ring, call them $i$ and $j$. Now, we have $2m - 2$ calls for which the lemma holds by inductive hypothesis. Thus, there is one call among these $2m - 2$ calls which has been assigned 0 for both its paths and the rest of them received 1 for one of their paths and 0 for the other. Further, the sum of the $x'$ values at an edge was $m - 1$ or $m - 2$. If edge 1 received a sum of $m - 1$ then assign $x'_{j1} = x'_{j2} = 0$ and $x'_{i2} = x'_{j2} = 1$. This makes the sum of $x'$ values at all edges other than 1 and the diametrically opposite edge of 1, go up by exactly one. And, for edge 1 the sum of the $x'$ values remains at $m - 1$ while for the diametrically opposite edge of 1 the sum of the $x'$ values goes up by 2 to $m$. The case when edge 1 received a sum of $x'$ values equal to $m - 2$ is complementary. But, these are exactly the conditions of the lemma for a set of $2m$ calls, which concludes the induction argument and the proof of the lemma.

For the sake of presentation, we introduce a fictitious call with index 0, which refers to none of the calls in the input instance.

**Corollary 5.1.5 (Rounding pairwise crossing calls)** Consider a set of $m$ mutually crossing calls in a ring and a route function $x$ such that $x_{i1} = x_{i2} = 0.5, i = 1, 2, \ldots, m$. There exists a $\{0, 1\}$-route function $x'$ and a call $j, 0 < j < m$ such that

(i) $x'_{j1} = x'_{j2} = 0$,

(ii) $x'_{i1} = x'_{i2} = 1$, $i \neq j$.

and (iii) the sum of the $x'$ values at any edge is at most the sum of the $x$ values rounded up.

Also, $\sum_{i=1}^{m} (x'_{i1} + x'_{i2}) = \left\{ \begin{array}{ll} \sum_{i=1}^{m} (x_{i1} + x_{i2}) - 1 & \text{for even } m \\ \sum_{i=1}^{m} (x_{i1} + x_{i2}) & \text{for odd } m \end{array} \right.$

**Proof:** If $m$ is even, the statement directly follows from Lemma 5.1.4. If $m$ is odd, introduce a new call 0 by splitting a pair of diametrically opposite edges into two. The new vertices are the end vertices of call 0. It is clear that this new call crosses all the other calls. Assign $x_{01} = x_{02} = 0.5$. The sum of the $x$ values at any edge is $(m + 1)/2$, the rounded up sum of the $x$ values of the $m$ calls. Now, invoking the previous lemma the $x'$ values of one of the $(m + 1)$ calls are set to zero. By symmetry, we can set $x'_{01} = x'_{02} = 0$. 
Thus, for the odd case, \( j = 0 \). Now, \( x'_{i1} + x'_{i2} = x_{i1} + x_{i2}, \forall i \neq j \). Hence,

\[
\sum_{i=1}^{m}(x'_{i1} + x'_{i2}) = \sum_{i, i \notin \{0,j\}}(x_{i1} + x_{i2}) - \sum_{i=j}^{m}(x_{i1} + x_{i2})
\]

\[
= \begin{cases} 
\sum_{i=1}^{m}(x_{i1} + x_{i2}) - 1 & \text{if } m \text{ even, since } j \neq 0 \\
\sum_{i=1}^{m}(x_{i1} + x_{i2}) & \text{if } m \text{ odd, since } j = 0
\end{cases}
\]

Also, it follows that the sum of the \( x' \) values at an edge is at most \( \lceil m/2 \rceil \), which is the sum of the \( x \) values at the edge rounded up.

Refer to Figure 5.4 for how the mutually crossing calls with \( x_{i1} = x_{i2} = 0.5 \) are routed by the rounding scheme.

Recall that class C calls had their corresponding \( x^* \) values summing to exactly one. We have just shown that if these \( x^* \) values are 0.5 each then there exists a rounding that loses at most 1 call compared to an optimal solution. The next step is to show how to achieve half-integer values from arbitrary ones. First, we will discard one of the calls from the set of crossing calls. Next, for the remaining calls we appeal to the powerful Okamura-Seymour theorem in [OS81] to get half-integer values.

\textbf{Theorem 5.1.6 (Okamura-Seymour theorem)} If \( G = (V, E, c) \) is a planar graph with capacities on the edges and can be drawn such that vertices \( s_1, s_2, \ldots, s_k, t_1, t_2, \ldots, t_k \) are all on the boundary of the infinite region, then the following are equivalent:

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5.4.png}
\caption{Rounding of class C calls. There are 6 mutually crossing calls corresponding to the chords. The rounding scheme routes 5 of them.}
\end{figure}
(i) For $1 \leq i \leq k$ there is a flow $F_i$ from $s_i$ to $t_i$ of value $q_i$ such that for each edge $e \in E$

$$
\sum_{i=1}^{k} |F_i(e)| \leq c(e).
$$

(ii) For each $X \subseteq V$,

$$
\sum_{e \in \partial(X)} c(e) \geq \sum_{i \in D(X)} q_i
$$

($\partial(X) \subseteq E$ is the set of edges with one end in $X$ and the other in $V \setminus X$. $D(X) \subseteq \{1, 2, \ldots, k\}$ is $\{i : 1 \leq i \leq k, \{s_i, t_i\} \cap X \neq \emptyset \neq \{s_i, t_i\} \cap (V \setminus X)\}$.)

Furthermore, if $q$ and $c$ are integer valued, then the flows $F_i$ may be chosen half-integer valued.

The relation between the theorem and our problem is readily apparent. The ring is a planar graph and all the vertices in it indeed lie on the outer infinite face. The flows correspond to the paths connecting the end vertices. Thus, if we are able to show that the mutually crossing $C$ calls satisfy condition (ii) of the theorem then we can obtain half-integer valued flows (or equivalently, half-integer values for the routes of a call). Lemma 5.1.9 addresses this. But, first we need to identify one call among the class $C$ calls which will be discarded for the above theorem to be applied.

We start with some more terminology. Given a ring on $2m$ vertices, two edges are almost diametrically opposite if they have $m - 2$ edges between them. Note that between any pair of diametrically opposite edges there are exactly $m - 1$ edges. For every edge there is exactly one diametrically opposite edge and there are two almost diametrically opposite edges. For a set of $m$ mutually crossing calls with a route function $x$ with $x_{i1} + x_{i2} = 1, \forall i$, the total of the rounded down sums of $x$ values at diametrically opposite edges is at least $m - 1$ and for almost diametrically opposite edges is at least $m - 2$.

**Lemma 5.1.7** Consider a set of $m$ mutually crossing calls in a ring with a route function $x$ such that $x_{i1} + x_{i2} = 1, \forall i$ and $x_{i1}, x_{i2} \notin \{0, 1\}$. Let there be an edge $e_0$ in the ring such that the total of the rounded down sum of $x$ values through it and the rounded down sum of $x$ values through its almost diametrically opposite edge is $m - 2$. There exist two consecutive edges in the ring such that the rounded down sums of the $x$ values through them are equal.
Proof: Assume to the contrary that for every pair of consecutive edges the rounded down values are unequal. Let the sum of the $x$ values at an edge $e$ be $x(e)$. For two consecutive edges $e, e'$ it is true that $|x(e) - x(e')| < 1$. Therefore, $|x(e) - x(e')| = \pm 1$. Consider the edge $e_0$ and one of its almost diametrically opposite edges $e_{m-1}$ in the ring. They have $m-2$ edges between them (traversing the ring in one of the two possible ways). Denote the edge that has exactly $k - 1$ edges between it and $e_0$ in this above traversal by $e_k, k = \{1, 2, ..., m-1\}$. It can be proved that $|x(e_0) - x(e_k)| \in \{\pm k, \pm(k-2), \pm(k-4), ..., \pm(k-2[\frac{m}{2}])\}$. Indeed for $k = 1$, it is trivially true. For $k \geq 2$, $|x(e_0) - x(e_k)| \in \{x(e_{k-1}) - x(e_k)| \pm 1\}$, since $|x(e_{k-1}) - x(e_k)| = \pm 1$. From here, the above statement follows. We now have $|x(e_0) - x(e_{m-1})| \in \{\pm(m-1), \pm(m-3), ..., \pm(m-(2[\frac{m}{2}]-1))\}$. But, $|x(e_0) + x(e_{m-1})| = m-2$, implying $|x(e_0) - x(e_{m-1})| = m-2-2|x(e_{m-1})|$. Or $|x(e_0) - x(e_{m-1})| \in \{\pm(m-2), \pm(m-4), ..., \pm(m-2[\frac{m}{2}])\}$. A contradiction. Thus our hypothesis that no two successive edges have equal rounded down sum of $x$ values is impossible, proving the claim.

\[\square\]

Corollary 5.1.8 Consider a set of $m$ mutually crossing calls together with a route function $x$ such that $x_1 + x_2 = 1, \forall i$. There exist two consecutive edges such that the total of the rounded down sums of $x$ values at each edge and the diametrically opposite edge of the other consecutive edge is at least $m-1$. (Note that the diametrically opposite edge of an edge is an almost diametrically opposite edge of each of its consecutive edges.)

Proof: If no pair of consecutive edges satisfying the conditions stated in the corollary exists, we can arrive at a contradiction. It follows from this assumption that there exists a pair of almost diametrically opposite edges whose total of the rounded down sums of $x$ values is $m-2$. By Lemma 5.1.7, there are two consecutive edges $e_1, e_2$ such that their rounded down sums of the $x$ values are equal. But, the rounded down sums of the $x$ values of diametrically opposite edges add up to at least $m-1$. Let $e'_1, e'_2$ be diametrically opposite edges of $e_1, e_2$, respectively. For two consecutive edges, one’s diametrically opposite edge is the other’s almost diametrically opposite edge. Thus, the rounded down sums of $x$ values of $e_1(e_2)$ and its almost diametrically opposite edge $e'_2(e'_1)$ add up to at least $m-1$. This contradicts our supposition.

\[\square\]

Now, we want to identify a call such that the set of mutually crossing calls excluding it satisfy condition (ii) of Theorem 5.1.6 when the capacity
on an edge is equal to the rounded down sum of the $x$ values through it. We shall use the edges $e_1, e'_1, e_2, e'_2$, identified in the above proof, towards this end. Let the call, one end vertex of which is incident on edges $e_1, e_2$, be $k$

A call is said to cross a set of edges in the ring if its end vertices lie in two different components of the graph obtained after removal of the set of edges from the ring. Consider a cut formed by any two edges $e_1, e_r$ in the ring. If call $k$ crosses the cut formed by $e_1, e_r$, then the total of the rounded down sums of $x$ values through them is at least the number of calls crossing them excluding call $k$. To see this, observe that every call that crosses the cut $e_1, e_r$ contributes a value of 1 to the sum of $x$ values on these edges. Thus, the sum of the $x$ values on the cut $e_1, e_r$ due to the crossing calls is an integer. Rounding the $x$ sums down would decrease the total by at most one. Hence, the rounded down sums of $x$ values at $e_1, e_r$ would be at least the number of crossing calls minus one. On the other hand, if they lie on the same side of call $k$ the above sum is at least the number of calls crossing them. Indeed, the total of the rounded down sums of $x$ values at $e_1$ and $e'_2$ is $m-1$. Without loss of generality, assume that all four edges, $e_1, e_r, e_1, e'_2$ lie on the same side of call $k$ and appear in that order on a clockwise traversal of the edges. The difference in rounded down sum of $x$ values at consecutive edges is at most 1. Thus, if there are $t$ vertices between $e_r$ and $e_1$ and $s$ vertices between $e_1$ and $e'_2$, the total of the rounded down sums of the $x$ values at $e_1$ and $e_r$ is at least $m - 1 - s - t$. But this value is exactly the number of calls crossing $e_1$ and $e_r$. Thus, we can conclude that after removing call $k$ from the set of mutually crossing calls, for any pair of edges in the ring the total of the rounded down sums of $x$ values through them is at least the number of calls crossing them. We are in a position to show what we set out to prove at the beginning of this paragraph. The details are in the lemma below.

**Lemma 5.1.9 (Half-integer rounding of crossing calls)** Consider a set of $m$ mutually crossing calls on a ring and a route function $x$ such that $x(i) + x(i) = 1, \forall$ calls $i$. There is a half-integer route function $x'$ and a call $j$ such that

(i) $x_{j1} = x_{j2} = 0$,
(ii) $x_{i1} + x_{i2} = 1, \forall$ calls $i \neq j$
(iii) the sum of the $x'$ values at an edge $e$ is at most the sum of the $x$ values rounded down.

**Proof:** For the call $j$, one of whose end vertices is incident on the consecutive edges identified in the proof of Corollary 5.1.8, set $x_{j1} = x_{j2} = 0$. For every edge, assign a capacity equal to the rounded down sum of the $x$ values through it. Now, consider any pair of edges in the ring. By the argument above, for every cut with two edges the inequality in condition (ii)
of Theorem 5.1.6 is satisfied. In [OS81] it is shown that if the inequality in condition (ii) of Theorem 5.1.6 holds for cuts that separate the planar graph into two connected components then it also holds for all cuts. For a ring, a cut with 2 edges separates it into two connected components while a cut with more than 2 edges separates it into more than 2 connected components. Therefore, the inequality in condition (ii) of Theorem 5.1.6 must hold for all cuts in the ring. Hence, the existence (and construction) of \(x'\) follows from it (as the proof of Okamura-Seymour theorem is also constructive).

