Optimisation models and algorithms for multicast message routing and power control in wireless multihop networks

by

Joseph Peter Hodgskiss

A thesis submitted to
The University of Birmingham
for the degree of
Doctor of Philosophy

School of Mathematics
The University of Birmingham
May 2010

Abstract

The data capacity of a link within a wireless network depends in a nonlinear way on the communication resources allocated to it. Finding the optimal way to transmit data through a network consisting of many wireless devices can therefore be represented as a nonlinear optimization problem over network flow variables and communication resource variables.

In this thesis we develop a nonconvex optimization problem for transmitting unicast and multicast messages through a time-slotted multi-hop wireless network. Multicast messages are handled in an optimal way through the use of network coding to allow data packets to be combined ensuring they are useful for multiple destinations. The benefits of network coding over other routing strategies are tested numerically.

We look at simple networks to gain insight into the way various parameters affect the nonconvex behaviour before going on to develop algorithms which can be applied in a distributed manner, and make use of the coupled structure of the problem.

We implement a subgradient method for solving the dual problem, and then look at ways to accelerate its convergence. We also investigate the behaviour and convergence of a simple but effective primal co-ordinate descent method before numerically investigating its performance.

Acknowledgements

I would like to give huge thanks to my supervisor Jan-J. Rückmann, and Jörg Fliege, who supervised me for the first year of my PhD and has continued to support me ever since. Not only has their intellectual input and academic support helped me immeasurably, but their wisdom and guidance has helped me to stay on track on the roller coaster that has been my PhD course! I would also like to thank Armin Dekorsy for his great help in technical matters, as well as for inviting me on an invaluable research visit.

I would also like to acknowledge my friends and colleagues within the maths department, without whom my four years here would have been nowhere near as much fun. I'd especially like to thank the members of the legendary Postgrad Mathletic football team for providing a highlight to even the dullest week, and enduring my countless recollections of goals in the pub afterwards. The other, possibly even greater team of which I am a proud member, Office 315, deserves my wholehearted gratitude for putting up with my endless arguments with LaTeXand MATLAB.

A massive thank you also goes to my family and Ella, for their encouragement and support and the countless other reasons I couldn't begin to list here.

Contents

1 Introduction **1**

2 A Mathematical Model for Transmitting Unicast Messages **5**
2.1 Physical Properties of Multi-hop Wireless Networks 5
2.1.1 Multihop Wireless Networks 6
2.1.2 Unicasts 8
2.1.3 Communications Resources and Access Schemes 9
2.2 State of the Art 11
2.3 Mathematical Optimization 13
2.4 Developing a Mathematical Framework 15
2.4.1 The Basic Topology 15
2.4.2 Scheduling 16
2.4.3 Channel Access Methods 17
2.4.4 Variables 20
2.4.5 Parameters 21
2.4.6 Objective Functions 22
2.4.7 Constraints 23
2.4.8 The Mathematical Program 25
2.4.9 Observations and simplifications 26
2.5 Some Theoretical Results for a Simple Problem Formulation 27
2.5.1 The Simplest Problem 27
2.5.2 Reducing the Problem to One Variable 30
2.5.3 The Behaviour of a Certain Function 33
2.5.4 Location of Global Minima for our Simple Problem 35
2.5.5 The Symmetrical Case 36
2.5.6 The Most General Case of the Simple Problem 38

3 Expanding the Model to Handle Multicasts **42**
3.1 The Naive Approach 44
3.2 Tree Finding Approach 46
3.3 Motivating Network Coding 50
3.3.1 Increased Throughput in Multicasts and Multiple Unicasts 51
3.3.2 Decreasing Energy Consumption 53

3.3.3 Decreasing Delay . . . 54
3.4 Technical Aspects of Network Coding . . . 55
3.4.1 Encoding . . . 55
3.4.2 Decoding . . . 57
3.4.3 Arithmetic Over Finite Fields . . . 59
3.4.4 Important Results . . . 62
3.5 A Worked Example . . . 64
3.5.1 Solving using the naive approach . . . 65
3.5.2 A More Intelligent Routing Solution . . . 65
3.5.3 A Network Coding Solution . . . 66
3.6 Using Network Coding to Extend our Model to Multicast . . . 67
3.6.1 Variables . . . 67
3.6.2 Objective Function . . . 68
3.6.3 Constraints . . . 68
3.6.4 The Complete Multicast Program . . . 70
3.7 A Model for Multiple Unicasts, Multicasts, and Broadcasts . . . 71
3.7.1 Variables . . . 72
3.7.2 New Parameters . . . 72
3.7.3 The General Model . . . 73
3.8 Computational Results Comparing Network Coding to Naive Routing . . . 74
3.8.1 Parameter Values . . . 76
3.8.2 Results . . . 76

4 Solution Methods for the General Problem 81
4.1 Tackling the Non-Convexity . . . 82
4.1.1 Convex Approximation . . . 83
4.1.2 Comparison of Approximation with Original Problem for Simplest Case . . . 88
4.2 Numerical Solution techniques . . . 89
4.2.1 Duality . . . 91
4.2.2 Solving the Dual Problem . . . 96
4.2.3 Subgradient Methods for Non-Smooth Optimization . . . 97
4.3 Application to Our Unicast Problem . . . 98
4.4 Algorithm for Solution Through Dual Decomposition . . . 101
4.4.1 Implementation Issues . . . 103
4.4.2 Application to the Simplest Problem . . . 105
4.4.3 Application to the 1-3-5-1 Network . . . 108
4.5 An Acceleration Technique . . . 110
4.5.1 Aitken's Method and Steffensen's Method . . . 111
4.5.2 Application of Acceleration Methods to Our Algorithm . . . 112

5 A Primal Co-ordinate Descent Approach **119**
5.1 Motivation . . . 119
5.2 Equivalence of Problem Statements . . . 123
5.3 The Two Subproblems . . . 128
5.3.1 Multi-Commodity Flow Problem . . . 128
5.3.2 Standard Power Control Problem . . . 129
5.4 The Co-ordinate Descent Algorithm . . . 131
5.5 Difficulties Encountered When Faced With Interference . . . 138
5.6 Numerical Testing of the RPCD Algorithm . . . 140
5.6.1 Implementation of the RPCD in our Tests . . . 140
5.6.2 Results . . . 141

6 Conclusions, Open Questions and Research Directions **146**

A Convex Programming **152**

B Glossary of Communication Engineering Terms **154**

C Additional Data for Dual Decomposition Results **157**

D Parameter Values used in Results for Chapters Three and Five **160**

List of References **164**

List of Figures

2.1 *Graphical representation of the three main access schemes* 21
2.2 *The simplest network exhibiting interference properties.* 28
2.3 *Graph of λ against transmit power in the four qualitatively different situations* 39
2.4 *Graph of λ against transmit power for increasing message size S* 40

3.1 *An example of a network where the naive approach to multicasting is also the best approach.* 44
3.2 *An example of a network where the naive approach to multicasting is clearly suboptimal* 45
3.3 *An example of how we can construct a network in which the naive approach performs arbitrarily badly.* 46
3.4 *An example of achieving maximal multicast throughput by packing Steiner trees.* 47
3.5 *The butterfly graph, a network where Steiner tree packing cannot achieve the theoretical maximum multicast flow* 49
3.6 *All possible Steiner trees which include nodes s, d_1 and d_2 of the butterfly graph* 50
3.7 *Example of how network coding can be used to increase total throughput in a network carrying multiple unicast sessions* 51
3.8 *Example of how network coding can be used to increase total throughput in a network carrying a single multicast session.* 52
3.9 *Example of how network coding can be used to decrease energy required per packet sent in a network carrying a single multicast session.* 53
3.10 *Example of how network coding can be used to decrease total delay in a network carrying a single broadcast session.* 54
3.11 *Feasible unicast flows of 100 packets from s to d_1 and d_2.* 64
3.12 *A naive multicast solution, treating the two unicasts in figure3.11 as independent.* 65
3.13 *A more intelligent routing strategy, using the fact that the same 100 packets are required at both destinations.* 65
3.14 *A Network Coding Solution, random linear encoding is performed at the source and the central node.* 66
3.15 *A 1-3-5-3 Backhaul network.* 75

3.16 *Total multicast transmit energy for varying message sizes in Megabits, with or without interference in the butterfly network over 20 timeslots.* 77
3.17 *Total multicast transmit energy for varying message sizes in Megabits, with various interference levels in the backhaul network over 20 timeslots* 78
3.18 *Total multicast transmit energy for varying encoding coefficient overheads, with or without interference in the butterfly network over 20 timeslots, with message size 20.* 79
3.19 *Total multicast transmit energy for varying encoding coefficient overheads, with various interference levels in the backhaul network over 20 timeslots, with message size 20.* 80

4.1 *Comparison of Original problem to Convexified Problem for varying interference levels* 90
4.2 *Convergence of the dual function for varying stepsizes in the simple case with a unique global minimizer* 107
4.3 *Convergence of the dual function when we have a local and global minimizer* 107
4.4 *Convergence of the dual and primal function values in the 1-3-5-1 network with zero interference* 109
4.5 *Convergence of the dual and primal function values in the 1-3-5-1 network with high interference* 110
4.6 *Convergence of the dual function value using Aitken acceleration with no interference* 114
4.7 *Convergence of the dual function value using Aitken acceleration with low interference* 114
4.8 *Convergence of the dual function value using Steffensen's Acceleration* . . . 115
4.9 *Failure of Steffensen's Acceleration* 116
4.10 *Steffensen's acceleration provides small gains with appropriately chosen step-size parameter* 117

5.1 *A simple linear program which cannot be solved by co-ordinate descent methods* 121
5.2 *A simple quadratic program which cannot be solved by co-ordinate descent methods* 122
5.3 *Total multicast transmit energy for varying message sizes, with full interference in the butterfly network over 20 timeslots, comparing RPCD to a blackbox solver.* 143
5.4 *Total multicast transmit energy for varying numbers of timeslots, with full interference in the butterfly comparing RPCD to a blackbox solver.* 143
5.5 *Total multicast transmit energy for varying message sizes, with full interference in the backhaul network over 20 timeslots, comparing RPCD to a blackbox solver.* 144