Lemma 5.1.9 in conjunction with Corollary 5.1.5 yields an integer rounding that is close to the fractional optimum by an additive constant of 2.

Finally, we can state the performance guarantee of the rounding scheme of crossing calls as follows:

**Corollary 5.1.10 (Rounding class C calls)** Consider a set of class C calls \(S(c)\) on a ring with a corresponding route function \(x^*\). There exists a \(\{0,1\}\)-route function \(x'\) such that

(i) for every edge \(e\) the sum of \(x'\) values of routes through it is at most the rounded down value of the sum of the \(x^*\) values.

(ii) \(x'(S(c)) = \sum_{i \in S(c)} (x_{i1}' + x_{i2}') \geq \sum_{i \in S(c)} (x_{i1}^* + x_{i2}^*) - 2 = x^*(S(c)) - 2.\)

**Proof:** Lemma 5.1.9 shows a rounding of \(x^*\) to half-integer values \(x\) losing one on the sum of the \(x^*\) values. If all calls got \(x\) values in \(\{0,1\}\) we are already done by setting these \(x\) values to be \(x'\) values. Otherwise, applying Corollary 5.1.5 on those calls that got \(x\) values 0.5 for both their variables we get a \(\{0,1\}\) route function \(x'\). The sum of the \(x'\) values is at most one less than the sum of the \(x\) values. Thus, in total we lose at most 2 from the sum of \(x^*\) values. Condition (i) follows from Lemma 5.1.9.

This settles the corollary.

**Assembling the pieces**

In this section, we shall piece together the different parts for solving the off-line RCC problem. Starting from the optimal fractional solution \(x^*\) to the relaxed LP, we adjust the values such that \(x^*\) is in the canonical form with respect to parallel and crossing calls, as set forth in the beginning of Subsection 5.1.2. If there is a class D call then make it a class B call by setting the lower of the two indicator variables to zero, say \(x_{D2}^*\) is the smaller of the
two. Next, perform the rounding on class B and class C calls as described in the Corollaries 5.1.3 and 5.1.10. For class B calls the sum of the rounded values at any edge is at most the rounded up value of the sum of the original $x^*$ values and for class C calls it is at most the rounded down value of the sum of the original $x^*$ values. Thus, combining the two sums at an edge will satisfy its capacity constraint. In other words, the rounded solution is a feasible one. As regards the objective value,

$$OPT^* = x^*(S_{(a)}) + x^*(S_{(b)}) + x^*(S_{(c)}) + x_{D1}^* + x_{D2}^*$$

$$< x^*(S_{(a)}) + [x^*(S_{(b)}) + x_{D1}^*] + x^*(S_{(c)}) + 0.5, x_{D2}^* < 0.5$$

$$\leq x^*(S_{(a)}) + x^*(S_{(b)}) + x_{D1}^* + 1 + x^*(S_{(c)}) + 2 + 0.5$$

$(x^*(S_{(a)}) = x^*(S_{(b)}) = 0$, rounding of class D call with the class B calls)$$

$$= x^*(S_{(a)}) + x^*(S_{(b)}) + x_{D1}^* + x^*(S_{(c)}) + 3.5$$

$$= x^*(S) + 3.5, S = S_{(a)} \cup S_{(b)} \cup S_{(c)} \cup \{D\}, x_{D2}^* = 0$$

But, $\lfloor OPT^* \rfloor$ is an upper bound on the objective value of the integer linear program. Therefore, the rounded solution is at most 3 away from an integer optimal solution to the ILP. Yielding,

**Theorem 5.1.11 (An “almost” optimal solution for RCC on rings)**

Consider an instance of off-line, uniform RCC given by an edge capacitated ring $C = (V, E, c)$, a set $S$ of $m$ calls with unit bandwidth requirements and unit profits. There is a polynomial time algorithm that produces a feasible solution routing at most 3 fewer calls compared to an optimal solution.

**A PTAS for off-line, uniform RCC on rings**

**Theorem 5.1.12** Given $\epsilon > 0$ and an instance of off-line, uniform RCC on a ring with all calls having a profit of 1 there is a polynomial time algorithm that achieves an approximation ratio of $1 + \epsilon$.

**Proof:** Let the rounding method output a feasible solution $A$. Denote by $OPT$ the number of calls accepted and routed in an optimal solution. We know that $|A| \geq OPT^* - 3.5$. Let $\epsilon' = \frac{\epsilon}{1+\epsilon}$. If $OPT \leq OPT^* \leq \frac{3.5}{\epsilon'}$, a constant, then we can try all possible $(m \choose k)$ subsets of the call set that have cardinality $k$, $1 \leq k \leq \frac{3.5}{\epsilon'}$. For each call in a subset, we can try the two possible routings. Finally, output the solution which has maximum cardinality and is feasible. Plainly, this takes time $O(m^{O(\frac{3.5}{\epsilon'})})$. On the other
5.2. Arbitrary profits case

We consider the same problem as in the previous section except that calls are now associated with an arbitrary profit function $p : S \rightarrow \mathbb{N}$. The objective,
in this case, is to find a feasible solution with maximum profit. We provide a simple 2-approximation for the problem. As most of the details overlap with those presented in the next chapter, we give a simple description of the algorithm here and refer the reader to the next chapter to fill in the missing details.

Let \( e_{\text{min}} \) be an edge of the ring \( C = (V, E, c) \) with the minimum capacity. Prohibit the routes of calls in \( S \) from passing through it. In this case, the routes of all calls in \( S \) are fixed on the line \( L = C \setminus \{e_{\text{min}}\} \). We show in the next chapter that it is possible to compute in polynomial time an optimal feasible set of calls \( S^*_{e_{\text{min}}} \subseteq S \) on \( L \). Next, order the calls in \( S \) in the decreasing order of their profits \( p(.) \). Let \( S^*_{e_{\text{min}}} \) be the first \( c(e_{\text{min}}) \) calls that appear in the order. For every call in \( S^*_{e_{\text{min}}} \) there is a unique route in \( C \) that passes through the edge \( e_{\text{min}} \). Since \( e_{\text{min}} \) has the minimum capacity, these routes are feasible in \( C \). Hence, \( S^*_{e_{\text{min}}} \) is feasible in \( C \). Output one of the sets \( S^*_{e_{\text{min}}} \), which has the maximum total profit, together with the appropriate routes for accepted calls. Analysis of this algorithm (similar to the one in the next chapter) verifies that it is a 2-approximation.

5.3 Remarks

The computational complexities of both these problems have remained unresolved even after our best efforts. Nothing we have expressed here discounts the possibility of finding polynomial time optimal algorithms for these problems. One restriction that allows optimal solutions to be found in polynomial time is the following. Let the minimum edge capacity be bounded by a constant. The essential idea of an optimal algorithm is to, first, try all possibilities of routing calls through one of the edges with minimum capacity (this is polynomial in the input size). Second, for each of the possibilities consider the call control problem on the line, obtained after removing this edge, and the remaining set of calls. The edge capacities on the line are set to be the original capacities minus the number of calls routed through them in the possibility that is being considered. For the call control problem on the line, an optimal feasible solution can be computed in polynomial time (as shown in next chapter). The optimum solution for the original problem can be found by going through optimal solutions of the enumerated possibilities.
Pre-routed Call Control

A 2-approximation for the off-line, uniform PCC problem on rings is presented. The computational complexity of the problem is shown to be closely linked to that of a matching problem in bipartite graphs. The latter half of the chapter discusses special cases of the problem and provides polynomial time optimal algorithms or approximation schemes for them.

We turn to the pre-routed variant of the problems discussed in Chapter 5, the corresponding PCC problems. In addition to the call set, we are also provided with a pre-specified route for every call. We showed in [AAAE02] that when the profits are 1 for all the calls, the off-line, uniform PCC problem on rings is polynomially solvable to optimality. When the profits are arbitrary the problem seemingly gets harder, though. Here, we show a 2-approximation for the problem in Section 6.1. Further, we provide an indication of the difficulty of finding a polynomial time optimal algorithm for it. We do so by showing that if there exists a polynomial time algorithm for the PCC problem then the “exact matching” problem on bipartite graphs can be solved in polynomial time. The latter problem has been studied for well over 15 years without being resolved to be in \(\mathcal{P}\) or \(\mathcal{NP}\)-complete. We use a result of Hochbaum et al. [HL03] for proving the above linkage. Finally, in Sections 6.2 and 6.3 we console ourselves by identifying a variety of special cases of the off-line, uniform PCC problem on rings for which either optimal polynomial time algorithms or (randomized) approximation schemes can be designed. It is satisfying to note that for two important special cases we can prove polynomial time solvability. This leaves only one special case, namely when the profits of pairs of paths do not satisfy the restricted profits property (see Subsection 6.2.3), unresolved in terms of complexity. An interesting side-light while designing these algorithms is the derivation of an integer optimal solution to the “dual” of the general off-line, uniform PCC problem.
6.1 Off-line, uniform PCC on rings

The off-line, uniform PCC problem on rings, formally, is the following:

**Input:** An edge capacitated ring \( C = (V, E, c) \) on \( n \) vertices, a call set \( S \) of \( m \) calls with unit bandwidth requirements together with, for each call, one of the two paths as its pre-specified route and a profit function \( p : S \rightarrow N \).

A feasible solution is a subset \( S' \subseteq S \) such that the number of routes of calls in \( S' \) through an edge \( e \in E \) is at most \( c(e) \), the capacity of the edge.

The profit of the feasible solution \( S' \), denoted \( p(S') \), is the sum of the profits of the calls in \( S' \).

**Objective:** A feasible solution \( OPT \subseteq S \) with maximum possible profit.

As in the approach to solving the RCC problem with unit profits, we shall formulate this problem as an integer linear program (ILP) and then show a rounding mechanism.

### 6.1.1 An integer linear program

Let the call set be \( S = \{\{u_i, v_i\} : i = 1, 2, ..., m\} \). Since the routes for calls are specified with the call set we have exactly one indicator variable \( x_i \) for call \( i \) corresponding to whether the call is accepted and routed along its pre-specified path. Let \( S_e = \{i : \text{call } i \text{ is routed through edge } e\} \), for edge \( e \in E \). The ILP can now be stated as:

\[
\begin{align*}
\text{max} & \sum_{i=1}^{m} p(i) \cdot x_i \\
\text{subject to} & \sum_{i \in S_e} x_i \leq c(e), \quad e \in E \\
& x_i \in \{0, 1\}
\end{align*}
\]

As in the previous chapter, we shall call the vector \( x \) a route function. Here, the interpretation would be that a call \( i \) is accepted when \( x_i = 1 \) and is rejected if \( x_i = 0 \). For the relaxation of the ILP, the fractional values of a feasible solution will be interpreted as a function that assigns these fractional values to the routes of the calls. A route function is called feasible if it satisfies the capacity constraints on the edges.

### 6.1.2 A 2-approximation

We shall start with an easy corollary to Lemma 5.1.1 and then show a weighted version of it which will be essential in deriving the 2-approximation.

**Corollary 6.1.1** Let \( R \) be a set of paths on a line \( L = (V, E, c) \). Consider a route function \( x_{\text{max}} : R \rightarrow [0, 1] \) such that the sum of the \( x_{\text{max}} \) values at
an edge e is at most \( c(e) \) and further \( x_{\text{max}}(R) \geq x(R) \), for any feasible route function \( x \). There exists a \( \{0,1\}\)-route function \( x'_{\text{max}} : R \to \{0,1\} \) such that for all \( e \in S_e \), \( x'_{\text{max}}(i) \leq c(e) \) and \( x'_{\text{max}}(R) = x_{\text{max}}(R) \).

**Proof:** The existence of a feasible \( x'_{\text{max}} \) such that \( x'_{\text{max}}(R) \geq \lfloor x_{\text{max}}(R) \rfloor \) follows directly from Lemma 5.1.1. But by hypothesis, \( x_{\text{max}}(R) \) is the greatest possible sum among all feasible route functions. Therefore, \( x_{\text{max}}(R) \geq x'_{\text{max}}(R) \) and thus, \( x'_{\text{max}}(R) = x_{\text{max}}(R) \).

When the set of paths on the line have profits \( p(.) \) associated with them, a similar statement is true with respect to the sum of the profits. However, this does not follow from the rounding scheme described in Lemma 5.1.1 but from the theory of totally unimodular matrices and network flows. A matrix \( A \) is said to be totally unimodular if the determinant of every square sub-matrix of \( A \) (including \( A \), if \( A \) is a square matrix) is 0, 1, or \(-1\). A consequence of a matrix \( A \) being totally unimodular is that if it appears as the constraint matrix of a linear program \( \max \{c^T x : Ax \leq b, 0 \leq x \leq 1\} \), the LP has an integer optimal solution whenever \( b \) is integral (see, for example, Corollary 5.20b in [Sch03, p. 76]). It is a well known fact that a \( (0,1) \)-matrix in which the ones appear consecutively in every column (or row) is totally unimodular. From these observations, we have the following weighted version of Corollary 6.1.1.

**Lemma 6.1.2** Consider a set of paths \( R \) on a line \( L = (V, E, c) \) with a profit function \( p: R \to \mathbb{N} \). There exists a \( \{0,1\}\)-route function \( x' \) such that

(i) \( x' \) is feasible

(ii) \( \sum_{i \in R} p(i) \cdot x'(i) \geq \sum_{i \in R} p(i) \cdot x(i) \), for every feasible route function \( x \)

and (iii) \( x' \) can be computed efficiently.

The 2-approximation for the PCC problem is rather simple. Identify an edge \( e_{\text{min}} \) on the ring which has the least capacity \( c(e_{\text{min}}) = c_{\text{min}} \). Consider the line obtained by removing the edge \( e_{\text{min}} \) from the ring and the set of calls that are not routed through \( e_{\text{min}} \), namely \( S \setminus S_{e_{\text{min}}} \). Lemma 6.1.2 asserts that there is a \( \{0,1\}\)-route function on \( S \setminus S_{e_{\text{min}}} \) that achieves the optimum profit and is feasible as well. Denote the set of calls in this route function that are assigned unity by \( S'_{e_{\text{min}}} \). Arrange the set of calls in \( S_{e_{\text{min}}} \) in descending order of their profits. Pick the first \( c_{\text{min}} \) calls in this order and call the set \( S^*_{e_{\text{min}}} \). Both the sets \( S^*_e, S^*_{e_{\text{min}}} \) are feasible solutions to the PCC problem. Let \( OPT \) represent an optimal solution to the PCC problem on the ring. Trivially, the sum of profits of calls in \( OPT \) routed through \( e_{\text{min}} \), \( p(OPT_{e_{\text{min}}}) \), is at most
Further, sum of profits of calls in OPT not routed through $e_{emin}$, $p(OPT_{e_{emin}})$, is at most $p(S^*_{e_{emin}})$. Hence,

\[
p(OPT) = p(OPT_{e_{emin}}) + p(OPT_{e_{emin}}) \\
\leq p(S^*_{e_{emin}}) + p(S^*_{e_{emin}}) \\
\leq 2\max\{p(S^*_{e_{emin}}), p(S^*_{e_{emin}})\}
\]

By choosing the set which has maximum profit among \{$S^*_{e_{emin}}$, $S^*_{e_{emin}}$\} we get a 2-approximation algorithm.

The above algorithm and the analysis can be modified to get a better ratio when the maximum of the lengths of the routes of calls is bounded by $L$. Let OPT be the set of calls in an optimal set and $p(OPT)$ their profit. For every edge $e$ in the ring, add up the profits of calls that are routed through it. Let this sum be $W$. In this sum, each call in OPT has its profit added exactly the length of its route many times. This alternative way of counting $W$ gives us $W = \sum_{i \in OPT} l_i \cdot p(i)$, where $l_i$ is the length of the route of call $i$. But, $l_i \leq L, \forall i$. Therefore, $W \leq Lp(OPT)$. There are $n$ edges in the ring. Invoking an averaging argument, there must exist an edge at which the sum of the profits of calls which are routed through it is at most $Lp(OPT)/n$. Thus, the profit of calls routed not through $e$ must sum up to at least $(n - L)p(OPT)/n$. But by Lemma 6.1.2 we can find an optimal profit set of calls that were not routed through $e$. Thus, the approximation ratio of the algorithm that computes the optimal profit obtained by excluding paths that pass through a particular edge and outputs the highest profit among the iterations for the $n$ edges of the ring, is at most $n/(n - L)$. For values of $L < n/2$, this is better than 2.

6.1.3 Notes on the complexity of off-line, uniform PCC

A problem that is closely related to the off-line, uniform PCC on rings is the cyclical scheduling problem. Hochbaum and Levin [HL03] studied it in the following setting. A set of workers in a factory are to be assigned to various shifts during a day, say. This assignment is repeated for several days. A day is treated as a set of non-overlapping time slots. A shift is a set of consecutive slots that might carry over from one day to the next. There is a cost associated with assigning a worker to a shift. For each time slot during the day there is a requirement in terms of the minimum number of workers working during that slot. The object is to minimize the cost of assigning workers to shifts such that the requirements for all time slots are met. This problem can be formulated as a bounded multicover problem (MCB) and we
can write an ILP as follows:

\[
\begin{align*}
\min & \sum_{j=1}^{m} c_j x_j \\
\text{subject to} & \sum_{j=1}^{m} a_{ij} x_j \geq b_i, i = 1, 2, \ldots, n \\
& 0 \leq x_j \leq u_j, j = 1, 2, \ldots, m \\
& x_j \in \mathbb{Z}
\end{align*}
\]

Here, \( A = (a_{ij}) \) is an \( n \times m \) \((0, 1)\)-matrix. The columns of \( A \) correspond to the possible shifts. \( a_{ij} = 1 \) if shift \( j \) contains the time slot \( i \), otherwise \( a_{ij} = 0 \). \( c_j \) is the cost of assigning a worker to the \( j \)th shift. \( b_i \) is the requirement for the \( i \)th slot during the day and \( u_j \) is the limit on the number of workers available for assignment to the \( j \)th shift. The matrix \( A \) exhibits the circular ones property along the columns, i.e. the ones along a column of the matrix are consecutive when the first constraint is viewed as following the last. This corresponds to the fact that shifts carry over from one day to the next.

Hochbaum et al. [HL03] showed that any instance of the exact matching problem on bipartite graphs can be written as an ILP which can be transformed to an MCB problem with the matrix \( A \) having the circular ones property. The exact matching problem is defined as follows. Given a graph \( G = (V, E) \), a subset of the edges \( E' \subseteq E \) and a positive integer \( k \leq |E'| \), is there a perfect matching in \( G \) with exactly \( k \) edges from \( E' \). This problem, proposed by Papadimitriou and Yannakakis [PY82], is known to be solvable in randomized polynomial time [MVV87]. But, to date it has not been determined whether the problem is solvable in (deterministic) polynomial time even when \( G \) is restricted to be bipartite.

Consider the exact matching problem on a bipartite graph given by the graph \( G = (U \cup V, E) \), the subset of edges \( E' \subseteq E \) and a positive integer \( k \leq |E'| \). One can write an ILP for it as follows:

\[
\begin{align*}
\min & \sum_{(u,v) \in E} 0 x_{uv} \\
\text{subject to} & \sum_{(u,v) \in E'} x_{uv} = k \\
& \sum_{v \in V \setminus \{u,v\} \in E} x_{uv} = -1, \forall u \in U \\
& \sum_{u \in U \setminus \{u,v\} \in E} x_{uv} = 1, \forall v \in V \\
& 0 \leq x_{uv} \leq 1, \forall \{u,v\} \in E \\
& x_{uv} \in \mathbb{Z}
\end{align*}
\]

It is easy to see that any feasible integer solution, if one exists at all, for
this program is a solution to the exact matching problem in \( G \). For every vertex in \( U \) and \( V \) exactly one edge is picked in the solution and the number of edges picked from the set \( E' \) is exactly \( k \).

To transform the above linear program into an MCB problem with circular ones property, first replace every variable \( x_{uv} \) corresponding to the edges \( \{u,v\} \in E \setminus E' \) by substituting \( y_{uv} = 1 - x_{uv} \). The bounds for these \( y_{uv} \) are exactly the same as those of the corresponding \( x_{uv} \), namely \( 0 \leq y_{uv} \leq 1 \).

Update the constraints so that the constants appear on the right hand sides. Next, consider the constraints in the above order; the first constraint is for the set \( E' \) followed by all constraints for vertices in \( U \) which are then followed by the constraints for vertices in \( V \). Add the first constraint to the second constraint to obtain a new second constraint. Add the new second constraint to the third to obtain a new third constraint and so on till the last constraint. Notice that the new set of constraints are equivalent to the old ones as all of them are equalities.

We observe that the ones appear in a circular fashion along the columns (looking at it in the above order). For the variables \( x_{uv}, \{u,v\} \in E' \), the ones wrap around and for the variables \( y_{uv}, \{u,v\} \in E \setminus E' \), the ones appear consecutively. Let the new constraint matrix be \( A \) where the first row is the constraint on the edge set \( E' \), the subsequent rows are constraints for the vertices \( u \in U \) followed by the constraints for vertices \( v \in V \). Let the rows of \( A \) be \( A_{E'}, A_u, A_v \) and the right hand sides be \( b_{E'}, b_u, b_v \). Let the vectors \( \mathbf{1}, \mathbf{x} \) and \( \mathbf{b} \) represent the all 1s vector, the set of variables and the right hand sides of the constraints, respectively. Let \( s_u = A_u \mathbf{x} - b_u, s_v = A_v \mathbf{x} - b_v, s_{E'} = A_{E'} \mathbf{x} - b_{E'} \) and \( y = 1 - \mathbf{x} \). We can now write the transformed linear program as:

\[
\begin{align*}
\min \sum_{u \in U} s_u + \sum_{v \in V} s_v + s_{E'} & \quad \text{max } \sum_{u \in U} A_u y + \sum_{v \in V} A_v y + A_{E'} y \\
\text{subject to} & \quad \text{subject to} \\
A \mathbf{x} \geq \mathbf{b} & \quad A y \leq A \cdot \mathbf{1} - \mathbf{b} \\
0 \leq \mathbf{x} \leq 1 & \quad 0 \leq \mathbf{y} \leq 1 \\
\mathbf{x} \text{ is an integer vector} & \quad \mathbf{y} \text{ is an integer vector}
\end{align*}
\]

It is not too difficult to see that the program on the left is an MCB problem (The program on the right of the equivalence is obtained by substituting \( y = 1 - \mathbf{x} \) in the left of the equivalence and omitting the constant term in the objective function). In addition, \( A \) has the circular ones property. The lower and upper bounds on the variables are 0 and 1 respectively. Now, we can produce an instance of off-line, uniform PCC on rings from the linear program on the right of the equivalence. Associate an edge to each constraint. Two edges are incident on each other if their corresponding constraints are consecutive. The edges corresponding to the first and last constraints are
incident on each other, completing the ring. The capacities on the edges are the right hand side values in the corresponding constraints. Note that these values have to be positive if there is a feasible solution to the original problem. The calls of the instance are obtained from each of the columns of $A$. The pre-specified path of a call corresponding to a column consists of those edges whose corresponding constraints have 1s in that column. The profit of a call is the coefficient of the corresponding variable in the objective function. Note that the objective of maximizing the profits of calls that can be accepted in the PCC problem is equivalent to the minimization objective of the MCB problem (The latter is simply the negative of the former in addition to a constant term). Thus, every instance of exact matching on bipartite graphs can be turned into an instance of off-line, uniform PCC on rings. An optimal solution to the instance of the call control problem would give us a solution to the underlying exact matching problem on bipartite graphs; on an optimal solution for the PCC problem, we apply the inverse of the sequence of transformations that were applied on the original variables, but in the reverse order. Thus, if we knew that the off-line, uniform PCC problem on rings is in $P$ then the exact matching problem on bipartite graphs would also be solvable in $P$. We conclude this section by summarizing the above discussion in the following observation.

**Observation:** There is a polynomial time reduction of the exact matching problem on bipartite graphs to the off-line, uniform PCC problem on rings. An optimal polynomial algorithm for the latter yields a polynomial algorithm to solve the former.

### 6.2 Optimal algorithms for special cases of off-line, uniform PCC

In this section, we consider three special cases of off-line, uniform PCC on rings and show optimal algorithms that run in time polynomial in the size of the input. The various special cases are:

(i) Calls have routes of equal length and their profits are arbitrary.

(ii) Calls have "proper" routes (defined later) and profits are arbitrary.

(iii) Calls have arbitrary routes. The profit of a call whose route is contained in that of another is at least as great as the profit of the latter.

(i) is in fact a special case of (ii). We have identified them as separate cases for reasons of presentation. Our algorithms for all these cases are based on network flow techniques and we shall use the following theorem
Theorem 6.2.1 Any linear program that contains (a) at most one +1 and at most one −1 in each column or (b) at most one +1 and at most one −1 in each row, can be transformed into a minimum cost flow problem.

Theorem 6.2.1 implies that such a linear program can be solved in strongly polynomial time but also an integral optimal solution to the linear program can be found if the right hand side is an integral vector. The best known strongly polynomial algorithm for the minimum cost flow problem has been shown in [Orl93]. If the linear program has at most one +1 and at most one −1 along every column then [AMO93, Chapter 9] show how it can be directly converted to a minimum cost flow problem. In this case, the following theorem in [AMO93, p. 318] confirms the existence of an integral optimal solution.

Theorem 6.2.2 If all arc capacities and supplies/demands of nodes are integer, the minimum cost flow problem always has an integer minimum cost flow.

On the other hand, if the property holds along the rows then the dual of the linear program can be converted to a minimum cost flow problem [AMO93, Chapter 9]. Given an optimal solution for the dual problem an optimal solution for the primal (the original LP) can be found (The details are presented in [AMO93, pp. 315-317]). It is not too difficult to see from there that an integral optimal solution to the dual and integer capacities on the edges of the network formulation imply an integer optimal solution for the primal. Since the edge capacities are integral in our instances, we shall see that we obtain integral optimal solutions for them.