5.6 *Total multicast transmit energy for varying message sizes, with inter-cell interference only in the backhaul network over 20 timeslots, comparing RPCD to a blackbox solver.* 145

List of Tables

3.1 *Addition and multiplication tables for the finite field* $\mathbb{F}_3$. 60
3.2 *Addition and multiplication tables for the finite field* $\mathbb{F}_4$. 60

C.1 *Uniqueness of Solution Data for the non-convex subproblem with no Interference* 157
C.2 *Uniqueness of Solution Data for the non-convex subproblem with Low Interference* 157
C.3 *Uniqueness of Solution Data for the non-convex subproblem with High Interference* 158
C.4 *Comparing dual decomposition solution to black box global solver solution in the 1-3-5-1 network with zero interference* 158
C.5 *Comparing dual decomposition solution to black box global solver solution in the 1-3-5-1 network with zero interference* 159

D.1 *Gain Matrix for Backhaul Network with zero Interference* 161
D.2 *Gain Matrix for Backhaul Network with Intracell Interference* 162
D.3 *Gain Matrix for Backhaul Network with Intercell Interference* 163

List of Commonly used variables and parameters

Variable	Unit	Description
$p_{e,t}$	Watts	Power used to transmit message along edge e in timeslot t
$f_{e,t}^{m}$	Bits	Amount of message m transmitted along edge e during timeslot t
$b_{n,t}^{m}$	Bits	Amount of message m stored in buffer at node n during timeslot t
$c_{e,t}^{m,d}$	Bits	Amount of conceptual flow of message m, destined for d, transmitted along edge e during timeslot t
$d_{n,t}^{m,d}$	Bits	Amount of conceptual flow of message m, destined for d, stored in buffer at node n during timeslot t

Parameter	Unit	Description
B	Hertz	Available transmission bandwidth
τ	Seconds	Timeslot length
N_0	Watts/Hertz	Spectral Noise Density
σ^2	Watts	Background Noise
$P_e^{\max}$	Watts	Edge power limit
$P_n^{\max}$	Watts	Node power limit
$B_n^{\max}$	Bits	Node buffer limit
A		Gain Matrix, $a_{i,j}$ is gain at receiver of edge j from transmitter of edge i
$t^{\max}$		Number of available timeslots
s^m		Source node of message m
D^m		Set of destination nodes of message m
S^m	Bits	Size of message m
V		Vertex set of wireless communication devices
E		Edge set of wireless communication links between devices
G		Directed graph representing communication network, $G = (V, E)$

In situations where there is clearly only one timeslot, message or destination, subscript t, and/or superscript m, d will be dropped. In addition to these variables and parameters, Ψ will commonly be used to represent channel capacity functions, and Φ will be used to represent objective functions.

Chapter 1

Introduction

Wireless data transmission is a technology for which new applications are constantly being found. As the technology improves, there are demands for ever-increasing amounts of data to be transmitted. While in wire line networks the amount of data able to be transmitted from one point to another depends simply on the quality and size of wire used, and can therefore be seen as fixed, in wireless networks the situation is predictably more complicated. Wireless network link capacities depend on the resources allocated to them, such as frequency, time, and transmit power, as well as background environmental conditions. As the demand on wireless networks increases, it becomes more important to make the best possible use of these resources, and in order to do this it is necessary to simultaneously consider the best way to allocate these communication resources as well as the best way to transmit data through the network.

The specific problem within the broad area of wireless communication that this thesis is concerned with is that of finding the optimal way to transmit at a number of unicast and multicast messages through a time-slotted multi-hop wireless network. A unicast is a message from a single sender to a single receiver and a multicast is a message from a single sender to multiple receivers.

A multi-hop network is a network where each communication device is able to act as

a relay for data destined for other devices, and so a typical path from sender to receiver will consist of many short "hops" between intermediate relaying devices. In the problem we are considering, we allow multipath routing, that is to say a message from sender to receiver need not flow along a single path through the network, but is able to split into smaller portions, each of which can take different paths.

It is worth explicitly mentioning that in the model developed in this thesis, we only consider point to point communication and not broadcast communication. This is an important distinction since in point to point communication, a message transmitted by a device can only be received by the one device it is intending to send to, whereas in broadcast communication a message transmitted by a device can be received by any intended receivers that the message reaches suitably clearly.

The optimization problem we are thus considering is that of allocating wireless communication resources, and designating multipath routing strategies so that each message reaches its destination within a designated number of timeslots in such a way as to minimize some measure of cost of using the network. The relationships between communication resources and data capacities are often complex, and in the code division multiple access scheme which we are most interested in, this relationship is in fact non-convex, leading to a non-convex optimization problem.

Put simply, the first half of this thesis is concerned with finding the best way to model this problem as a mathematical optimization problem and trying to understand some properties of the problem. The second half of the thesis is concerned with designing and investigating suitable algorithms for solving the problem presented in the first part.

Chapter two beings by looking briefly at the physical processes involved in wireless communication, including the various channel access schemes, before introducing a mathematical framework and nonlinear program for the problem of transmitting unicast messages. The chapter concludes with a study of the simplest possible instance of the problem

in order to illustrate and investigate the nonconvexity of the problem and find conditions under which the problem has a unique solution.

In chapter three we tackle the problem of incorporating multicast transmissions into the model developed in chapter two. We investigate three possible approaches, namely a naive routing approach which ignores the fact that the same data is being sent to multiple destinations, a combinatorial tree packing approach, and a new approach using the recently developed principle of network coding. With network coding, rather than just allowing the intermediate devices to forward and copy incoming messages, we allow them to perform additional operations on the incoming packets, such as forming linear combinations of them. This mixing up of packets means data travelling through the network is more likely to be useful to more receiving devices of the multicast transmission. At the end of the chapter we detail an extended optimization problem for the transmission of multiple unicast and multicast transmissions, and finally we compare the naive approach to the network coding approach through numerical simulations.

At the beginning of chapter four we look into ways of tackling the non-convexity of the optimization problem and examine the suitability of some of the commonly employed approaches. The bulk of this chapter is involved in employing the well known technique of solving the dual problem through a subgradient method, which lends itself to being solved in a distributed manner and exploits the coupled structure of the problem but unfortunately converges to a solution extremely slowly. The final part of the fourth chapter is spent improving this low convergence rate by using an acceleration technique based on Aitken's Δ^2 technique.

In the fifth chapter we look at an algorithmic approach where our main goals are to find a solution quickly and exploit the underlying structure of the problem. These goals are met through a primal block co-ordinate descent method. We investigate the validity of such a method and show why it finds a solution so quickly, as well as finding

conditions under which it is guaranteed to reach a global optimum. We look briefly at ways in which this approach can be solved in a distributed fashion before highlighting potential difficulties of employing this method in certain situations, and finally investigate numerically how close the algorithm comes to an optimal solution in the general case for which global convergence is not guaranteed.

Finally, in chapter six we reflect on the main outcomes of the thesis as well as outlining some of the most interesting open questions, and highlighting the most promising directions for future research.

Due to the nature of this work, it is relevant to both mathematicians and communications engineers. In an attempt to make the communications engineering sections easier to read for mathematicians, and to make the mathematics easier for non-mathematicians, there is a glossary of commonly used communication engineering terms in appendix B and a brief recap of the theory of convex programming in appendix A.

Chapter 2

A Mathematical Model for Transmitting Unicast Messages

In this chapter we will build the mathematical programming problem which we will later set out to solve. In order to ensure that the mathematical problem we construct is appropriate, it is of course first necessary to look in detail at the real-world situation we wish to model. For a thorough treatment of the way data networks work, see [4], and for an in depth technical discussion of the way a wireless network works, see [28] and [22].

2.1 Physical Properties of Multi-hop Wireless Networks

To achieve the stated aims of this work, it is important to first understand as deeply as possible the physical situation we hope to model and the real-world problem we wish to solve. Without this understanding it is impossible to know whether approximations, assumptions and simplifications we make in order to lead us to a tractable optimization problem leave us with meaningful techniques and data, or a problem which differs from the original physical problem in some fundamental way to such an extent that the model no longer suitably resembles the physical process, and so solving it becomes meaningless.

To this end, we will outline the concept of a wireless multihop network as well as describing how messages are passed through such a network. It is also important to understand applications of such a technology, both to provide motivation for solving these problems and again to understand how best to design the model to capture the key characteristics of the physical process.

2.1.1 Multihop Wireless Networks

In order to consider the problem of transmitting messages through a multihop wireless network, we first need to know what is meant by such a network. By a multihop network, we mean that a source device wishes to transmit data to a destination device by using intermediate nodes to relay the information. This differs from the widely used single hop network, where the transmitter node sends the messages directly to a receiver node.

By allowing intermediate nodes to relay data in this way, we allow the message to take multiple smaller steps. In wire line networks, there would be little advantage in this approach since there is not much attenuation when transmitting through cables as compared to through the air, but in wireless networks it is natural to see that two devices close together find it much easier to communicate with each other than two devices separated by a great distance, and so hopefully less effort is required for the communication to take place. The question of whether the total effort required to make multiple small hops is less than the effort required to make one large hop requires in depth understanding of just what this effort entails and is fundamental to the idea of multi-hop networks.

There are two main types of wireless multihop network we would like to consider, infrastructure networks and ad-hoc networks, both of which have many applications.

In infrastructure networks, the wireless devices work in a pre-specified way to determine the routing of information. There are multiple applications of infrastructure networks including sensor networks and wireless backhaul networks.

In wireless ad-hoc networks, there is no pre-existing infrastructure, all devices forward data for other users. The routing scheme needs to be adaptive as devices are free to enter and leave the ad-hoc network.