6.2.1 Calls with routes of equal length

For convenience, let us assume that no two calls which have the same end points have been assigned the same route. We shall drop this condition later on. Assume that the routes of all calls have equal length of $L$. Let the vertices of the ring be numbered $0, 1, ..., n - 1$ in a clockwise fashion and edge $i$ be incident on vertices $i$ and $i + 1 \pmod n$, $i = 0, 1, ..., n - 1$. Let the call set be rearranged such that call $i$ is routed by a path containing vertices $i$ through $(i + L) \pmod n$, $i = 0, 1, ..., n - 1$. If no such call appears in the original call set then introduce such a call with profit 0. This does not alter the properties of the original instance. With this rearrangement of indices, for any edge $j$, precisely the following calls pass through it; namely, calls with
indices \((j - L + 1) \mod n\) through \(j\). For \(j \geq L - 1\), this implies all calls with indices \(j - L + 1\) through \(j\). For \(j < L - 1\), all calls with indices 0 through \(j\) and those with indices \((j - L + 1) \mod n\) through \(n - 1\). Thus, we can rewrite the relaxation of the ILP in Subsection 6.1.1 as:

\[
\begin{align*}
\max \sum_{i=0}^{n-1} p(i) \cdot x_i \\
\text{subject to} \\
\sum_{i=j-L+1}^{j} x_i \leq c(j), \quad n - 1 \geq j \geq L - 1 \\
\sum_{i=0}^{j} x_i + \sum_{j=(j-L+1) \mod n}^{n-1} x_i \leq c(j), \quad 0 \leq j < L - 1 \\
0 \leq x_i \leq 1, \quad i = 0, 1, \ldots, n - 1
\end{align*}
\]

Now, define

\[
X(-1) = 0 \\
X(k) = \sum_{i=0}^{k} x_i, \quad k = 0, 1, \ldots, n - 1.
\]

Substituting these new variables in the above LP we obtain:

\[
\begin{align*}
\max \sum_{i=0}^{n-1} p(i) \cdot (X(i) - X(i - 1)) \\
\text{subject to} \\
X(j) - X(j - L) \leq c(j), \quad n - 1 \geq j \geq L - 1 \\
X(j) + X(n - 1) - X((j - L) \mod n) \leq c(j), \quad 0 \leq j < L - 1 \\
0 \leq X(i) - X(i - 1) \leq 1, \quad i = 0, 1, \ldots, n - 1 \text{ corresponding to } 0 \leq x_i \leq 1 \\
X(-1) = 0
\end{align*}
\]

Naturally, for integer solutions, \(X(n - 1)\) is an integer between 0 and \(n\). Thus, we can set \(X(n - 1) = t\), for some integer \(t, 0 \leq t \leq n\). This reduces the constraint matrix to one where each row has at most one \(+1\) and one \(-1\) by taking \(X(n - 1)\) to the right hand side. Also, since \(c(e)\) is an integer, so is \(c(e) - t\), for integer \(t\). That the above LP has an integer optimal solution can be deduced from Theorem 6.2.1 (see comments appearing immediately after the theorem). Moreover, integral optimal solutions can be obtained using network flow techniques. Integer solution for the modified LP implies integer solutions for the original LP as \(x_i = X(i) - X(i - 1), i = 0, 1, \ldots, n - 1\). Note also that if \(X^*_t\) denotes a feasible vector for the above LP with \(X(n - 1) = t\) then \(\lambda X^*_t + (1 - \lambda)X^*_2\) is a feasible solution to the LP when \(X(n - 1) = \lambda t_1 + (1 - \lambda) t_2\). Thus, the approach to solve the original problem is:

Step 1: Set \(P_1 = \frac{t_1}{2}, P_2 = P_1 + 1, P_{\min} = 0, P_{\max} = n\).
Step 2: If for \(X(n - 1) = P_2\) the LP is infeasible and for \(X(n - 1) = P_1\) it is feasible, set \(P_{\max} = P_1\). Else, if for \(X(n - 1) = P_1\) the LP is infeasible, set \(P_{\max} = P_1 - 1\). Otherwise, the LP is feasible for both \(X(n - 1) = P_1\) and
6.2. Optimal algorithms for special cases of off-line, uniform PCC

$X(n-1) = P_2$. If $X^*_p$ has a larger objective value than $X^*_p_2$, set $P_{\text{max}} = P_1$. If $X^*_p$ has a smaller objective value than $X^*_p_2$, set $P_{\text{min}} = P_2$. If the objective values are equal then set $P_{\text{min}} = P_1$ and $P_{\text{max}} = P_2$.

**Step 3:** If $P_{\text{max}} - P_{\text{min}} \leq 1$ output one of the solution vectors $X^*_p_{\text{min}}$ or $X^*_p_{\text{max}}$, whichever is feasible and has larger objective value. Else, set $P_1 = \lceil \frac{P_{\text{min}} + P_{\text{max}}}{2} \rceil$ and $P_2 = P_1 + 1$. Go to **Step 2**.

The property of the ILP for this special case that we exploited was that its constraint matrix has only 0 and 1 entries and the Is appeared circularly along the rows. The technique we employ here, of variable substitution to find an integer optimum and then to do a binary search for finding an optimal solution for the original problem, was devised by Bartholdi et al. [BOR80].

In the foregoing instance we had assumed that no two calls had the same route if they shared the same end points. This can be easily patched. If the call set had $k$ calls all routed along the same path, say, from vertex $i$ to $(i + L) \mod n$ then we create $k$ indicator variables $x_{i_j}^{(j)}$, $j = 1, 2, \ldots, k$, corresponding to the $k$ copies. The new order of the indices of the calls looks as follows; the distinct routes of the calls are ordered as before and the copies of each of the routes are ordered between themselves arbitrarily. The variable transformation described above can be done according to this new order of the indices of the calls. It is easy to see that the arguments continue to hold for this case too. It should be remarked that the profits of calls play no role in the integrality of the optimal vector. In particular, when two calls are routed along the same path they could have arbitrary profits.

### 6.2.2 Calls with “proper” routes

A circular arc graph is defined on a set of open arcs on a circle. The vertices correspond to the individual arcs and an edge is drawn between two vertices if the corresponding arcs intersect. A circular arc graph is said to be proper if no arc is contained in another. Assume that the input to PCC is such that no route of a call is strictly contained in that of another. We call such a set of routes proper. The result given in the previous subsection is in reality a special case of proper routes. However, we can deal with this instance also in a similar manner to the last one. The proper-routes-for-calls assumption implies that we can order the calls according to the increasing order of their clock-wise end points of their routes. Once the indices of calls are rearranged according to this order it follows that through an edge $e$ one of the following occurs:

(a) Calls with indices $1, 2, \ldots, k$, $1 \leq k \leq m$ pass through it.
(b) Calls with indices $1, 2, \ldots, k_1, k_2, k_2 + 1, \ldots, m$, $1 \leq k_1 < k_2 \leq m$ pass through it.
(c) Calls with indices $k, k + 1, \ldots, m, 1 \leq k \leq m$ pass through it.

(d) Calls with indices $k_1, k_1 + 1, \ldots, k_2, 1 < k_1 \leq k_2 < m$ pass through it.

Exactly the same set of transformations as before show that for this case too one can derive an optimal integer solution. The constraints for edges that fall into category (a), (b), (c) and (d) look respectively as $X(k) \leq c(e)$, $X(k_1) + X(m) - X(k_2 - 1) \leq c(e)$, $X(m) - X(k - 1) \leq c(e)$, $X(k_2) - X(k_1 - 1) \leq c(e)$. A binary search, as given in the previous subsection, for values of $X(m)$ between 0 and $m$ outputs an optimal integral solution.

Recall the definition of crossing calls from Subsection 5.1.2. Notice that if the set of calls were mutually crossing then the routes are proper. Thus, the special case of mutually crossing calls can also be solved optimally.

### 6.2.3 Calls with restricted profits

In this case, the profits assigned to calls are restricted. For any pair of parallel calls, the profit of the call whose route is strictly contained in that of the other is at least as great as the profit of the latter.

First, we show that an optimal solution to the ILP in Subsection 6.1.1 can be found if an extra constraint is added to it. Consider the linear program, call it PCC-$Y$, obtained from the relaxation of the ILP in Subsection 6.1.1 by adding the constraint $\sum_{i=1}^{m} x_i = Y$, where $0 < Y < m$ is an integer.

Assume that PCC-$Y$ is feasible. Solve PCC-$Y$ to obtain an optimal vector $y^*$. If $y^*$ is fractional then we can obtain a feasible integral solution (from $y^*$) with the same objective value as the optimal objective value of PCC-$Y$. For any pair of parallel calls $i$ and $j$, with fractional $y^*$ values and the route of $i$ contained in that of $j$, we carry out a transformation similar to the one described in Subsection 5.1.2 of Chapter 5 for parallel calls. This transformation ensures that either $y^*_i = 1$ or $y^*_j = 0$. Since by assumption $p(i) \geq p(j)$, the objective value with this transformation is unchanged. At the end of this transformation, we have fractional $y^*$ values only for calls whose routes are not strictly contained in each other, i.e. the routes of these calls are proper. Notice that since we required that the sum of the components of the solution vector add up to $Y$, an integer, the sum of the $y^*$ values of these calls must also be an integer, say $Z$. Consider the linear program, PCC-$(Y, Z)$, obtained by substituting the $y^*$ values of those calls whose $y^*$ values were 0 or 1 after the transformation, in PCC-$Y$. PCC-$(Y, Z)$ has variables corresponding to calls whose routes are proper and one of the constraints in PCC-$(Y, Z)$ is that the sum of these variables is $Z$. Thus, by the result of the previous subsection, we can find an optimal solution for PCC-$(Y, Z)$ that is integral. The $y^*$ values (after the transformation) of calls in PCC-$(Y, Z)$ are feasible for it. Therefore, the optimal objective value for PCC-$(Y, Z)$ is
at least the objective value of PCC-(Y, Z) with these \( y^* \) values. It is now easy to verify that an integral optimal solution to PCC-Y is the set of those calls whose \( y^* \) values were 1 after the transformation step together with the set of those calls in the optimal solution of PCC-(Y, Z). Indeed, after the transformation step the objective value with new \( y^* \) values was unchanged from what it was before the transformation. The optimal solution of PCC-(Y, Z) has an objective value that is at least the objective value of PCC-(Y, Z) with the new \( y^* \) values as noted above.

Now, we obtain an optimal solution to the original ILP as follows. Solve the relaxation of the ILP, yielding an optimal vector \( x^* \). If \( x^* \) is integral then we are already done. Otherwise, let \( Z = \sum_{i=1}^{m} x_i^* \). If \( Z \) is an integer, then by the above paragraph we can find an integer optimal solution for PCC-Z which is also an optimal solution for the relaxation and, therefore, for the ILP. Otherwise, let \( Z1 = \lfloor Z \rfloor \) and \( Z2 = \lceil Z \rceil \). PCC-Z1 is a feasible program as the components of \( x^* \) can be decreased to satisfy its constraints. Solve PCC-Z1 and PCC-Z2, as explained in the previous paragraph, to obtain integral optimum vectors for them. It is possible that PCC-Z2 is infeasible in which case we consider the optimal integral solution of PCC-Z1 alone. We claim that the solution between the (at most) two integral optimal solutions which has the greater objective value is an optimal solution for the original ILP. Indeed, the non-zero components of an optimal solution for the ILP must sum to an integer, say \( Z^* \). If \( Z^* \) is one of \( Z1 \) or \( Z2 \) then the claim trivially holds. Otherwise, one of \( Z1 \) or \( Z2 \) lies between \( Z \) and \( Z^* \), say \( Z1 \) without loss of generality. We can write \( Z1 \) as a convex combination of \( Z \) and \( Z^* \). A convex combination of an optimal solution of PCC-Z\(^*\) and \( x^* \) would be a feasible solution of PCC-Z1. The objective value of this feasible solution will be a convex combination of the objective values of the above optimal solutions. But then the optimal objective value of PCC-Z1 is at least the minimum of the objective values of the two solutions. Therefore, an integral optimal solution of PCC-Z1 is also an optimal solution for the ILP.

Note that when calls have equal profits, the problem studied in [AAAE02], they form a special case of restricted profits as defined above.

### 6.2.4 The dual LP

It is interesting to note that the dual of the relaxation of the ILP in Subsection 6.1.1 is rather special: An integer optimal solution to it can be found in polynomial time. The details follow.

Let the calls be referred to by their indices 1, 2, ..., \( m \) and the edges numbered 1, 2, ..., \( n \). Define \( E_i = \{ j : \text{edge } j \text{ is contained in route of call } i \} \).
The dual LP can be stated as:

\[
\begin{align*}
\min & \sum_{i=1}^{m} y_i + \sum_{j=1}^{n} c(j) \cdot z_j \\
\text{subject to} & \\
& y_i + \sum_{j \in E_i} z_j \geq p(i), \ i = 1, 2, \ldots, m \\
& y_i, z_j \geq 0, \ i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, n
\end{align*}
\]

We carry out the, by now familiar, variable substitutions. Namely, \(Z(j) = \sum_{k=1}^{j} z_k, \ j = 1, 2, \ldots, n\). Also, set \(Z(0) = 0\). For a call \(i\), let \(r_i\) be the highest indexed edge in its route such that edge \(r_i + 1\) is not in the route and \(l_i\) the least indexed edge in the route such that edge \(l_i - 1\) is not in the route (where, for \(r_i = n\), we treat \(r_i + 1\) as 1 and for \(l_i = 1\), \(l_i - 1\) as \(n\)). For a call \(i\) whose route does not contain both edges 1 and \(n\), we have:

\[
\sum_{j \in E_i} z_j = \sum_{j = l_i}^{r_i} z_j = Z(r_i) - Z(l_i - 1)
\]

and otherwise:

\[
\sum_{j \in E_i} z_j = \sum_{j = l_i}^{r_i} z_j + \sum_{j = 1}^{n} z_j = Z(r_i) + Z(n) - Z(l_i - 1)
\]

To take care of the variables \(y_i\), we make the variable substitution:

\[
Y(i) = y_i + Z(r_i), \ i = 1, 2, \ldots, m.
\]

The dual LP can now be rewritten with each constraint having at most one +1 and one −1:

\[
\begin{align*}
\min & \sum_{i=1}^{m} (Y(i) - Z(r_i)) + \sum_{j=1}^{n} c(j)(Z(j) - Z(j - 1)) \\
\text{subject to} & \\
& Y(i) - Z(l_i - 1) \geq p(i), \ \text{if call } i \ \text{is not routed through edges } 1 \ \text{and } n \\
& Y(i) - Z(l_i - 1) \geq p(i) - Z(n), \ \text{if call } i \ \text{is routed through edges } 1 \ \text{and } n \\
& Y(i) - Z(r_i) \geq 0, \ i = 1, 2, \ldots, m, \ \text{corresponding to non-negativity of } y_i \\
& Z(j) - Z(j - 1) \geq 0, \ j = 1, 2, \ldots, n, \ \text{corresponding to non-negativity of } z_j \\
& Z(0) = 0
\end{align*}
\]

A feasible solution to the dual is obtained by letting \(Y(i) = p(i), \ i = 1, 2, \ldots, m\) and \(Z(j) = 0, \ j = 1, 2, \ldots, n\). Thus, the objective value of the dual LP is at most \(P = \sum_{i=1}^{m} p(i)\). But, in an optimal solution,

\[
Z(n) = \sum_{j=1}^{n} (Z(j) - Z(j - 1)) \\
\leq \sum_{j=1}^{n} c(j)(Z(j) - Z(j - 1)), \ c(j) \geq 1 \\
\leq \sum_{i=1}^{m} (Y(i) - Z(r_i)) + \sum_{j=1}^{n} c(j)(Z(j) - Z(j - 1)) \\
\leq P
\]

Thus, to obtain an integral optimal solution for the dual we perform a
6.3 Algorithms for special cases of a different “kind”

The special cases considered in Section 6.2 restricted the nature of calls, like equal length routes or proper routes et cetera. In this subsection, by contrast, we shall place restrictions on the different numerical parameters that appear in the input, namely the edge capacities, profits and lengths. More concretely, we study instances where either:

(a) The minimum capacity on edges is bounded by a constant, or
(b) The profits are proportional to the length of the route of the calls and the length of the route of the calls is at most \([n/2]\), where \(n\) is the number of edges in the ring, or
(c) The length of the route of the calls is bounded by a constant.

For the first, a polynomial time optimal algorithm is shown. For the second, a polynomial time approximation scheme (PTAS) is derived while for the last, an optimal algorithm and a randomized PTAS (when the edge capacities are sufficiently high) are given. The randomized PTAS for the last case is useful to obtain a fast approximation to the optimal solution.

6.3.1 Minimum edge capacity bounded by a constant

Let the minimum edge capacity in the ring be \(c_{\min}\), a constant. Let \(e_{\min}\) be an edge with capacity \(c_{\min}\). Note that any optimal solution to the PCC problem routes at most \(c_{\min}\) calls through the edge \(e_{\min}\). Let the number of calls with routes through \(e_{\min}\) be \(m_{\min}\) \(\leq m\), where \(m\) is the total number of calls. It is straightforward to see that the number of subsets of calls routed through \(e_{\min}\) each with cardinality at most \(c_{\min}\) is \(O((m_{\min} + 1)^{c_{\min}}) = O((m + 1)^{c_{\min}}) = O(\text{poly}(m))\). For each such subset, by Lemma 6.1.2, we can find in polynomial time a maximum profit feasible call set among the rest of the calls not routed through \(e_{\min}\) and that satisfies the edge capacity.
Chapter 6. Pre-routed Call Control

6.3.2 Calls with profits proportional to length of routes

Here we look at a "natural" profit assignment to a call; viz., the profit is the length of the route of the call (could as well be proportional to it). Further, we consider only routes of length at most \([n/2]\), where \(n\) is the number of edges in the ring.

Identify two edges \(e\) and \(e'\) that have \([n/2]\) vertices between them. Since every route is of length at most \([n/2]\), none of them pass through both \(e\) and \(e'\). Now, compute an optimal fractional solution \(x^*\) to the LP. Partition the set of calls \(S\) as \(S_e\) and \(S_e' = S \setminus S_e\), where \(S_e\) is the set of calls whose routes do not pass through \(e\). The set of routes in \(S_e\) (respectively, \(S_e'\)) do not pass through edge \(e'\) (respectively, \(e\)). Consider the line \(L\) (respectively, \(L'\)) obtained by opening the ring at edge \(e'\) (respectively, \(e\)). For an edge in \(L\) (respectively, \(L'\)) set its capacity to be the rounded up sum of the \(x^*\) values of calls in \(S_e\) (respectively, \(S_e'\)) that pass through it. Now for each of the call control problems on \(L\) and \(L'\) with the calls \(S_e\) and \(S_e'\), respectively, there is an optimal feasible \(\{0, 1\}\)-route function, by Lemma 6.1.2. Thus, the combined profits of the calls accepted in the optimal solutions is at least the optimal objective value of the original LP. However, combining the two optimal solutions could violate the capacity of the edges in the ring by at most 1 (The edge capacities were set to the rounded up sum of the \(x^*\) values).

For a given set of routes, a minimum profit circle cover is a subset of these routes such that for any edge there is at least one route in the subset that passes through it and the sum of the profits of the routes in the subset is the minimum possible. It is easy to see that in a minimum profit circle cover no route is strictly contained in another. Thus, in a minimum profit circle cover no edge has more than 2 routes passing through it. For otherwise, if 3 routes that are not contained in each other pass through an edge then by removing the route which is contained in the union of the other two routes we have a smaller profit circle cover. Computing a minimum profit circle cover for the routes of calls in a ring can be done in polynomial time [ACL95].

A feasible solution can be obtained from the infeasible rounded solution by discarding the routes in a minimum profit circle cover. Since every edge in a minimum profit circle cover has at most 2 routes passing through it, the profit of the minimum profit circle cover is at most \(2n\).

Given an \(\epsilon > 0\), fix the constant \(K = 2\lfloor \frac{1+\epsilon}{\epsilon} \rfloor\). If the objective value of the fractional optimal solution \(p(OPT^*) \leq Kn\), then there exists an edge in the ring through which at most \(K\) paths pass in any optimal integral solution.
It follows from this argument; the profit of a feasible solution can be counted in two ways: (i) sum of the profits of calls in the solution and (ii) total of the number of routes through every edge. If \( p(OPT^*) \leq Kn \) then \( p(OPT) \leq Kn \), where \( OPT \) is an optimal solution to the PCC problem. \( p(OPT) \) is the sum of the number of routes through every edge in the optimal solution. There are \( n \) edges, thus the number of routes through one of the edges should be at most \( K \). Thus, we can find an optimal solution to the instance by assigning for each edge in turn an edge capacity equal to \( K \). By the previous special case, an optimal solution can be found in polynomial time when the minimum edge capacity is bounded by a constant.

On the other hand, let \( p(OPT^*) > Kn \). We have \( p(OPT^*) > 2^{\lceil \frac{1}{\epsilon} \rceil} n \). Therefore, the feasible solution computed by throwing out a minimum profit circle cover from a rounded solution has a profit at least \( p(OPT^*) - 2n \geq p(OPT^*) - \frac{1}{1+\epsilon} \cdot p(OPT^*) \). Or, the profit of the feasible solution is at least \( \frac{p(OPT^*)}{1+\epsilon} \). But \( p(OPT^*) \) is an upper bound on the profit of any feasible solution. Thus, we have a PTAS for this special instance.

### 6.3.3 Calls with bounded route lengths

Let the maximum length of a route of a call be \( L \), a constant. Since we have shown a polynomial time optimal algorithm for bounded minimum edge capacity, we shall assume that the minimum edge capacity is greater than a given constant.

There is a dynamic programming (DP) approach that solves this instance optimally. However, the running time is exponential in \( L \). The dynamic program works as follows: Fix an edge \( e \) in the ring. Put the route of a call which contains \( e \), has length \( i \) and contains exactly \( r \) consecutive edges that lie clockwise to \( e \), into the set \( S_{i,r}, i = 1, 2, ..., L, r = 0, 1, 2, ..., i - 1 \). There are \( O(L^2) \) such sets. All these sets have a cardinality of at most \( m \), the total number of calls. The dynamic program is to try all possible at most \( m \) values for each set, \( S_{i,r} \). In each set, we naturally pick the calls in decreasing order of their profits. For the rest of the paths that do not pass through \( e \) we can apply the result of Lemma 6.1.2 with the edge capacities reduced by the appropriate number of paths. Every entry of the DP table is the profit that is achieved for a particular way of fixing values for \( S_{i,r} \). Finally, we pick that entry of the DP table that has the maximum profit. The running time is \( O(m^2L^2) \).

If the edge capacities are sufficiently high then we can get a good, fast approximation to an integral optimal solution from the fractional optimal solution. The details are presented below.
For a call \( z \) and an edge \( e \) through which its route passes, let \( S_{i,e} \) denote the set of calls with indices less than \( i \) and whose routes contain \( e \). Let \( x^* = (x_1^*, x_2^*, ..., x_m^*) \) be an optimal fractional solution to the relaxation of the ILP of Subsection 6.1.1. Let \( OPT \) and \( OPT^* \) denote the objective value of an optimal integral and fractional solution respectively. An adaptation of a randomized rounding method described in [CCKR02] gives a polynomial time randomized approximation scheme.

Define \( f(\delta) = \sqrt{1.65\delta \ln(1/\delta)} \). Given \( 1/3 > \epsilon' > 0 \), let \( \delta' \) be such that: \( \epsilon' = f(\delta') \) and \( \delta' \leq 1/e \). We shall need the following numerical details for the analysis, \( f(0.00288) = 0.1667 \). Also, for \( 0 < \delta \leq 0.00288 \), \( f(\delta) > 57\delta \) and \( f(\delta) > \delta^{0.68} \). See Figure 6.1. Now, we show that \( \min\{f(\frac{\delta'}{2L^2}), f((\frac{1}{2L^2})^2)\} \leq (1/L)f(\delta') \).

Case A: \( 2L^2 \leq \frac{1}{\delta'} \)

We have \( \ln(2L^2) \leq \ln \frac{1}{\delta'} \). Therefore,

\[
\begin{align*}
  f(\frac{\delta'}{2L^2}) &= \sqrt{1.65(\delta'/2L^2)\ln(2L^2/\delta')} \\
  &= (1/L)\sqrt{1.65(\delta'/2)(\ln(2L^2) + \ln(1/\delta'))}
\end{align*}
\]
98 6.3. Algorithms for special cases of a different “kind”

\[ \leq (1/L) \sqrt{1.65( \delta'/2)(2 \cdot \ln(1/\delta'))} = (1/L)f(\delta') \]

Case B: \( 2L^2 \geq \frac{1}{\delta'} \) or \( 1/(2L^2) \leq \delta' \)

Since \( \delta' < 1/e \) and \( f(.) \) is increasing in \((0, 1/e)\), \( f(1/(2L^2)) \leq f(\delta') \).

Now,

\[ f((1/(2L^2))(1/(2L^2))) = \sqrt{1.65(1/(2L^2))^2 \ln((2L^2)^2)} \]
\[ = (1/L) \sqrt{1.65(1/(4L^2))(2 \ln(2L^2))} \]
\[ = (1/L)f(1/(2L^2)) \]
\[ \leq (1/L)f(\delta') \]

Let \( \delta'' = \min\{\delta'/2L^2, 1/(4L^4)\} \). Since \( L \) and \( \delta' \) are constants so is \( \delta'' \). If the minimum edge capacity is bounded by the constant \( 1/\delta'' \) then we have a polynomial time optimal algorithm. Hence, we may assume the minimum edge capacity \( c_{\min} \) is greater than \( 1/\delta'' \).

\[ \text{implying, } c(e)\delta'' > 1, \forall e \]

Let \( \epsilon = \frac{\delta'}{L} \). Let \( \delta \) be such that \( \epsilon = f(\delta) \) and \( \delta \leq \delta' \). Clearly, \( \delta \geq \delta'' \), since \( f(\delta) = \epsilon = \epsilon'/L \geq f(\delta'') \). Therefore, \( c_{\min} \delta \geq c_{\min} \delta'' > 1 \). Now, we do the following randomized rounding. Set variable \( x_i \) corresponding to call \( i \) to 1 with probability \((1 - \epsilon)x_i^* \). Call these random variables \( Y_i, i = 1, 2, ..., m \).

Consider the following feasible solution to the PCC problem:

\[ Z_i = \begin{cases} 1 & \text{if } Y_i = 1 \text{ and } \sum_{j \in S_{i,e}} Z_j \leq c(e) - 1, \forall e \in E_i \\ 0 & \text{otherwise} \end{cases} \]

We state without proof the following lemma in [CCKR02]:

**Lemma 6.3.1** Let \( X_1, X_2, ..., X_m \) be independent random variables and let \( 0 \leq \beta_1, \beta_2, ..., \beta_m \leq 1 \) be reals, where for \( i \in \{1, 2, ..., m\} \), \( X_i = \beta_i \) with probability \( p_i \), and \( X_i = 0 \) otherwise. Let \( X = \sum_i X_i \) and \( \mu = E[X] \). Then

(i) \( \sigma(X) \leq \sqrt{\mu} \).

(ii) For any \( \lambda \) with \( 0 < \lambda < \sqrt{\mu} \), \( Pr[X > \mu + \lambda \sqrt{\mu}] < \exp(-\frac{\lambda^2}{2}(1 - \lambda/\sqrt{\mu})) \).