Wireless backhaul networks are used for extending the coverage area of communications networks without extending the actual wire line infrastructure. It is often cheaper to build a network of large transmitters to link users back to the internet or telephone network than to lay cable to individual local transmitters. These backhaul devices then communicate to the internet or telephone network, using any backhaul devices between them and the destination as relays.

Sensor networks are used for monitoring environmental conditions. Sensor devices could range from temperature sensors to motion sensors, all needing to communicate their findings to a central control. Since there could be many sensors in a network, it would make sense to give them cheap, low powered transmitters allowing them to find a multihop path along which to send their data to the relevant control.

Clearly, depending on the application of a wireless network, there are different challenges to be addressed and different priorities. For example, mobile devices in ad-hoc networks are, by their mobile nature, likely to be reliant on batteries as a power source, and so optimization with respect to power is likely to be more important than in a static network where each device may be connected to a permanent power supply.

Also, in ad-hoc networks, the rate at which the topology of the network changes could necessitate a model which generates adaptive solutions which can quickly change as the network changes, or robust solutions, which are likely to succeed even if the network changes. In static multi-hop networks known as mesh networks the topology is likely to be much more stable. It is also the case that communication overheads become more or less significant for different applications:

In a static mesh network with a constant flow of data, it could be the case that a

centralized computation could be made to determine the best way to route the data, and then this information could be distributed amongst the intermediate devices with details of how to route the data. In a network where either the traffic requirements are likely to change quickly, or the network topology is liable to change, it would be more useful for the decisions to be made locally, in a distributed manner, whereby each node reacts to its immediate surroundings in order for a global best course of action to be taken.

Having seen that different types of network and applications require different things, a mathematical model would have to be flexible enough to accommodate any type of network or application, or decide which of the above aspects (distributivity, adaptivity, robustness of solutions, speed of solution) needs to be prioritised.

Having looked into the various types of wireless network, it is important now to look at the physical processes taking place. However, since wireless network functionality is such a deep and involved topic with many layered processes happening simultaneously, it is impossible to go into full detail of everything that happens in a wireless network, we will instead present an overview focussing on the aspects most important to our research goals.

2.1.2 Unicasts

Suppose we have a network of communication devices and device A wishes to transmit a message to device B, possibly through intermediate devices $N_1, \ldots, N_k$. This is known as a multihop **unicast**.

The information at user A is encoded into a string of 1s and 0s and then divided into packets of data. These packets are stored in a memory at device A known as a buffer.

The next step is to wirelessly transmit these packets, possibly sending them along separate paths, through a series of intermediate devices, to be finally stored at device B. Transmissions are made in distinct time slots and at any given time, a device is able to

send packets or receive packets, but not both. At any given device, the packets can be stored in buffer for as long as required, giving the network a great deal of flexibility, but also inevitably leading to a great number of decisions to be made.

Even before worrying about how these packets are sent, we already have to decide how many packets to send between any two devices in any time slot, and how many packets to store at each buffer. Inevitably, transmitting these packets between devices requires communications resources, and so we need to be able to decide how to make use of these resources in the best possible way.

2.1.3 Communications Resources and Access Schemes

Lets now take a closer look at the communication resources we need to manage:

Due to the finite bandwidth of the parts of the light spectrum used for wireless communication, the immediacy of information required by service users, and requirements on battery life, it becomes increasingly important to make sure we use our resources in the best way possible. As the number of applications for wireless networking increases, and the number of communication devices using the same bandwidth increases, this matter becomes all the more pressing.

There are four main ways in which several different messages can be transmitted at the same time over a wireless medium, and the standard analogy, which we shall use also, is to imagine two separate conversations happening in the same room.

The first way to allow multiple channels access to the medium is known as **frequency division multiple access**, (FDMA). Here, the bandwidth allocated to wireless communication is split further, and different devices transmit at different frequencies. In the room analogy, this would be equivalent to different pairs of speakers and listeners speaking at different pitches, and with ears capable of hearing different pitches. Unfortunately, at any time when a device is not transmitting, we are not using the entire available bandwidth.

A second approach is known as **time division multiple access**, (TDMA). Here all channels use the entire available bandwidth, but transmit at different times. In the room analogy, this is obviously equivalent to allocating each speaker times at which they are allowed to communicate.

The third approach is to open up the entire available bandwidth and time period to all sessions and distinguish between different sessions using codes. In the room analogy, this is like allowing all conversations to occur simultaneously, but with each in a different language, and with conversationalists equipped with ears that can block out conversation which is not in their language. This approach is known as **code division multiple access**, (CDMA), and potentially makes the greatest use of the available bandwidth and time.

A fourth, and in a sense simplest way to allow the conversations to happen is to have the conversations happening at opposite ends of a large room, making it easy to hear your partner but with the other voices as a distant murmur. Obviously this is impractical on a local scale as it would require all wireless devices to move close to each other if they wanted to communicate, which would somewhat defeat the point of the exercise. However, using multiple antennas it is possible to split the space into separate entities known as beams. This approach is already applied in several existing technologies.

If we have a large area using multihop ad-hoc networks, we can break it down into smaller cells, and then if two cells are not too close to each other, anyone in these cells can communicate at the same time over the same frequency as someone in a different cell, without worrying about overhearing them. This approach is known as **frequency reuse**.

It is of course important to know how resource use relates to data capacity for these various access schemes. With any access scheme, the theoretical capacity of a wireless link can be found using the ground breaking work of Claude Shannon, [42]. He established that the amount of information able to be transmitted over a wireless link in a given

time slot is related to the transmit power, noise and interference, bandwidth and the length of the timeslot. From the definitions of FDMA, TDMA and CDMA, we see that interference from other transmitting devices is only a factor under CDMA, and under the other schemes the only interference comes from background noise, usually in the form of Gaussian white noise. With this in mind, we see that:

For FDMA, capacity = Ψ(transmit power, allocated bandwidth, background noise),

for TDMA, capacity = Ψ(transmit power, allocated timeslot length, noise),

for CDMA, capacity = Ψ(transmit power, other transmit powers in the network, noise).

Depending on which access scheme is used, we have various different communication resources which we need to decide how to deploy. All of this will be formalized later in the chapter where we shall develop a mathematical framework and optimization problem for modelling this physical process. First, with a better understanding of the problem we are trying to model, we are able to appreciate other research aimed at solving this, and similar problems.

2.2 State of the Art

In this section we shall look at the history of research involved in optimization in communication networks, specifically in wireless networks.

The problem of single path routing of unicast messages through data networks with fixed capacity links has been heavily studied as far back as the 1970s, when most data communication was carried through fixed capacity wireline networks. Work in this area often focuses on minimizing delay. Many centralized as well as distributed algorithms exist for finding minimum delay routing strategies in fixed capacity networks, see for example [20] for a distributed algorithm to compute the best routing strategy in the case where we want to alter an existing network, and [44] for an adaptive distributed algorithm using

relaxation. The extension to multipath routing has also been well studied, see [32] for an algorithm for optimal multipath routing and flow control. In [2], Bertsekas provides an excellent overview of the optimization problems associated with networks, such as minimum cost flow problems and max-flow problems, as well as discussing efficient ways of solving them.

The problem of optimal resource allocation in interference limited wireless networks where we are given the capacity requirements of a number of users in a network where they are all connected directly to a base station has also been investigated in considerable detail over the last twenty years or so. The problem is often to minimize the total power used whilst ensuring all users demands are satisfied, see [30], [47] and [46]. In [14] scheduling and power control are considered simultaneously to try and limit high interference situations. In [52], Yates shows that many different types of resource allocation problem where one seeks to minimize transmit power can be reduced to a fixed point iteration which can be performed in a distributed way. In [16], this idea is extended by accelerating the fixed point algorithm.

More recently, simultaneous optimization over both resource variables and network flow variables has been considered. As mentioned above, depending on the resources and access schemes being considered, the problem of simultaneous optimization over scheduling, routing and resource variables may or may not be linear or even convex. A lot of recent work has gone into cross-layer optimization in the convex case. In [51], the problem of simultaneous routing and resource allocation is considered under the assumption that the relationship between capacities and resource variables are convex. The problem is formulated as a convex optimization problem and a dual decomposition approach is suggested. In [26], further decomposition techniques are investigated for this same formulation . In [50] a framework is again presented for a convex formulation as well as a distributed gradient projection algorithm. In [11], CDMA interference limited networks

are considered but it is assumed that the relationship between the signal to interference and noise ratio (SINR) and link capacity is linear, which is valid for very low SINR values.

The nonconvex optimization problem resulting from assuming that the Shannon capacity can be achieved, as we shall, is considered in [27], but it is assumed that SINR values are all high and a convex approximation can therefore be made. A detailed discussion of possible ways of dealing with this non-convex problem can be found in [9]. Little work exists investigating when these approximations are valid, or ways in which we can solve the original non-convex problem. Before going on to develop the mathematical optimization problem, it is sensible to outline what we mean by a mathematical optimization problem, and what we mean by solving a mathematical optimization problem.

2.3 Mathematical Optimization

We will now outline some simple notions from mathematical optimization, for more information see A. A general continuous real optimization problem can be stated as:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in \Omega \end{aligned} \tag{2.1}$$

It is assumed that $\mathbf{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuous and Ω is connected. If we put restrictions on the objective function f and the constraint set Ω, this problem becomes a more specific type of optimization problem:

- If f is linear and Ω is polyhedral, 2.1 is a linear optimization problem.
- If f is convex and Ω is convex, 2.1 is a convex optimization problem.
- If f is nonconvex, or Ω is nonconvex, 2.1 is a nonconvex optimization problem.

In order to specify solutions to 2.1, we introduce the following definitions of minimizers:

Definition 2.1 *Consider the optimization problem 2.1.* $x^* \in \Omega$ *is a* ***Local Minimizer*** *of the optimization problem if* $f(x^*) \leqslant f(x)$ *for all* $x \in \Omega$ *satisfying* $\|x - x^*\| < \varepsilon$ *for some* $\varepsilon > 0$.