Now, we prove the following lemma:

**Lemma 6.3.2** Given \( 1/3 > \epsilon > 0 \) and \( L \geq 2 \), let \( \epsilon = \epsilon'/L \) \((< 1/6)\) and \( \delta \) be such that \( \epsilon = f(\delta) \) and \( \delta < 0.00288 \). Then, \( Pr[Z_i = 1] \geq (1 - 1.5\epsilon')x_i^* \).

**Proof:** Consider an edge \( e \in E_i \) of a call \( i \). Let \( \pi_{i,e} = \{Z_i = 0\} \) because \( e \) was saturated by routes of calls in \( S_{i,e}|Y_i = 1\). That is, \( \pi_{i,e} \) denotes the event that call \( i \) was not accepted because edge \( e \) has already \( c(e) \) calls in \( S_{i,e} \) routed through it.
Pr[π_{i,e}] = Pr[\sum_{j \in S_{i,e}} Z_j = c(e) > c(e) - 1]

But Z_j \leq Y_j, for all j. Therefore,

Pr[π_{i,e}] \leq Pr[\sum_{j \in S_{i,e}} Y_j > c(e) - 1]

Equivalently,

Pr[π_{i,e}] \leq Pr[\sum_{j \in S_{i,e}} \frac{Y_j}{\delta c(e)} > \frac{c(e)-1}{\delta}] \leq Pr[\sum_{j \in S_{i,e}} \frac{Y_j}{\delta c(e)} > \frac{1-\delta}{\delta}]

The random variables \{Y_j/\delta c(e)\}_{j \in S_{i,e}} satisfy the conditions of Lemma 6.3.1. Let \( Y = \sum_{j \in S_{i,e}} \frac{Y_j}{\delta c(e)} \) and \( \mu = E[Y] \). By linearity of expectation,

\[
\mu = \sum_{j \in S_{i,e}} \frac{1}{\delta c(e)} \cdot (1 - c) = \frac{1-\delta}{\delta} \sum_{j \in S_{i,e}} \frac{x_j}{\delta c(e)} \leq \frac{1-\delta}{\delta}
\]

The last inequality follows from the capacity constraint on edge e. Case I: \( \mu < (7/8)(1 - \delta)/\delta \). Since \( \frac{\sigma(Y)}{\sqrt{\mu}} \leq 1 \) by Lemma 6.3.1,

\[
Pr[π_{i,e}] \leq Pr[Y > \frac{1-\delta}{\delta}] \leq Pr[|Y - \mu| > \frac{1}{8} \cdot \frac{1-\delta}{\delta} \cdot \frac{\sigma(Y)}{\sqrt{\mu}}] \leq \frac{64\delta^2 \mu}{(1-\delta)^2}
\]

The last inequality follows from Chebyshev's inequality. Given that \( \delta < 0.00288 \), we obtain, \( Pr[π_{i,e}] < 57\delta < \epsilon \).

Case II: \( \mu \geq (7/8)(1 - \delta)/\delta \). Set \( \lambda \) such that \( \mu + \lambda \sqrt{\mu} = (1 - \delta)/\delta \). We now have,

\[
\lambda = \frac{1-\delta}{\sqrt{\mu}} \geq \frac{1-\delta}{\sqrt{\frac{1-\delta}{\delta}}} = \frac{\delta^2}{\sqrt{\delta(1-\delta)}} \geq \frac{656}{57} \frac{\epsilon}{\sqrt{\delta}}
\]

Further, \( 1 - \lambda/\sqrt{\mu} = 2 - (1 - \delta)/(\delta \mu) \geq 6/7 \). By Lemma 6.3.1 on random variables \{Y_j/\delta c(e)\}_{j \in S_{i,e}}, we get,

\[
Pr[π_{i,e}] \leq Pr[Y > \mu + \lambda \sqrt{\mu}] \leq \exp(-\frac{\lambda^2}{2}(1 - \lambda/\sqrt{\mu})) < \exp(-\frac{1}{2} \frac{56}{57} \frac{\epsilon^2}{\delta}) \leq \delta^{0.68}, \text{ substituting } \epsilon = \sqrt{1.65 \delta \log(1/\delta)} \leq \epsilon
\]

Therefore, \( Pr[π_{i,e}] < \epsilon \). Now,

\[
Pr[Z_i = 0|Y_i = 1] = Pr[π_{i,e} \text{ for at least one } e \in E_i] \leq \sum_{e \in E_i} Pr[π_{i,e}] < l_i \epsilon, \text{ } l_i \text{ is length of route of call } i \leq L \epsilon, \text{ since } l_i \leq L
\]
Finally, we obtain,

$$\Pr[Z_i = 1] = (1 - \Pr[Z_i = 0|Y_i = 1]) \cdot \Pr[Y_i = 1]$$

$$\geq (1 - Le)(1 - e)x_i^*$$

$$= (1 - e')(1 - e'/L)x_i^*$$

$$\geq (1 - (1 + 1/L)e'x_i^*$$

$$\geq (1 - 1.5e')x_i^*, \quad L \geq 2$$

\[ \square \]

**Theorem 6.3.3** Given $e' > 0$ and an off-line, uniform PCC instance on rings with the length of routes of calls bounded by a constant there is a randomized rounding achieving an approximation ratio of $\frac{1}{1 - 3e'}$ with probability at least $1/2$.

**Proof:** If the minimum capacity of the edges is bounded by a constant there is a polynomial time optimal algorithm to solve the PCC problem. Assume the minimum capacity is not bounded by a constant. If $e' \geq 1/3$ there is nothing to prove. Otherwise, the previous analysis goes through. Let $P$ denote the profit. The expected profit of the randomized rounding is $E[P] = \sum_{i=1}^{m} p(i) \cdot \Pr[Z_i = 1]$. This is at least $(1 - 1.5e')OPT \geq (1 - 1.5e')OPT$. But $P$ never exceeds $OPT$. Therefore, we have $\Pr[P \geq (1 - 3e')OPT] \geq 1/2$. Hence, the randomized algorithm achieves the desired ratio with a probability of at least $1/2$.

\[ \square \]
Off-line, non-uniform call control problems on lines are tackled in this chapter. For several special cases, constant or logarithmic factor approximation algorithms or polynomial time approximation schemes are presented.

So far, we have considered call control problems when all the calls in the input call set have uniform bandwidth requirements. In this chapter, we shall allow the calls to have varying bandwidth requirements, i.e. non-uniform call control problems. The KNAPSACK problem, which is the off-line, non-uniform call control problem when the network is restricted to be just an edge, is a weakly \( \mathcal{NP} \)-hard problem. Even if the topology is restricted to that of lines, for the off-line, non-uniform call control problem approximation algorithms with constant ratios are not known. With the so-called bottleneck assumption, where the maximum bandwidth requirement is at most the minimum edge capacity, constant factor approximation algorithms are shown by Chekuri and Khanna [CK03] for off-line PCC problems on lines, rings and trees.

The purpose of this chapter is to analyze call control problems on the line without the bottleneck assumption. We identify other kinds of restrictions for which it is possible to obtain good approximation algorithms. We refer to the call control problems on lines as PCC problems as the routes for calls are fixed as soon as the end vertices are known. In Section 7.1, we consider lines with a constant number of vertices and show an FPTAS for the off-line PCC problem. In Section 7.2, we consider calls with bounded route lengths and show a PTAS for the same problem. In Section 7.3, we show constant or logarithmic ratio approximation algorithms for off-line PCC problems when the routes of calls have equal lengths.

The off-line PCC problems on lines are specified by giving an edge capacitated line \( L = (V,E,c) \), a set of calls \( S = \{r_1, r_2, ..., r_m\} \) and a profit function \( p : S \to \mathbb{N} \). A call \( r_i \) is specified by giving its route \( s_i \) (the end vertices of a call already specify its route) and its bandwidth requirement \( b_i \), ignoring all the other parameters as they are common for all the calls. Let \( P_{\min} \) (\( P_{\max} \)) and \( B_{\min} \) (\( B_{\max} \)) denote the minimum (maximum) profit and
7.1 PCC on lines with constant number of vertices

Consider an instance of PCC where the number of vertices in the line is a constant, say $|V| = K$. Let $V = \{1, 2, ..., K\}$. The number of different possible pairs of vertices is $l = K \cdot (K - 1)/2 = O(K^2)$, a constant. Thus, in any call set there can be at most $l$ different types of calls in it (classifying calls based on their end vertices). Given any $\epsilon > 0$, we shall show a polynomial (in $m, 1/\epsilon$) time algorithm that achieves an approximation ratio of $1 + \epsilon$ for the PCC problem, i.e. an FPTAS for the problem.

The algorithm is based on a dynamic programming (DP) approach. Let the calls be grouped into types based on their end vertices. Consider the $(i, j)$-th type calls, $S_{(i,j)}, 1 \leq i < j \leq K$ (These calls have $i$ and $j$ as their end vertices). Let there be $n_{(i,j)}$ calls in $S_{(i,j)}$. Order their routes arbitrarily, say $s_{(i,j)}^{k}, k = 1, 2, ..., n_{(i,j)}$. Scale the profits by dividing all of them with $\frac{P_{\text{max}}}{(1+\epsilon)m}$ and thereafter round them down. Denote the scaled and rounded profit of call $r_l$ as $p_{\#}(r_l) = \left\lfloor \frac{(1+\epsilon)p(r_l) - m}{P_{\text{max}}} \right\rfloor$. Now, we build a DP table where an entry $B_{(i,j)}(k, P_{\#})$ is a subset of the first $k$ calls in $S_{(i,j)}$ which has the minimum possible bandwidth and achieves a scaled profit of $P_{\#}$. Let $b(B_{(i,j)}(k, P_{\#}))$ represent the sum of the bandwidth requirements of calls in $B_{(i,j)}(k, P_{\#})$. The relations between the entries in the table are given by,

$$B_{(i,j)}(k, P_{\#}) = \arg \min \{ b(B_{(i,j)}(k - 1, P_{\#})), b(B_{(i,j)}(k - 1, P_{\#} - p_{\#}(r_{(i,j)}^{k}))) \cup \{r_{(i,j)}^{k}\} \}$$

The initial conditions being $B_{(i,j)}(0, 0) = \emptyset$ and $B_{(i,j)}(0, P) = \perp$, for any $P > 0$. A $\perp$ indicates that it is infeasible to achieve the profit with the set of calls in the corresponding entry. We let $b(\emptyset) = 0$ and $b(\perp) = \infty$. Further, define $\perp \cup A = \perp$, for any set $A$. The recurrence relation is easily explained; the set on the left hand side either contains the call $r_{(i,j)}^{k}$ or not. In the latter case, the argument of the first term in the right hand side is the required set. Otherwise, the second term on the right hand side has as its argument the singleton set $\{r_{(i,j)}^{k}\}$ in union with the optimal set among the remaining calls with the appropriate profit.

For each of the different types of calls this is similar to the DP approach to solving a KNAPSACK problem. Clearly, the maximum possible scaled profit for any solution is $\frac{(1+\epsilon)m^2}{\epsilon}$. Thus, for each of the $l$ different types we have a
Chapter 7. Call Control on Lines 103

Next, we combine the entries in the \( l \) different tables to obtain a \((1 + \epsilon)\)-approximation. We do so by asking the following question: Does there exist a feasible subset of \( S \) whose overall scaled profit is \( P^\# = \sum_{1 \leq i < j \leq K} P_{(i,j)}^\# \) such that the sum of the scaled profits of the \((i, j)\)-th type of calls in it is \( P_{(i,j)}^\# \)? A crucial observation in answering this question is that the number of ways of splitting a fixed profit \( P \) to be achieved into \( l \) positive integers is \( O((1 + e)^l) \). Thus, there are a polynomial number of ways of splitting any of the (polynomial number of) achievable scaled profits. Fix a particular integer profit \( P, 0 \leq P \leq \frac{(1 + e)^2}{\epsilon} \) and \( P_{(i,j)}, 1 \leq i < j \leq K \) with \( P = \sum_{1 \leq i < j \leq K} P_{(i,j)} \). Now, define

\[
B(P, P_{(1,2)}, P_{(1,3)}, \ldots, P_{(K-1,K)}) =
\begin{cases}
\bigcup_{1 \leq i < j \leq K} B_{(i,j)}(n_{(i,j)}, P_{(i,j)}) & \text{if } B_{(i,j)}(n_{(i,j)}, P_{(i,j)}) \neq \perp, \\
\perp & \text{otherwise}
\end{cases}
\]

Compute the sets for each possible overall scaled profit with each of the polynomial number of ways of splitting it. Output the set corresponding to the largest possible scaled profit that is not \( \perp \). The profit of the calls in this feasible solution is at least a \((1 + e)\)-fraction of that of an optimal solution. Indeed, consider an optimal solution \( OPT \subseteq S \) to the PCC problem. It can be written as the union of sets, \( OPT_{(i,j)} \), of calls in \( OPT \) of type \((i, j)\). By construction of the DP tables for each of the types, \( b(B_{(i,j)}(n_{(i,j)}, P^\#(OPT_{(i,j)}))) \leq b(OPT_{(i,j)}), \forall (i, j) \). Or stated differently, \( B(p^\#(OPT), p^\#(OPT_{(1,2)}), p^\#(OPT_{(1,3)}), \ldots, p^\#(OPT_{(K-1,K)})) \) is feasible. Thus the profit of the feasible solution output is at least

\[
p\left(\bigcup_{1 \leq i < j \leq K} B_{(i,j)}(n_{(i,j)}, P^\#(OPT_{(i,j)}))\right)
= \sum_{1 \leq i < j \leq K} p(B(n_{(i,j)}, P^\#(OPT_{(i,j)})))
\geq \sum_{1 \leq i < j \leq K} \frac{P_{\text{max}}}{(1 + e)m} P^\#(OPT_{(i,j)})
= \frac{P_{\text{max}}}{(1 + e)m} P^\#(OPT)
\geq \frac{P_{\text{max}}}{(1 + e)m} \sum_{r \in OPT} \left(\frac{(1 + e)m}{P_{\text{max}}} p(r) - 1\right)
= p(OPT) - \frac{P_{\text{max}}}{(1 + e)m} |OPT|
\geq p(OPT) - \frac{1}{1 + e} p(OPT), \quad p(OPT) \geq P_{\text{max}}, \ m \geq |OPT|
= \frac{P_{\text{max}}}{(1 + e)}
\]

The first inequality holds because for a call \( r, p(r) \geq \frac{P_{\text{max}}}{(1 + e)m} p^\#(r) \) and the scaled profit of the term in the summation is exactly \( p^\#(OPT_{(i,j)}) \). \( p(OPT) \geq P_{\text{max}} \) because any individual call is feasible for the line. The total running time of the FPTAS is the number of DP table entries that are computed (assuming constant work for looking up the two earlier
entries in the table). For each of the \( l = O(K^2) \) types, the number of entries is \( O((1 + e)m^3/e) \). The number of entries for each of the different possible overall scaled profits is \( O(((1 + e)m^2/e + 1)^{O(K^2)}) \). Thus, the running time of the algorithm is \( O(((1 + e)K^2m^3/e + ((1 + e)m^2/e + 1)((1 + e)m^2/e + 1)^{O(K^2)}) \). The last term in the above expression dominates and we can write the running time to be \( O(((1 + e)m^2/e + 1)^{O(K^2)}) \).