$x^* \in \Omega$ *is a* ***Global Minimizer*** *of the optimization problem if* $f(x^*) \leqslant f(x)$ *for all* $x \in \Omega$.

Clearly a global minimizer is also automatically a local minimizer. The distinction between convex optimization problems and nonconvex optimization problems is an important one since convex optimization problems have no local minimizers which are not global minimizers, whereas a nonconvex optimization problem can have local minimizers which are not global. It is often difficult for algorithms to distinguish between local minimizers and global minimizers, and so in a nonconvex optimization problem the best we can do is often find a local minimizer.

A natural way to tell if a point $\mathbf{x}^*$ is a local minimizer involves feasible descent directions.

Definition 2.2 *Consider* $\mathbf{x^0} \in \Omega$. $\mathbf{d} \in \mathbb{R}^n$ *is a* ***feasible descent direction at*** $\mathbf{x}^0$ *if there exists* $\varepsilon > 0$ *such that for all* $0 < \delta < \varepsilon$, $\mathbf{x}^0 + \delta\mathbf{d} \in \Omega$, *and* $f(\mathbf{x}^0 + \delta\mathbf{d}) < f(\mathbf{x}^0)$.

x^0 is a local minimizer if and only if there does not exist a feasible descent direction at $\mathbf{x}^0$.

In the following section we will gradually build a mathematical framework and optimization problem for the problem of transmitting unicast messages through a multihop wireless network in a manner which is optimal over both network flow variables and communication resource variables.

2.4 Developing a Mathematical Framework

In this section, we develop the mathematical framework. Beginning by detailing how we represent the network itself as a mathematical object, and then discussing how we handle scheduling in the network. We then discuss how the various access schemes discussed at the start of this chapter can be included in the model. This leads to a discussion of the variables of the optimization problem, the various problem parameters, the objective function and constraints.

2.4.1 The Basic Topology

Consider a set V consisting of $|V|$ communications devices. We construct our network flow model using a directed graph as follows. If device $v \in V$ is capable of sending data directly to device $w \in V$, then $e = (v, w) \in E \subseteq V \times V$, that is to say there is an edge in our directed graph from node v to node w representing the corresponding communication link. We denote the set of all links by $E \subseteq V \times V$. Let the total number of links be denoted by $|E|$. We have that $G := (V, E)$, is a *directed graph*, V is the *node set* of the graph and E is the *edge set*. We will now define two types of subsets of E which we shall make use of later. For a node $v \in V$, let

$$E^+(v) := \{e \in E | e = (v, w)\},$$
$$E^-(v) := \{e \in E | e = (w, v)\}.$$

That is, $E^+(v)$ is the set of all *outgoing edges* at node v and $E^-(v)$ is the set of all *incoming edges* at node v. We make the following assumptions about the graph G:

- If $v \in V$, $(v, v) \notin E$, that is to say there are no loops in the graph.

- The underlying graph formed by removing all directions from G is *connected*. That is to say that if $v_1, v_2 \in V$, there is either a path from v_1 to v_2, or a path from v_2 to v_1.

For an overview of the graph theory concepts and definitions we use, see [45].

We are interested in modelling a number of messages being transmitted through the network over a certain number of time slots. Let $M = \{1, \ldots, m^{max}\}$ index the set of messages to be considered. With each $m \in M$ we associate a *source node* $s^m \in V$ and a *destination node* $d^m \in V$. We assume $s^m \neq d^m \; \forall \; m \in M$. Our task for each m is to send a message of a size S^m from source node s^m to destination d^m.

In our model, a message from s^m can be broken up into pieces, each of which follows a different path to d^m. The information encoded in the different pieces enables the destination node to reform the original message. This is known as *multipath routing*.

2.4.2 Scheduling

In any given time slot, we assume that a node can be acting as a transmitter or a receiver, but not both. This is known as a half-duplex constraint. The decision of which nodes are allowed to transmit in a given time slot is the problem of scheduling. In order to incorporate this scheduling into the model we assume there is a given *colouring* of the nodes of the graph. That is to say, we can *partition* the node set V into C nonempty subsets $V_1, V_2, \ldots V_C$. In other words, $V = \bigcup_{c=1}^{C} V_c$, and $V_i \cap V_j = \emptyset, \forall i \neq j \in \{1 \ldots C\}$. To be a colouring it is required that any two adjacent nodes of V are of different colours. Here $C \geqslant \chi(G)$, the *chromatic number of G*.

Definition 2.3 *The least number of colours needed to colour a graph G is called its* chromatic number*,* $\chi(G)$.

To find the chromatic number of a graph is in general extremely hard, in fact it is an NP-complete problem, [43], that is to say no polynomial time algorithms are know for

solving this problem. Luckily for us, we do not need an optimal colouring for our graph G. Many greedy algorithms exist which provide near optimal colourings very cheaply, and even in a distributed manner, see [23] and it is a greedy algorithm that we will use to colour our graph. Having created a colouring, we now go on to think about the time slots we will use. Let $t^{\max} \in \mathbb{N}$ be the maximum number of time slots we will allow for the message transmission. We number the time slots by indices, $t \in T := \{1, \ldots, t^{\max}\}$, and then we give the time slots a colouring, with no two adjacent time slots the same colour. The easiest way to do this is cyclically, so time slots $1, C+1, 2C+1$.. have colour 1, time slots $2, C+2, 2C+2$ have colour 2 etc., but the best choice of time slot colouring is naturally problem specific. The scheduling is then put into practice by saying that nodes of colour c can only transmit in time slots of colour c. It is natural to define a colouring on the directed edges by saying the colour of an edge is the same colour as the node it originated from.

The colourings of nodes, edges and time slots can be viewed as functions $col_V : V \mapsto \{1, \ldots, C\}$, $col_E : E \mapsto \{1, \ldots, C\}$ and $col_T : T \mapsto \{1, \ldots, C\}$, where the function col_V is defined by the greedy colouring algorithm, $col_E(e) = col_V(v)\ \forall e \in E^+(v)$, and col_T is defined cyclically as above, or in any way suitable for the specific problem.

2.4.3 Channel Access Methods

Before we can go further with development of our model we need to look closely at the mathematics governing wireless communication under the various access methods described earlier in the chapter. The key result we need from Shannon is that a wireless channel is able to transmit data perfectly with no errors at a rate which is proportional to the length of time over which the transmission takes place, the bandwidth used for the transmission, and $\log_2(1+\text{SINR})$, where SINR is the signal to interference and noise ratio, that is the ratio of received signal power from the source of the message, to background

noise and the sum of received powers of interference coming from other transmissions going on in the network at the same time [42].

Time Division Multiple Access

In time division multiple access, different users transmit at different times. Time is split into slots, and in each slot, only one transmitter is active. This means all the users can transmit over the same frequency range, with the same bandwidth. The capacity of a link depends on the length of the time slot allocated for it to transmit over, as well as the transmit power for that link. Let Ψ_e be the capacity of edge e, p_e the power transmitted along that edge, and τ_e the length of the time slot allocated to that edge. Let B be the bandwidth available to transmit over and N_0 the spectral noise density. We have that, by [51]

$$\Psi_e(p_e, \tau_e) = \tau_e B \log_2 \left(1 + \frac{p_e}{BN_0}\right)$$

In the case of TDMA, the communication variables are the lengths of the time slots allocated to the various edges, and the transmit powers allocated to the various edges. This function is jointly concave in p_e and τ_e together, and is concave in p_e if τ_e is kept constant, and linear in τ_e if p_e is kept constant.

Frequency Division Multiple Access

In frequency division multiple access, different users transmit at the same time, but over different frequencies, meaning again there is no interference. The capacity of a link depends on the bandwidth allocated for it to transmit over, as well as the power allocated to that link. If we let Ψ_e be the capacity of edge e, p_e be the power transmitted along that edge, B_e be the bandwidth allocated to transmitting along edge e. Let τ be the timeslot

available to transmit over and N_0 the spectral noise density. Then, we have that, by [51],

$$\Psi_e(p_e, B_e) = B_e \tau \log_2 \left(1 + \frac{p_e}{B_e N_0}\right)$$

In the case of FDMA, the communication variables are the bandwidths allocated to the various edges, and the transmit powers allocated to the various edges. This function is jointly concave in p_e and B_e.

Code Division Multiple Access

In code division multiple access, different users transmit over the same bandwidth at the same time. Different transmitters are assigned different codes in order to distinguish the different signals. Since many different messages are being transmitted at the same time and over the same frequency, interference is experienced. The capacity of a link depends on the transmit powers on all links that are being used at the same time. Let $A = (a_{i,j})_{i=1,\ldots,|E|,j=1,\ldots,|E|} \in R_+^{|E|\times|E|}$ be the channel gain matrix for the network. $a_{j,i}$ is the power gain from the transmitter of link j to the receiver of link i and models the effects of signal attenuation due to the distance between transmitter and receiver and channel fading, [22]. High values in the diagonal terms $a_{i,i}$ indicate a good channel for link i, whereas high values in the off-diagonal terms, $a_{i,j}, i \neq j$ indicate high interference from link j to link i. Based on the well known Shannon Capacity formula, we have that the capacity for a link e is related to the power values on all links active at that time, as well as the bandwidth, B, and the length of time over which the transmission takes place, τ. Suppose we have a total of $|E|$ communication links active. The relationship between the capacity of link e, Ψ_e and the powers on all the edges is, by [27]:

$$\Psi_e(p_1, p_2, \ldots, p_{|E|}) = B\tau \log_2 \left(1 + \frac{a_{e,e}p_e}{\sum_{l=1,\ldots,|E|,l\neq e} a_{l,e}p_l + BN_0}\right).$$

With CDMA, the communication variables are the powers used to transmit over the wireless links. We shall look at the some of the second derivatives of Ψ_e to show briefly that this function is neither convex nor concave, and therefore a constraint involving this and a linear term must also be neither convex nor concave.

Now,

$$\frac{\partial^2 \Psi_e}{\partial p_e^2} = -\frac{B\tau a_{e,e}^2 (\sum_{l=1,\ldots,|E|, l\neq e} a_{l,e} p_l + BN_0)}{\ln 2 (\sum_{l=1,\ldots,|E|} a_{l,e} p_l + BN_0)^2},$$

which, by the non-negativity of all variables and parameters involved is always negative. This means if all powers are fixed on all other edges, this function is concave in the transmit power on edge e, p_e. Similarly,

$$\frac{\partial^2 \Psi_e}{\partial p_l^2} = \frac{B\tau a_{e,e} a_{l,e}^2 p_e \left(a_{e,e} p_e + 2 \left(\sum_{l=1,\ldots,|E|, l\neq e} a_{l,e} p_l + BN_0 \right) \right)}{\ln 2 (\sum_{l=1,\ldots,|E|} a_{l,e} p_l + BN_0)^2 (\sum_{l=1,\ldots,|E|, l\neq e} a_{l,e} p_l + BN_0)^2}.$$

Due to the non-negativity of all variables and parameters involved, this term is always non-negative. This means that if the power on edge e is fixed, and all powers on interfering transmissions are fixed except one, p_l, the function Ψ_e is convex in p_l.

Since Ψ_e is concave in p_e, and convex in p_l, it cannot be a convex or concave function, and so any constraint formed from it can also be neither convex nor concave.

In order to visualize the different ways in which the access methods utilize time and space, it is useful to look at Figure 2.1.

2.4.4 Variables

In order to proceed with the modelling of the problem, we now need consider the variables which we will have control over. Our variables can be split into two kinds of variables, network variables and communication variables. We consider all our variables to be real, although in physical applications, some of them would be discrete.

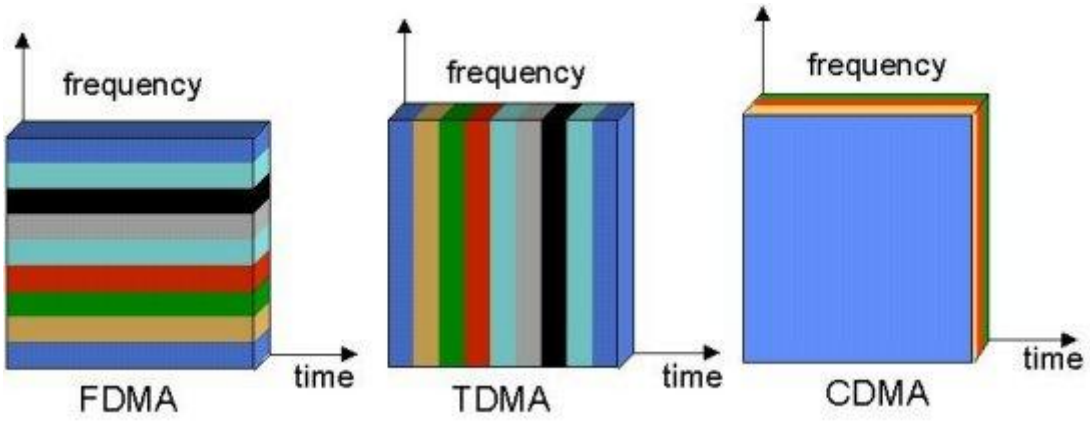

Figure 2.1: *Graphical representation of the three main access schemes*

Network Variables

Network variables are variables directly concerned with the flow of data through our network

- $f_{e,t}^m \in \mathbb{R}_+$: amount of message $m \in M$ sent along link $e \in E$ during time slot $t \in T$ (in bits).
- $b_{v,t}^m \in \mathbb{R}_+$: amount of message $m \in M$ stored in a buffer at node $v \in V$ during time slot $t \in T$ (in bits).

Communication Variables

Communication variables are variables that govern the capacity of the links in our network. Focussing on CDMA networks, the only communication variables we need to consider are the transmit powers:

- $p_{e,t} \in \mathbb{R}_+$: power used for sending data along link $e \in E$ during time slot $t \in T$ (in Watt).

2.4.5 Parameters

Our model will also use the following parameters:

- $S^m \in \mathbb{R}_+$: The size of message $m \in M$ (in bits).
- $s_m \in \{1, \ldots, |E|\}$: The source node of message m.
- $d_m \in \{1, \ldots, |E|\}$: The destination node of message m.
- $B_v^{\max} \in \mathbb{R}_+$: The maximal total buffer size at node $v \in V$ (in bits).
- $P_v^{\max} \in \mathbb{R}_+$: The maximal power usage of node $v \in V$ (in Watts).
- $P_e^{\max} \in \mathbb{R}_+$: The maximal power usage for sending data along edge $e \in E$ (in Watts).
- $B \in \mathbb{R}_+$: The bandwidth available for transmitting over (in Hertz).
- $\sigma_{e,t}^2 \in \mathbb{R}_+$: The background noise (in Watts) observed when sending part of message $m \in M$ along edge $e \in E$ at time $t \in T$. If we assume the background noise is thermal noise which is the same for all edges for all time, then $\sigma_{e,t}^2 = BN_0$, where N_0 is noise spectral density (in Watts/Hertz).
- $\tau \in \mathbb{R}_+$: The length of each time slot (in seconds).

2.4.6 Objective Functions

The main objective we will focus on is minimization of the total energy required to transmit the messages. That is to say, our objective is:

$$\text{minimize} : \sum_{t \in T} \sum_{e \in E} \tau p_{e,t}, \tag{2.2}$$

or equivalently, in FDMA or CDMA networks:

$$\text{minimize} : \sum_{t \in T} \sum_{e \in E} p_{e,t}.$$

We would also like our framework to be able to handle different objectives of course. An example of another useful objective function would be the minimization of the maximum power usage of a given node, that is:

$$\text{minimize} : \max_{v \in V} \sum_{t \in T} \sum_{e \in E^+(v)} p_{e,t}$$

In most of what follows we will assume the objective to be convex, and monotone increasing in the power variables. When the objective function is non-specific we will denote it Φ.

2.4.7 Constraints

From now on we will consider the only communication variables to be the powers allocated to the edges. We consider the bandwidth and timeslots to be constant. In order to formulate our problem as a mathematical program we need to consider all the constraints that need to be applied to our variables in the minimization of our objective function.

- First off, all our variables need to be non-negative.

$$\begin{aligned} p_{e,t} &\geqslant 0 \quad e \in E, t \in T, \\ f^m_{e,t} &\geqslant 0 \quad e \in E, m \in M, t \in T, \\ b^m_{v,t} &\geqslant 0 \quad v \in V, m \in M, t \in T. \end{aligned}$$

- Secondly, the power variables are bound above by the power capabilities of the transmitters, and the buffer variables are bound above by the storage capacities of the buffers.

$$\begin{aligned} p_{e,t} &\leqslant P^{\max}_e \quad e \in E, t \in T, \\ \textstyle\sum_{e \in E^+(v)} p_{e,t} &\leqslant P^{\max}_v \quad v \in V, t \in T, \\ \textstyle\sum_{m \in M} b^m_{v,t} &\leqslant B^{\max}_v \quad v \in V, t \in T. \end{aligned}$$

- The messages need to be at their source nodes in their entirety at the first time step, and at their destinations at the end of the time interval considered. All other buffers need to be empty at the start of transmission.

$$\begin{aligned} b^m_{s^m,1} &= S^m \quad m \in M, \\ b^m_{d^m,t^{\max}+1} &= S^m \quad m \in M \\ b^m_{v,1} &= 0 \quad m \in M, v \in V \backslash \{s^m\} \end{aligned}$$

- For scheduling, we only allow nodes to transmit data if the node is the same colour as the current time slot.

$$f^m_{e,t} = 0 \quad m \in M, e \in E, t \in T \text{ s.t. } col_E(e) \neq col_T(t).$$

- For routing, we use a modified Kirchhoff's Law, [39], to ensure that the difference between the amount of a given message that leaves a given node, and the amount of that message that enters that node is equal to the difference in the amount of that message stored at that node. This can be formulated as:

$$b^m_{v,t+1} - b^m_{v,t} = \sum_{e \in E^-(v)} f^m_{e,t} - \sum_{e \in E^+(v)} f^m_{e,t} \quad m \in M, v \in V, t \in T.$$

  Notice that due to the colouring, one of the terms on the right hand side will equal zero.

- By far the most interesting constraint is the coupling constraint. Up to now, constraints have only involved either the networking variables $f^m_{e,t}$ and $b^m_{v,t}$, or the power variables $p_{e,t}$. These variables are coupled by constraints on the capacity of the links as discussed above. Adapting the CDMA capacity formula for our problem where

we have several messages, and various time slots we have that

$$\sum_{m\in M} f_{e,t}^m \leqslant \Psi_{e,t}(\mathbf{p}) := B\tau \log_2\left(1+\frac{a_{e,e}p_{e,t}}{\sum_{l=1\ldots E, l\neq e} a_{l,e}p_{l,t}+BN_0}\right) \quad e\in E, t\in T \quad (2.3)$$

As discussed in the previous section, this is not a particularly nice constraint to deal with since it is neither convex nor concave. For brevity, we will often use $\Psi_{e,t}(\mathbf{p})$ in place of the full expression.