**Theorem 7.1.1** Given an \( \epsilon > 0 \) and an instance of off-line PCC on lines with constant number of vertices, there is an algorithm that achieves an approximation ratio of \( 1 + \epsilon \) and runs in time polynomial in the number of calls in the instance and \( 1/\epsilon \).

### 7.2 PCC on lines when calls have bounded length routes

Consider instances of PCC on lines where the input set of calls is such that the lengths of the routes of calls in it are bounded by a constant, say \( L \). We use the result of Section 7.1 and the idea of “shifting” to obtain a PTAS for such instances. Given \( \epsilon > 0 \), let \( K = \lceil \frac{1+\epsilon}{\epsilon} \rceil \), a constant. We construct \( K \) different instances of PCC on lines obtained from the original one. For each of these \( K \) different instances we compute a \( (1 + \epsilon) \)-approximation and then output the solution which is the largest. Let us call the \( K \) different instances \( A(1), A(2), \ldots, A(K) \). \( A(i) \) is obtained by cutting the input line at vertices numbered \( j \equiv iL \mod KL \) and discarding those calls whose routes pass through the vertices that are cut, i.e. use both edges incident on the vertices. Note that a call that is discarded in one of the instances is not discarded in any other instance. Now, any instance \( A(i) \) can be treated as smaller PCC problems on lines of length at most \( KL \). We can, therefore, use the FPTAS from the previous section to obtain a \( (1 + \epsilon) \)-approximation for them. For instance \( A(i) \), the algorithm that outputs all the calls accepted in the FPTASes for the smaller PCC problems is a \( (1 + \epsilon) \)-approximation. The algorithm for the original instance is to compute the \( (1 + \epsilon) \)-approximation for each of the instances \( A(i) \) and then to output the solution of that instance which has the maximum profit. To see that the best of these \( K \) different solutions is a \( (1 + O(\epsilon)) \)-approximation for the original instance, consider an optimal solution \( OPT \) of it. We can group the calls in \( OPT \) that were discarded in the \( K \) constructed instances into \( K \) different types. Group calls that were discarded in instance \( A(i), i = 1, 2, \ldots, K \) and call them type \( i \) calls. Then, the profit of at least one of these types in \( OPT \), say type \( j \), is at most \( 1/K(\leq \frac{1}{1+\epsilon}) \) fraction of \( p(OPT) \). Denote this profit by \( p(OPT_j) \). Note that
this type of calls is discarded in the instance $A(j)$. Now, the profit of calls computed for the instance $A(j)$ by the algorithm can be lower bounded as follows. The set $OPT \setminus OPT_j$ is feasible for instance $A(j)$. Thus (we use $S(j)$ to denote the set of calls in the solution computed by the algorithm for the instance $A(j)$),

$$p(S(j)) \geq \frac{p(OPT) - p(OPT_j)}{1 + \epsilon} \geq \frac{p(OPT) - \frac{3p(OPT)}{2(1 + \epsilon)}}{1 + \epsilon} \geq \frac{p(OPT)}{1 + 3\epsilon}, \text{ for } \epsilon < 1, \epsilon^2 \leq \epsilon$$

If $\epsilon > 1$ then $K = 2$. Now, $p(OPT_j) \leq p(OPT)/2$. Thus, $p(S(j)) \geq \frac{p(OPT)}{4(1 + \epsilon)}$. For $\epsilon > 1$, this is a $(1 + O(\epsilon))$-approximation.

For each of the $K$ instances we run the FPTAS of the previous section on $\left\lceil \frac{n}{K} \right\rceil$ pieces of the input line, where $n$ is the number of vertices on the line. The running time for computing a solution for each of the instances is thus, $O\left(\frac{n}{K}((1 + \epsilon)m^2/\epsilon + 1)O((1+\epsilon)L/\epsilon^2)\right)$. There are $K = \left\lceil \frac{1+\epsilon}{\epsilon} \right\rceil$ instances for which solutions have to be computed. We get a total running time of $O\left(\frac{n}{K}((1 + \epsilon)m^2/\epsilon + 1)O((1+\epsilon)L/\epsilon^2)\right)$. Since the running time is exponential in $1/\epsilon$, we obtain a PTAS instead of an FPTAS.

**Theorem 7.2.1** Given an $\epsilon > 0$ and an instance of PCC on lines such that the length of the routes of calls is bounded by a constant, there is an algorithm that achieves an approximation ratio of $1 + O(\epsilon)$ and runs in polynomial time.

### 7.3 PCC on lines when calls have equal length routes

Now, we derive approximation algorithms for instances of PCC on lines when the input call set have equal length routes. The arguments that we employ in this section can be seen to hold also for instances where the routes of calls are not strictly contained in each other.

#### 7.3.1 $\min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\}$ is a constant

Let the minimum between $\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}$ be $\frac{P_{\text{max}}}{P_{\text{min}}}$ and equal to $M$, a constant. The other case can be handled using arguments similar to the ones that follow.

The algorithm itself is simple to describe. For every call compute the ratio of its profit to its bandwidth requirement. We accept calls greedily in
106 7.3. PCC on lines when calls have equal length routes

this order. More formally, let the calls in S be arranged in decreasing order of their computed ratios as \{r_1, r_2, ..., r_m\}. Initially, \(S'_0 = \emptyset, i = 1\) and all edges have capacities \(c_0(e) = c(e), \forall e \in E\), as given in the input. Consider call i. If the route of i contains an edge e such that \(c_{i-1}(e) < b_i\), reject it. Otherwise accept the call i. If call i is accepted then set \(S'_i = S_{i-1}' \cup \{r_i\}\) and \(c_i(e) = c_{i-1}(e) - b_i\), for all edges e in the route of call i and \(c_i(e) = c_{i-1}(e)\) for all edges e not in the route of call i. If call i is rejected then set \(S'_i = S_{i-1}'\) and \(c_i(e) = c_{i-1}(e)\), for all edges e \(\in E\). If i is not already \(m\), increment i and repeat.

Let \(S'\) be the set of calls accepted by the algorithm. We shall show the sum of profits of calls in \(S'\) is at least a constant fraction of the sum of profits of calls in an optimal solution. Let \(OPT\) represent an optimal feasible solution for the given instance. Denote by \(OPT'_0, S''\) the set of calls \(OPT \setminus S'\), \(S' \setminus OPT\). Arrange the calls in the set \(S''\) in decreasing order of the ratio of their profits to bandwidth requirements. Say \(S'' = \{r_1', r_2', ..., r_i'\}\). Initially i = 1. Let \(OPT''_{(i-1, \text{left})}\) be the set of calls in \(OPT_{i-1}'\) that intersect \(r_i'\) such that if \(r \in OPT''_{(i-1, \text{left})}\) then the right end vertex of \(r\) lies to the left of or coincides with the right end vertex of \(r_i'\). Let \(OPT''_{(i-1, \text{right})}\) be the set of calls in \(OPT_{i-1}'\) that intersect \(r_i'\) such that if \(r \in OPT''_{(i-1, \text{right})}\) then the left end vertex of \(r\) lies to the right of the left end vertex of \(r_i'\). Let the calls in \(OPT''_{(i-1, \text{left})}\) \((OPT''_{(i-1, \text{right})}\) be arranged in decreasing (increasing) order of their left (right) end vertices. From each of these sets select calls in that order till the cumulative bandwidth requirements of these calls just exceed the bandwidth requirement \(b_{r_i'}\) of call \(r_i'\) or the set is exhausted. Let \(OPT''_{(i-1, \text{left})}\) and \(OPT''_{(i-1, \text{right})}\) denote the sets so picked. Set \(OPT''_i = OPT''_{i-1} \setminus (OPT''_{(i-1, \text{left})} \cup OPT''_{(i-1, \text{right})})\). Increment i and repeat the above steps till i is greater than \(l\).

Claim 7.3.1 For i = 1, 2, ..., l, if \(r \in OPT_{i-1}'\) then \(\frac{p(r)}{b_r} \leq \frac{p(r_i')}{b_{r_i'}}\). Further, \(OPT''_l = \emptyset\).

Proof: Fix some i \(\in \{1, 2, ..., l\}\). Let there exist a call \(r \in OPT_{i-1}'\) such that \(\frac{p(r)}{b_r} > \frac{p(r_i')}{b_{r_i'}}\). From the order in which the calls are considered by the algorithm for selection it is clear that \(r\) would have been considered before \(r_i'\). Since it was not picked it must be the case that at least one of the edges through which its route passes would have been violated if it were picked. \(r\)'s route must therefore be incident on at least one of the calls in \(\{r_1', r_2', ..., r_{i-1}'\}\). For \(i = 1\), the set \(\{r_1', r_2', ..., r_{i-1}'\}\) is empty and therefore there can be no such \(r\) as supposed. Assume \(i > 1\) and consider an edge e on the line that would have been violated when \(r\) was considered for selection. Since \(r \in OPT_{i-1}'\)
it was not removed by any of the calls in \( \{r_1', r_2', ..., r_{l-1}'\} \) with whose routes it intersected. In particular, all calls in \( \{r_1', r_2', ..., r_{l-1}'\} \) whose routes passed through \( e \) did not remove it from their \( OPT' \) sets. Let these calls be denoted by the set \( S'_{l,e,r} \). Now, \( S'_{l,e,r} \neq \emptyset \). Otherwise, \( r \) could have intersected at the edge \( e \) only with calls in \( OPT \cap S' \) that were picked before it. Note that \( r \) is feasible together with other calls in \( OPT \) and therefore also feasible together with calls in \( OPT \cap S' \). Therefore, \( S'_{l,e,r} \neq \emptyset \). But then, the calls in \( S'_{l,e,r} \) must have removed calls (from one of their \( OPT' \) sets) whose routes contain the edge \( e \). Here, we use the fact that all calls have equal lengths. The total bandwidth requirements of all these calls that are removed is at least the total bandwidth requirements of calls in \( S'_{l,e,r} \). We conclude, therefore, that the sum of the bandwidth requirements of calls in \( OPT \) that pass through \( e \) exceeds the capacity of edge \( e \). But \( OPT \) is a feasible set. A contradiction. Thus, the first part of the claim holds.

For the second part of the claim, assume that \( OPT_i \) is non-empty and call \( r \) belongs to it. Arguing exactly as in the previous paragraph we can again show that \( OPT \) would then be infeasible.

\[ \square \]

**Claim 7.3.2** For \( i = 1, 2, ..., l \) the sum of the profits of calls in \( OPT''_{(i-1, \text{left})} \) or \( OPT''_{(i-1, \text{right})} \) is at most \( (1 + M)p(r_i') \).

**Proof:** The proof is the same for both of these sets. We show the claim for \( OPT''_{(i-1, \text{left})} \). Let \( r^* \) be the last call that was picked in \( OPT''_{(i-1, \text{left})} \). The sum of the bandwidth requirements of the calls other than \( r^* \) is less than \( b_{r_i'} \).