2.4.8 The Mathematical Program

In this section we will present in its entirety the mathematical program we wish to solve, using everything we have built up in this chapter so far.

$$\begin{array}{rcll}
 & \text{minimize} & \Phi(\mathbf{b},\mathbf{c},\mathbf{p}) & \\
 & \text{such that} & & \\
p_{e,t} & \geqslant & 0 & e\in E,\ t\in T,\\
p_{e,t} & \leqslant & P_e^{\max} & e\in E, t\in T,\\
\sum_{e\in E^+(v)} p_{e,t} & \leqslant & P_v^{\max} & v\in V, t\in T,\\
f_{e,t}^m, b_{v,t}^m & \geqslant & 0 & e\in E,\ m\in M, t\in T,\\
\sum_{m\in M} b_{v,t}^m & \leqslant & B_v^{\max} & v\in V, t\in T,\\
b_{s^m,1}^m & = & S^m & m\in M,\\
b_{d^m,t^{\max}}^m & = & S^m & m\in M,\\
b_{v,1}^m & = & 0 & m\in M, v\in V\backslash\{s^m\}\\
f_{e,t}^m & = & 0 & m\in M, e\in E, t\in T\\
 & & & \text{s.t.} col_E(e)\neq col_T(t)\\
b_{v,t+1}^m - b_{v,t}^m & = & \sum_{e\in E^-(v)} f_{e,t}^m - \sum_{e\in E^+(v)} f_{e,t}^m & m\in M, v\in V, t\in T\\
\sum_{m\in M} f_{e,t}^m & \leqslant & B\tau\log_2\left(1+\frac{a_{e,e}p_{e,t}}{\sum_{l=1\ldots E,l\neq e} a_{l,e}p_{l,t}+BN_0}\right) & e\in E, t\in T
\end{array} \quad (2.4)$$

2.4.9 Observations and simplifications

We shall now make several observations and simplifications concerning the Mathematical Program we outlined in the previous section.

- For brevity, if we want to discuss all of one type of variable at once we will use vector notation. That is,

$$\begin{aligned} \mathbf{p} &= (p_{e,t})_{e\in E, t\in T} \\ \mathbf{f} &= (f^m_{e,t})_{e\in E, m\in M, t\in T} \\ \mathbf{b} &= (b^m_{v,t})_{v\in V, m\in M, t\in T} \end{aligned}$$

- The objective function and all constraints but the coupling constraints are linear. If the coupling constraints were linear we would have a linear program, and if the coupling constraints were convex, we would have a convex program. As it is we have a nonlinear program.

- Ignoring the coupling constraints for now, we can split the constraints into constraints in the network variables, $\mathbf{f}$ and $\mathbf{b}$, and constraints in the communications variables, $\mathbf{p}$. These sets of constraints describe polyhedral sets in $\mathbf{f}$ and $\mathbf{b}$, and in $\mathbf{p}$.

  Definition 2.4 *Let*

$$\mathbf{C}_{\mathbf{b},\mathbf{f}} := \{(\mathbf{b},\mathbf{f}) \mid \text{all linear constraints in } \mathbf{b},\mathbf{f} \text{ hold}\},$$

  Let

$$\mathbf{C}_{\mathbf{p}} := \{\mathbf{p} \mid \text{all linear constraints in } \mathbf{p} \text{ hold}\}.$$

With these definitions, we now have the compact form of the problem definition:

$$\begin{aligned} \text{minimize} \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{p}) \\ \text{such that} \quad & (\mathbf{b}, \mathbf{f}) \in \mathbf{C}_{\mathbf{b},\mathbf{f}} \\ & \mathbf{p} \in \mathbf{C}_{\mathbf{p}} \\ & \textstyle\sum_{m \in M} f_{e,t}^{m} \leqslant \Psi_{e,t}(\mathbf{p}) \quad \forall e \in E, \forall t \in T. \end{aligned} \tag{2.5}$$

2.5 Some Theoretical Results for a Simple Problem Formulation

In this section we will look to solve the simplest non-convex problem of the type modelled in the previous section of this chapter. This problem is interesting in itself, but also serves as an illustrative tool for furthering our understanding of the more general problem.

2.5.1 The Simplest Problem

The simplest problem to consider is that of sending one message in one time slot through a network consisting of only two distinct, non intersecting paths from the source node to the sink node. To simplify the problem further, we do not concern ourselves with the nodes along these two paths, and instead consider a network consisting of two nodes, and two links between these two nodes, see Figure 2.2.

The optimization problem we would like to solve is that of sending a message from the source to the sink using the minimum amount of power. The links suffer from cross channel interference, and we will therefore have a non-convex problem. The problem can be formulated as follows: A message of size S is to be sent from node s to node d. All or part of the message can be sent down paths 1 and 2. The variables we need to consider in this problem are f_1 and f_2, the amounts of data to be sent down paths 1 and 2 respectively, and p_1 and p_2, the transmit powers to be used for sending messages down

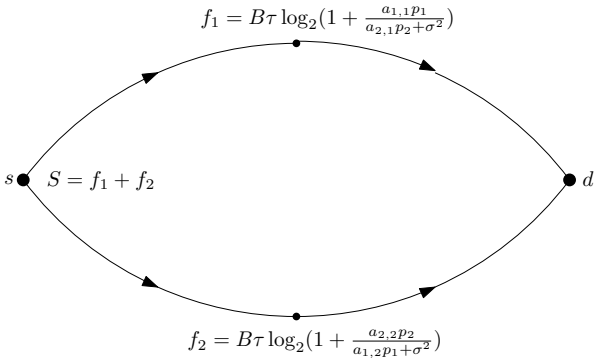

Figure 2.2: *The simplest network exhibiting interference properties.*

links 1 and 2 respectively. All four of these variables are non-negative. Our goal is to minimize the total transmit energy required to send the message. The capacity of each link is given by the Shannon Capacity formula:

$$C_1 = B\tau \log_2 \left(1 + \frac{a_{11}p_1}{a_{12}p_2 + \sigma^2}\right)$$

$$C_2 = B\tau \log_2 \left(1 + \frac{a_{22}p_2}{a_{21}p_1 + \sigma^2}\right),$$

where B is the bandwidth available for transmitting, τ is the length of the time slot, σ^2 is background noise, and a_{ij} represents the cross-channel interference from channel j to channel i, in the case that $i \neq j$, and the gain within channel i in the case that $i = j$. All six of these parameters are non-negative, and in realistic scenarios, $a_{ii} \geqslant a_{ij}$ $i, j \in \{1, 2\}$. For the problem to make sense, we also need that $a_{ii} > 0$ for $i = 1, 2$. In addition to these constraints, we require that the total transmit power used at node s is less than or equal to some maximum power. That is,

$$p_1 + p_2 \leqslant P_{\max}.$$

We are now ready to formulate this simplest of problems as a mathematical program:

$$\begin{array}{rrcl}
\text{minimize} & \Phi(p_1, p_2, f_1, f_2) & = & p_1 + p_2 \\
\text{subject to :} & & & \\
 & p_1, p_2, f_1, f_2 & \geqslant & 0, \\
h_1(p_1, p_2, f_1, f_2) & = f_1 + f_2 - S & = & 0, \\
g_1(p_1, p_2, f_1, f_2) & = p_1 + p_2 - P_{\max} & \leqslant & 0, \\
g_2(p_1, p_2, f_1, f_2) & = f_1 - B\tau \log_2 \left(1 + \frac{a_{11}p_1}{a_{12}p_2 + \sigma^2}\right) & \leqslant & 0, \\
g_3(p_1, p_2, f_1, f_2) & = f_2 - B\tau \log_2 \left(1 + \frac{a_{22}p_2}{a_{21}p_1 + \sigma^2}\right) & \leqslant & 0.
\end{array} \tag{2.6}$$

We will now look at the gradient and hessian of the nonlinear constraints g_2 and g_3.

$$\nabla g_2(p_1, p_2, f_1, f_2) = \left(-\frac{Ba_{11}}{\ln 2(a_{11}p_1 + a_{12}p_2 + \sigma^2)}, \frac{Ba_{11}a_{12}p_1}{\ln 2(a_{12}p_2 + \sigma^2)(a_{11}p_1 + a_{12}p_2 + \sigma^2)}, 1, 0 \right)^T. \tag{2.7}$$

Similarly, $\nabla g_3(p_1, p_2, f_1, f_2)$ can be obtained by swapping all ones and twos in the above equation. And so,

$$\nabla^2 g_2(p_1, p_2, f_1, f_2) = \frac{B\tau}{\ln 2(a_{11}p_1 + a_{12}p_2 + \sigma^2)^2} \begin{pmatrix} a_{11}^2 & a_{11}a_{12} & 0 & 0 \\ a_{11}a_{12} & -\frac{(a_{11}p_1 + 2a_{12}p_2 + 2\sigma^2)a_{11}a_{12}^2 p_1}{(a_{12}p_2 + \sigma^2)^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{2.8}$$

Looking at the leading principle minors of 2.8, we see that

$$\triangle_{11} = \frac{B\tau a_{11}^2}{\ln 2(a_{11}p_1 + a_{12}p_2 + \sigma^2)^2} > 0$$

and

$$\triangle_{22} = -\left(\frac{B\tau}{\ln 2}\right)^2 \left(\frac{(a_{11}p_1 + 2a_{12}p_2 + 2\sigma^2)a_{11}^2 a_{12}^2 p_1}{(a_{11}p_1 + a_{12}p_2 + \sigma^2)^4 (a_{12}p_2 + \sigma^2)^2} - \frac{a_{11}^2 a_{12}^2}{(a_{12}p_2 + \sigma^2)^2}\right) < 0.$$

Thus, by Proposition A.7 in Appendix A, $(\nabla^2 g_2(p_1, p_2, f_1, f_2))$ is nowhere positive semidefinite, and thus by Proposition A.5 in Appendix A, $g_2(p_1, p_2, f_1, f_2)$ is nonconvex. By an identical calculation we come to the conclusion that $g_3(p_1, p_2, f_1, f_2)$ is also nonconvex. This means that the mathematical model described in (2.6) need not have a unique global minimizer. We next look to find conditions on the problem parameters under which this problem has a unique solution

2.5.2 Reducing the Problem to One Variable

We will now present a Theorem which will enable us to reduce the number of variables we need to consider, and prove useful to us through the remainder of this thesis.

Theorem 2.5 *Suppose that the objective function* $\Phi : (\mathbf{p}, \mathbf{b}, \mathbf{f},) \mapsto \Phi(\mathbf{p}, \mathbf{b}, \mathbf{f})$ *is strictly monotone in* $\mathbf{p}$*, and that we want to solve the optimization problem detailed in equation (2.4). Then all constraints (2.3) are active at each local minimizer of this problem.*

Proof. Let $\mathbf{x} = (p, b, f)$ be a feasible point of the problem (2.4) such that not all coupling constraints of the form (2.3) are active. Let $I \subseteq E \times T$ be the set of all edge-timeslot pairs for which the coupling constraint is inactive, that is for $(e, t) \in I$,

$$\sum_{m \in M} f_{e,t}^m < B\tau \log_2 \left(1 + \frac{a_{e,e}p_{e,t}}{\sum_{l=1\ldots E, l \neq e} a_{l,e}p_{l,t} + BN_0}\right) \tag{2.9}$$

If we can show there is a feasible descent direction from this arbitrary point for which not all coupling constraints are active, then we have shown this point is not a local minimizer and therefore any local minimizer must have all coupling constraints active.

Pick any $(e,t) \in I$ and consider the vector $\mathbf{d} = (0, 0, \ldots, 0, -1, 0, \ldots, 0)^T$ where the -1 is in the position corresponding to power variable $p_{e,t}$. Clearly this is a descent direction since the objective function is strictly monotone in $\mathbf{p}$. We now need to check that for suitably small $\alpha \in \mathbb{R}^+$, $\mathbf{x}_{\text{new}} := \mathbf{x} + \beta\mathbf{d}$ is feasible for all $\beta \in [0, \alpha]$.

Clearly $\mathbf{x}_{\text{new}}$ will remain in $\mathbf{C}_{\mathbf{b},\mathbf{f}}$ since we have not altered $\mathbf{b}$ or $\mathbf{f}$. Since we have not increased any power variables, there is no chance we have violated the maximum node power or maximum edge power constraints by moving to $\mathbf{x}_{\text{new}}$.

Since $\sum_{m \in M} f_{e,t}^m \geqslant 0$, by (2.9) we have that $p_{e,t} > 0$ and so $p_{e,t} - \beta \geqslant 0$ for $\beta \leqslant p_{e,t}$ and so for suitably small β we do not violate the non-negative power constraint.

In constraint (2.9), by continuity there exists some $q < p_{e,t}$ such that

$$\sum_{m \in M} f_{e,t}^m = B\tau \log_2 \left(1 + \frac{a_{e,e} q}{\sum_{l=1\ldots E, l \neq e} a_{l,e} p_{l,t} + BN_0} \right).$$

As long as $p_{e,t} - \beta \geqslant q$, constraint (2.9) remains feasible at $\mathbf{x}_{\text{new}}$. For all other coupling constraints, $p_{e,t}$ appears on the denominator in the log term on the right hand side, and so decreasing $p_{e,t}$ increases the right hand side and therefore cannot compromise feasibility. $\mathbf{x}_{\text{new}} = \mathbf{x} + \beta\mathbf{d}$ is therefore feasible $\forall\ \beta \in [0, p_{e,t} - q]$. We have found a feasible descent direction from $\mathbf{x}$ and so $\mathbf{x}$ was not a local minimizer. $\square$

Using this theorem, with our objective which clearly satisfies the condition in the statement of the theorem, we can replace the inequality constraints g_2 and g_3 with equality constraints without losing any local minimizers of the original problem from the new constraint set. Since our objective is to minimize $p_1 + p_2$, we can also ignore the constraint $g_1 = p_1 + p_2 - P_{\max} \leqslant 0$ since if we find that our optimal solution does not satisfy this constraint, then there is no feasible solution to the problem.

The total amount of data we wish to send is S bits, so if we let $f_1 = \lambda$, we need that $f_2 = S - \lambda$, and we can track all possible solutions by letting λ range between zero and S.

Rearranging our new equality coupling constraints we should be able to arrive at $p_1(\lambda)$ and $p_2(\lambda)$, and observe the behaviour of $(p_1 + p_2)(\lambda)$ as λ is allowed to vary from zero to S.

Letting $f_1 = \lambda$ and $f_2 = S - \lambda$, and rearranging the coupling constraints, we have that

$$2^{\frac{\lambda}{B\tau}} - 1 = \frac{a_{11}p_1}{a_{21}p_2 + \sigma^2} \tag{2.10}$$

and

$$2^{\frac{S-\lambda}{B\tau}} - 1 = \frac{a_{22}p_2}{a_{12}p_1 + \sigma^2}. \tag{2.11}$$

For ease of notation we treat $B\tau := \upsilon$ as a single parameter. Rearranging for p_1 in (2.10) and p_2 in (2.11) and then summing we have that

$$\left(1 - (2^{\frac{\lambda}{\upsilon}} - 1)(2^{\frac{S-\lambda}{\upsilon}} - 1)\frac{a_{12}a_{21}}{a_{11}a_{22}}\right)(p_1 + p_2) = \frac{\sigma^2}{a_{11}a_{22}}\left((2^{\frac{\lambda}{\upsilon}} - 1)(2^{\frac{S-\lambda}{\upsilon}} - 1)(a_{12} + a_{21}) + (2^{\frac{\lambda}{\upsilon}} - 1)a_{22} + (2^{\frac{S-\lambda}{\upsilon}} - 1)a_{11}\right)$$

and therefore as long as

$$\left(1 - (2^{\frac{\lambda}{\upsilon}} - 1)(2^{\frac{S-\lambda}{\upsilon}} - 1)\frac{a_{12}a_{21}}{a_{11}a_{22}}\right) \neq 0,$$

we can divide through by it, multiply out all terms and simplify to get that:

$$(p_1{+}p_2)(\lambda) = \sigma^2\frac{(a_{22} - a_{12} - a_{21})2^{2\frac{\lambda}{\upsilon}} + ((2^{\frac{S}{\upsilon}} + 1)(a_{12} + a_{21}) - a_{11} - a_{22})2^{\frac{\lambda}{\upsilon}} + 2^{\frac{S}{\upsilon}}(a_{11} - a_{12} - a_{21})}{a_{12}a_{21}2^{2\frac{\lambda}{\upsilon}} + (a_{11}a_{22} - (2^{\frac{S}{\upsilon}} + 1)a_{12}a_{21})2^{\frac{\lambda}{\upsilon}} + 2^{\frac{S}{\upsilon}}a_{12}a_{21}}.$$

We would now like to be able to describe the behaviour of this function depending on the values taken by our seven parameters, $a_{11}, a_{12}, a_{21}, a_{22}, \sigma^2$, S and υ.

Lemma 2.6 *The convexity of this problem is independent of the value of the noise vari-*

able σ^2.

Proof. By [6], $f(x)$ is convex, if and only if $\alpha f(x)$ is convex for $\alpha > 0$. □

2.5.3 The Behaviour of a Certain Function

In order to understand the behaviour of our function $(p_1 + p_2)(\lambda)$ it is easier to look at a slightly tidier function. Namely, we shall consider functions of the form

$$Q(x) = \frac{a2^{2x} + b2^x + \gamma a}{d2^{2x} + e2^x + \gamma d} \tag{2.12}$$

for $Q : \mathbb{R}_+ :\rightarrow \mathbb{R}$, $a, b, c, d \in \mathbb{R}$, and $\gamma \in \mathbb{R}_+$. To understand the behaviour of this function, we calculate its first derivative:

$$Q'(x) = \frac{\ln 2(ae - bd)2^x}{(d2^{2x} + e2^x + \gamma d)^2}(2^{2x} - \gamma) \tag{2.13}$$

Calculating the second derivative, it becomes apparent that the sign is in no way clear. In this situation, it will be easier to find information about the shape of the function by looking at the function value itself and the first derivative. Indeed, several interesting observations can be made concerning the shape of the function and location of its stationary points.

Lemma 2.7 *If* $ae = bd$, $Q(x)$ *is a constant, otherwise* $Q(x)$ *has exactly one stationary point when* $x = \frac{\log_2(\gamma)}{2}$.

Proof. By equation (2.13), $Q'(x) = 0$ if $ae - bd = 0$, or $2^{2x} - \gamma = 0$. If $ae - bd = 0$, $Q'(x) = 0 \; \forall \; x$, therefore $Q(x) = K$ for some $K \in \mathbb{R}$. If $2^{2x} - \gamma = 0$, we can solve for x to obtain that $x = \frac{\log_2(\gamma)}{2}$. □

Lemma 2.8 $Q(x) = Q(\log_2(\gamma) - x) \; \forall \; x \in \mathbb{R}_+$.

Proof.

$$Q(\log_2(\gamma)-x)=\frac{a2^{2(\log_2(\gamma)-x)}+b2^{(log_2(\gamma)-x)}+\gamma a}{d2^{2(\log_2(\gamma)-x)}+e2^{(log_2(\gamma)-x)}+\gamma d}=\frac{\gamma^2 a2^{-2x}+\gamma b2^{-x}+\gamma a}{\gamma^2 d2^{-2x}+\gamma e2^{-x}+\gamma d}.$$

We can now multiply top and bottom by 2^{2x} and divide top and bottom by γ to get

$$Q(\log_2(\gamma)-x)=\frac{\gamma a+b2^x+a2^{2x}}{\gamma d+e2^x+d2^{2x}}=Q(x). \qquad \square$$

This Lemma shows us that the function is symmetrical about the point $x=\frac{\log_2(\gamma)}{2}$.

Although we have no second derivative information, the above lemmas give us plenty of information concerning the shape of $Q(x)$.