Now, the sum of profits of calls in \( OPT''_{(i-1, \text{left})} \) can be bounded as follows:

\[
p(OPT''_{(i-1, \text{left})}) = \sum_{r \in OPT''_{(i-1, \text{left})}} p(r) = \sum_{r \in OPT''_{(i-1, \text{left})}: \forall r' \neq r} p(r) + P_{\text{max}} \leq \sum_{r \in OPT''_{(i-1, \text{left})}: \forall r \neq r^*} p(r) b_{r_i} + P_{\text{max}}\cdot b_{r_i} + M p(r_i') \leq \sum_{r \in OPT''_{(i-1, \text{left})}: \forall r \neq r^*} p(r) b_{r_i} + M p(r_i') = (1 + M)p(r_i')
\]

The first inequality follows from \( P_{\text{max}} \) being the maximum profit. The second inequality follows from \( P_{\text{max}} = M \cdot P_{\text{min}} \). The third inequality follows from \( \frac{p(r)}{b_r} \leq \frac{p(r_i')}{b_i} \), \( \forall r \in OPT''_{(i-1, \text{left})} \) by Claim 7.3.1. The last inequality follows from the above observation that the sum of the bandwidth requirements of the calls other than \( r^* \) is less than \( b_{r_i} \).

\[ \square \]
When the minimum between $\frac{B_{\text{max}}}{B_{\text{min}}}$ and $\frac{P_{\text{max}}}{P_{\text{min}}}$ is $\frac{B_{\text{max}}}{B_{\text{min}}}$, the bandwidth requirement of the call $r^*$ is at most $Mb(r'_i)$. Now, the bound of $(1 + M)p(r'_i)$ follows from $\frac{p(r^*)}{b_{r^*}} \leq \frac{p(r'_i)}{b_{r'_i}}$ implying $p(r^*) \leq Mp(r'_i)$.

**Theorem 7.3.3** Let an instance of off-line PCC on lines with the input call set having equal length routes be given. If $\min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\}$ is a constant there is an algorithm that achieves a constant approximation ratio for the problem.

**Proof:** Associate the calls in $OPT' \setminus OPT''$ to the call $r'_i \in S''$, $i = 1, 2, ..., l$. By Claim 7.3.2 and the observation made after it, the total sum of the profits of calls in these two sets is at most $2(1 + M)p(r'_i)$, where $M = \min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\}$. Note that any call in $OPT' = OPT \setminus S'$ has now been associated with exactly one call in $S' \setminus OPT$ as there are no more calls left in $OPT''$. For every call in $OPT \cap S'$ associate the call to itself. Therefore, the sum of the profits of calls in the optimal set is bounded by $2(1 + \min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\})p(S' \setminus OPT) + p(OPT \cap S') \leq 2(1 + \min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\})p(S')$. Hence, the algorithm we have described above achieves an approximation ratio of $2(1 + \min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\})$, a constant.

---

**7.3.2 \( \min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\} \) is arbitrary**

We can achieve an approximation ratio that is logarithmic in $\min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\}$. The method is a standard one. We partition the input call set such that a fixed one of the two ratios is a constant in all sets of the partition. The number of sets in the partition will be $\log(\min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\}) + 1$. Assume that the minimum among the two ratios is $\frac{B_{\text{max}}}{B_{\text{min}}}$. A call with bandwidth $b$ is in the $i$th partition $S_i$ if $2^{i-1}B_{\text{min}} \leq b < 2^iB_{\text{min}}$, $i = 1, 2, ..., \log(\frac{B_{\text{max}}}{B_{\text{min}}}) + 1$. For each call set of the partition we run the algorithm of the previous subsection on the input line to obtain an approximation ratio of $6(= 2 \cdot (1 + 2))$. In an optimal solution to the original problem there is one set in the partition whose contribution to the total profit is at least a $(\log(\frac{B_{\text{max}}}{B_{\text{min}}}) + 1)$-fraction of it. Thus, we get a $6(\log(\frac{B_{\text{max}}}{B_{\text{min}}}) + 1)$-approximation by choosing the output of the algorithm on that set of the partition on which the output set has the maximum profit. If we were to partition the sets based on their profits geometrically as before, we would get an approximation ratio that is logarithmic in $\frac{P_{\text{max}}}{P_{\text{min}}}$. Now, the following theorem is straightforward.

**Theorem 7.3.4** Let an instance of off-line PCC on lines with the input call set having equal length routes be given. There is an algorithm that achieves an approximation ratio of $O(\log(\min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\}))$ for the problem.
When $\frac{B_{\text{max}}}{B_{\text{min}}}$ is bounded, Chakrabarti et al. [CCGK02] show a constant factor algorithm for the general PCC problem on lines. This algorithm is based on a combination of rounding an optimal fractional solution of an LP and a dynamic programming approach. When $\frac{B_{\text{max}}}{B_{\text{min}}}$ is arbitrary, they also show an $O(\log B_{\text{max}})$-ratio algorithm for the general PCC problem on lines. Our algorithm is still interesting, although it solves only a special case, for two reasons: (a) it is based on a simple greedy approach and (b) it considers the ratio $\frac{B_{\text{max}}}{B_{\text{min}}}$ in addition to $\frac{B_{\text{max}}}{B_{\text{min}}}$. We are not aware of any results for the PCC problem on lines bearing this ratio in mind.
8 Conclusions and Open Problems

The results presented in the foregoing chapters are summarized. The chapter concludes with a list of interesting problems that stem from the research conducted in this thesis.

In this thesis, we have investigated a variety of call control problems. We restricted ourselves mainly to studying them on the ring topology for two reasons. The first reason is the practical necessity of addressing these problems in optical networks where rings form the building blocks. The second reason is theoretical in that they have not been resolved or addressed before in literature, although the ring topology is a relatively simple one. In Section 8.1, we summarize the results elaborated in Chapters 3 through 7. In Section 8.2, we close this final chapter by listing open problems related to the results of the thesis.

8.1 Summary of results

- In Chapter 3, we studied the on-line MEDP and MEDPwPP problems on trees of rings. For the on-line MEDP problem on a tree of rings $T$, we gave an $O(\log d_T)$-competitive randomized algorithm, where $d_T$ is the minimum possible diameter of a tree that results from deleting one edge each from every ring in $T$. For the on-line MEDPwPP problem on a tree of rings $T$, we showed an $O(\log \ell_T)$-competitive randomized algorithm, where $\ell_T$ is the length of a longest path in $T$. Both these algorithms were shown to be the best possible randomized and non-preemptive algorithms up to constant factors.

- In Chapter 4, we studied the off-line MEDPwPP-$k$ problem on arbitrary graphs and the off-line MEDP-$k$ problem on undirected and bidirected trees of rings. We showed fixed parameter tractable (FPT) algorithms for all of them.

- In Chapter 5, we considered the off-line, uniform RCC problem on rings. When the profits of calls are all equal to unity, we gave an algorithm
that accepts and routes at most 3 fewer calls as compared to an optimal algorithm. For arbitrary profits, a 2-approximation was presented.

- In Chapter 6, we investigated the off-line, uniform PCC problem on rings. A simple 2-approximation was presented for the general problem. For various special cases optimal algorithms or approximation schemes were detailed. We also showed that finding a polynomial time optimal algorithm for the general problem would yield a polynomial time algorithm for the exact matching problem on bipartite graphs. The complexity of the latter problem has not been resolved to date.

- In Chapter 7, we examined the off-line, non-uniform call control problem on lines. Algorithms with good approximation ratios are not known for the general problem. We gave an FPTAS and a PTAS, respectively, for the problem when the number of vertices on the input line is constant and when the length of the routes of calls is bounded by a constant. Let $B_{\text{max}}$ ($B_{\text{min}}$) and $P_{\text{max}}$ ($P_{\text{min}}$) be the maximum (minimum) bandwidth requirement and profit of the calls, respectively. If $\min\{\frac{B_{\text{max}}}{B_{\text{min}}}, \frac{P_{\text{max}}}{P_{\text{min}}}\}$ is bounded by a constant, a constant factor algorithm was provided when the input set of calls have routes of equal length. If the above value is arbitrary, an algorithm that achieves an approximation ratio that is logarithmic in that value (apart from constant factors) was presented when the input set of calls have routes of equal length.

8.2 Further research

In this section, we discuss open problems that are either closely related to or are a follow up of the problems addressed in this thesis.

- The algorithms of Chapter 3 were randomized and non-preemptive in nature. When the underlying topology is a line, $O(1)$-competitive algorithms (randomized and preemptive) are shown for the on-line MEDP problem and for instances of the on-line call control problem on lines when the profits of calls are proportional to their bandwidth requirements in [AA99]. Constant factor competitive algorithms for the same problem when preemption is allowed are not known even for trees. This is certainly a problem that merits consideration.

- In Chapter 4, we addressed the off-line MEDP-$k$ problems on undirected and bidirected trees of rings and showed fixed parameter algorithms for them. A natural extension is to consider the fixed parameter
tractability of the off-line RCC-$k$ problems on them. A first such re-
sult in that direction is shown in [AEHS03], where the off-line RCC-$k$
(or equivalently, PCC-$k$) problem on trees, when calls have uniform
bandwidth requirements, is shown to be fixed parameter tractable.

- In Chapter 5, while we gave approximation algorithms that guaranteed
additive and constant factors for the off-line, uniform RCC problems
on rings we could not resolve their computational complexities. Also,
these algorithms are not combinatorial in nature as they start off by
solving linear programs. It would be worthwhile to first resolve their
complexities and then consider designing combinatorial algorithms for
them.

- We have indicated the difficulty of finding a polynomial time optimal
algorithm for the off-line, uniform PCC problem on rings in Chapter 6.
The 2-approximation shown there is a straightforward one. A natural
question, therefore, would be: Can the ratio be improved by devising
a better (and sophisticated) algorithm?

- The results of Chapter 7 on off-line, non-uniform call control on lines
are without doubt only a starting point in addressing those problems.
The only known lower bound on the complexity of the problem in its
generality is its $\mathcal{NP}$-hardness. The upper bound is an algorithm with
approximation ratio that is logarithmic in the input values. Constant
factor algorithms are known only for special cases, like instances under
the bottleneck assumption. Narrowing the gap is an important open
problem to further study the off-line, non-uniform call control problems.
Bibliography


116 Bibliography


Index

2-cut, 49, 53
W[2]-hard, 48
W[\nu]-hard, 48
\textit{NP}, 10
  -complete, 10, 20, 22
  -hard, 10–11, 15–16
\rho-competitive, 12
3-HittingSet-\textit{k}, 51
d-HittingSet-\textit{k}, 60
\textit{HittingSet-}\textit{k}, 51
\textit{Knapsack}, 20
\textit{MinVertexCover}, 15

adjacent vertices, see vertex, adjacent
algorithm, 9
  approximation, 7, 11
  execution time of, 9
  fixed parameter tractable, see \textit{FPT}
off-line, 12
on-line, 12, 28
polynomial time, 10
randomized, 11, 100
running time of, see algorithm,
  execution time of
approximation ratio, 11
approximation scheme, 11
  fully polynomial time, 11
  polynomial time, 11

bidirected, 7, 15
  chain, 53
  ring, 20, 52–59
  tree, 21
  tree of rings, 48, 52–60
bottleneck assumption, 23
bounded search tree technique, 46
call, 4, 16
  crosses a cut, 75
  profit of, 5, 16
  route of, 5, 16
call admission control, see call control
call control, 4, 5
  non-preemptive, 6
  non-uniform, 6
  off-line, 6
  on-line, 6
  pre-routed, 6
  preemptive, 6
  routing and, 6
  uniform, 6
chain, 13
chord, 49
  directed, 53
circular ones property, 85
competitive ratio, 12
crossing calls, 63
decision problem, 9
\textit{DP}, 96, 102
dynamic programming, see \textit{DP}
edge disjoint, 18

123
Index

edges
  incident, 12

exact matching, 81, 85
feasible, 17
solution, 10, 17–18
FPT, 16, 46
FPTAS, 11

graph
  circular-arc, 20
  component of a, 13
cut in a, 13
diameter of, 13
directed, 15, 21, 48
diameter, 13
dege capacitated, 15
directed, 15

Hub Network, 2
ILP, 62
incident edges, see edges, incident
incident vertices, see vertex, incident

integer linear program, 62
  relaxation of, 63

language, 10
lca, 15
lca-ring, 31
least common ancestor, 15
line, 13

maximization problem, 10
MCB, 84
MEDP, 18, 28
MEDP-k, 19, 48
MEDPwPP, 18, 30
MEDPwPP-k, 19, 47
minimization problem, 10
minimum cost flow, 88
multi-commodity flow, 21–23

multicover problem, 84

Okamura Seymour theorem, 72
optical
  fibers, 1
  networks, 1
optimization problem, 10
parallel calls, 63
parameter, 15, 45
parameterized complexity, 15
PCC, 17, 82, 101
PCC-k, 19
profit, see call, profit of
PTAS, 11, 77, 94
quality of service (QoS), 4
RCC, 17, 61
RCC-k, 19
ring, 13, 61, 81
topology, 2–3
Ring Architecture, 2
route, see call, route of
route function, 63
SONET, 2
sub-call, 49
totally unimodular matrix, 83
tree, 13
tree of rings, 13, 27, 45
vertex
  adjacent, 12
  neighbour of, 12
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>26th May 1976</td>
<td>Born in Alappuzha, India</td>
</tr>
<tr>
<td>1981 – 1993</td>
<td>Attended school in Palakkad, India</td>
</tr>
</tbody>
</table>
| June 1993 – March 1996 | Bachelor’s student at Sri Sathya Sai Institute  
                        | of Higher Learning, Bangalore, India                               |
| November 1996    | B.Sc (Honours) in Mathematics                                      |
| August 1996 – July 2000 | Master’s student at Indian Institute of Science,  
                        | Bangalore, India                                                  |
| October 2000     | M.E. in Computer Science & Engineering                             |
| October 2000 – March 2001 | Pre-doc student at Institute for  
                        | Theoretical Computer Science, ETH Zürich                          |
| April 2001 –     | Doctoral student under Prof. Thomas Erlebach at Computer Engineering & Networks Lab, ETH Zürich |