Corollary 2.9 *Let $\gamma > 0$.Consider the function $Q : [0, \log_2(\gamma)] \to \mathbb{R}$, where $Q(x) = \frac{a2^{2x}+b2^x+\gamma a}{d2^{2x}+e2^x+\gamma d}$, then one of three situations can occur.*

1. *If $ae - bd = 0$, then $Q(x) = \frac{(1+\gamma)a+b}{(1+\gamma)d+e}$ $\forall x \in [0, \log_2(\gamma)]$.*
2. *If $ae - bd \neq 0$ and $Q(\frac{\log_2(\gamma)}{2}) > Q(0)$, then the set of global minima is $\{0, \log_2(\gamma)\}$.*
3. *If $ae - bd \neq 0$ and $Q(\frac{\log_2(\gamma)}{2}) < Q(0)$, then the global minimum is $\frac{\log_2(\gamma)}{2}$.*

Proof. Follows almost immediately from Lemmas 2.7 and 2.8 when considering that the function Q is either constant or has one stationary point halfway along the interval. Since the function value is the same at either end of the interval, the midpoint can be either the unique minimum, or the unique maximum, or share the same value with every other point. $\square$

A potentially easier way to characterise the solution set is to check the first derivative at the origin.

Corollary 2.10 *1. If $Q'(0) = 0$, the set of global minimizers is $[0, \log_2(\gamma)]$*

2. *If* $Q'(0) > 0$, *the set of global minimizers is* $\{0, \log_2(\gamma)\}$.

3. *If* $Q'(0) < 0$, *the global minimizer is* $\frac{\log_2(\gamma)}{2}$.

Proof. By the fact that there is only one stationary point from Lemma 2.7 and the symmetry of Q from Lemma 2.8, if the derivative is positive at the origin, then the stationary point is a maximizer, if it is negative the stationary point is a minimizer. If the derivative is zero at the origin, then the function is constant. □

Now, since

$$Q'(0) = \frac{\ln(2)(ae - bd)}{(d + e + \gamma d)^2}(1 - \gamma)$$

we see that the minimizers of Q can be determined by the signs of $(ae - bd)$, and $(1 - \gamma)$.

2.5.4 Location of Global Minima for our Simple Problem

The first question to ask is, under what conditions on the variables a_{11}, a_{12}, a_{21} and a_{22}, υ and S can we use the results of the previous section? Let us compare our two functions:

$$(p_1+p_2)(\lambda) = \sigma^2 \frac{(a_{22} - a_{12} - a_{21})2^{2\frac{\lambda}{\upsilon}} + ((2^{\frac{S}{\upsilon}} + 1)(a_{12} + a_{21}) - a_{11} - a_{22})2^{\frac{\lambda}{\upsilon}} + 2^{\frac{S}{\upsilon}}(a_{11} - a_{12} - a_{21})}{a_{12}a_{21}2^{2\frac{\lambda}{\upsilon}} + (a_{11}a_{22} - (2^{\frac{S}{\upsilon}} + 1)a_{12}a_{21})2^{\frac{\lambda}{\upsilon}} + 2^{\frac{S}{\upsilon}}a_{12}a_{21}}. \tag{2.14}$$

$$\lambda \in [0, S].$$

$$Q(x) = \frac{a2^{2x} + b2^x + \gamma a}{d2^{2x} + e2^x + \gamma d} \quad x \in [0, \log_2(\gamma)]$$

The first thing to notice is that if a_{11}=a_{22}, then equation (2.14) is in the required form, with

- $a = \sigma^2(a_{11} - a_{12} - a_{21})$
- $b = \sigma^2((2^{\frac{S}{\upsilon}} + 1)(a_{12} + a_{21}) - 2a_{11})$.

- $d = a_{12}a_{21}$
- $e = a_{11}^2 - (2^{\frac{S}{\upsilon}} + 1)(a_{12}a_{21})$.
- $x = \frac{\lambda}{\upsilon}$.
- $\gamma = 2^{\frac{S}{\upsilon}}$.

Notice that the two intervals being considered are also identical since

$$x \in [0, log_2\gamma] \Leftrightarrow \frac{\lambda}{\upsilon} \in [0, \frac{S}{\upsilon}] \Leftrightarrow \lambda \in [0, S].$$

With this in mind, we can use the last lemma of the previous section to find the minima of $(p_1 + p_2)(\lambda)$ under various conditions on the parameters. Now,

$$\begin{aligned} ae - bd &= \sigma^2((a_{11} - a_{12} - a_{21})(a_{11}^2 - (2^{\frac{S}{\upsilon}} + 1)a_{12}a_{21}) - \\ &\quad a_{12}a_{21}((2^{\frac{S}{\upsilon}} + 1)a_{12} + 3a_{21} - 2a_{11})) \\ &= \sigma^2 a_{11}(a_{11}^2 - a_{11}a_{12} - a_{11}a_{21} - a_{12}a_{21}(1 - 2^{\frac{S}{\upsilon}})). \\ (1 - \gamma) &= 1 - 2^{\frac{S}{\upsilon}}. \end{aligned}$$

Now, since $2^{\frac{S}{\upsilon}} > 1$, and so $(1 - \gamma) < 0$ for all feasible choices of S, B and τ, we only have one equation left to consider in determining the solution set for this simple problem under the condition that $a_{11} = a_{22}$.

2.5.5 The Symmetrical Case

It is interesting to look at a completely symmetrical version of this problem whereby $a_{11} = a_{22}$, and $a_{12} = a_{21}$. With this symmetry we reduce the number of parameters and are able to give a line which is the border between the case where we have a unique solution and the case where we have two solutions. This line is the line along which we

have infinitely many solutions. In this symmetrical case, we have that

$$ae - bd = \sigma^2 a_{11}(a_{11}^2 - 2a_{11}a_{12} + a_{12}^2(1 - 2^{\frac{S}{v}})).$$

Let us find the relationship between a_{11} and a_{12} under which $ae - bd$ is greater than, equal to or less than zero. Since it was assumed that $a_{11} > 0$, we have that:

$$\begin{aligned} ae - bd = 0 \Leftrightarrow \quad & (a_{11}^2 - 2a_{11}a_{12} + a_{12}^2(1 - 2^{\frac{S}{v}})) = 0 \\ \Leftrightarrow \quad & (a_{11} - a_{12})^2 + a_{12}^2(1 - 2^{\frac{S}{v}}) - a_{12}^2 = 0 \\ \Leftrightarrow \quad & (a_{11} - a_{12})^2 = a_{12}^2 2^{\frac{C}{v}} \\ \Leftrightarrow \quad & a_{11} = (1 \pm 2^{\frac{S}{2v}})a_{12}. \end{aligned}$$

Since we assume that both a_{11} and a_{12} are non-negative, and $2^{\frac{S}{2v}} \geqslant 1$ we can reject the case where $a_{11} = (1 - 2^{\frac{S}{2v}})a_{12}$, so we have that $ae - bd = 0 \Leftrightarrow a_{11} = (1 + 2^{\frac{S}{2v}})a_{12}$. Following through the above argument with inequalities, we see that $ae - bd < 0 \Leftrightarrow a_{11} < (1 + 2^{\frac{S}{2v}})a_{12}$, and $ae - bd > 0 \Leftrightarrow a_{11} > (1 + 2^{\frac{S}{2v}})a_{12}$.

Theorem 2.11 *For the symmetric problem, if $a_{11} > (1 + 2^{\frac{S}{2v}})a_{12}$, the unique global minimum is attained at $\lambda = \frac{S}{2}$. If $a_{11} < (1 + 2^{\frac{S}{2v}})a_{12}$, the set of global minimizers is $\{0, S\}$. If $a_{11} = (1 + 2^{\frac{S}{2v}})a_{12}$, the set of global minimizers is $[0, S]$.*

Proof. Evident. □

This result provides a valuable insight into the way different parameter combinations can affect the set of minimizers of the problem. This theorem confirms the natural intuition that if sending messages along the two edges will cause them to interfere with each other due to a high a_{12} value, it is easier just to send all the message along one edge, whereas if there is a limited amount of interference, represented by a large difference between a_{11} and a_{12} it makes sense to split the message up and send half along each

edge. The larger the message, the larger the difference between a_{11} and a_{12} is required to be before it is worthwhile splitting the message. Similarly, the greater the available bandwidth or timeslot, the smaller the ratio between a_{11} and a_{12} is required to be before the multipath option becomes worthwhile.

2.5.6 The Most General Case of the Simple Problem

If $a_{11} \neq a_{22}$, we cannot use any of the results from section 2.5.3, and the best we can do is to say whether or not the function is convex, based on the function's second derivative. As we have seen previously, this is not a pleasant expression to evaluate. Numerically, one can see whether or not the second derivative remains non-negative on the interval $[0, S]$, and if it does we can say that the function has a unique global minimizer. We plotted p_1, p_2 and $(p_1 + p_2)(\lambda)$ for several different parameter combinations to investigate the behaviour of the problem, and observed four qualitatively different possibilities. These four behaviours can all be seen by keeping S, υ and σ^2 fixed and varying the values in the gain matrix. In the following four examples we set $\upsilon = S = 10$ and $\sigma^2 = 1$.

- The problem has a unique global minimizer. For example setting $a_{11} = a_{22} = 1$, $a_{12} = a_{21} = 0.1$,see top left of Figure 2.3.

- The problem has two global minimizers. For example setting $a_{11} = a_{22} = 1$, $a_{12} = a_{21} = 1$, see top right of Figure 2.3.

- The problem has a local minimizer and a global minimizer. For example setting $a_{11} = 1$, $a_{12} = \sqrt{2} - 1$, $a_{21} = 1$ $a_{22} = \sqrt{2} - 1$, see bottom left of Figure 2.3.

- The problem has infinitely many global minimizers. For example setting, $a_{11} = a_{22} = 1$, $a_{12} = a_{21} = \sqrt{(2)} - 1$, see bottom right of Figure 2.3.

Having seen the effect of adjusting the gain matrix, it is also interesting to see how altering the ratio between S and υ changes the behaviour of the problem. For this, we keep

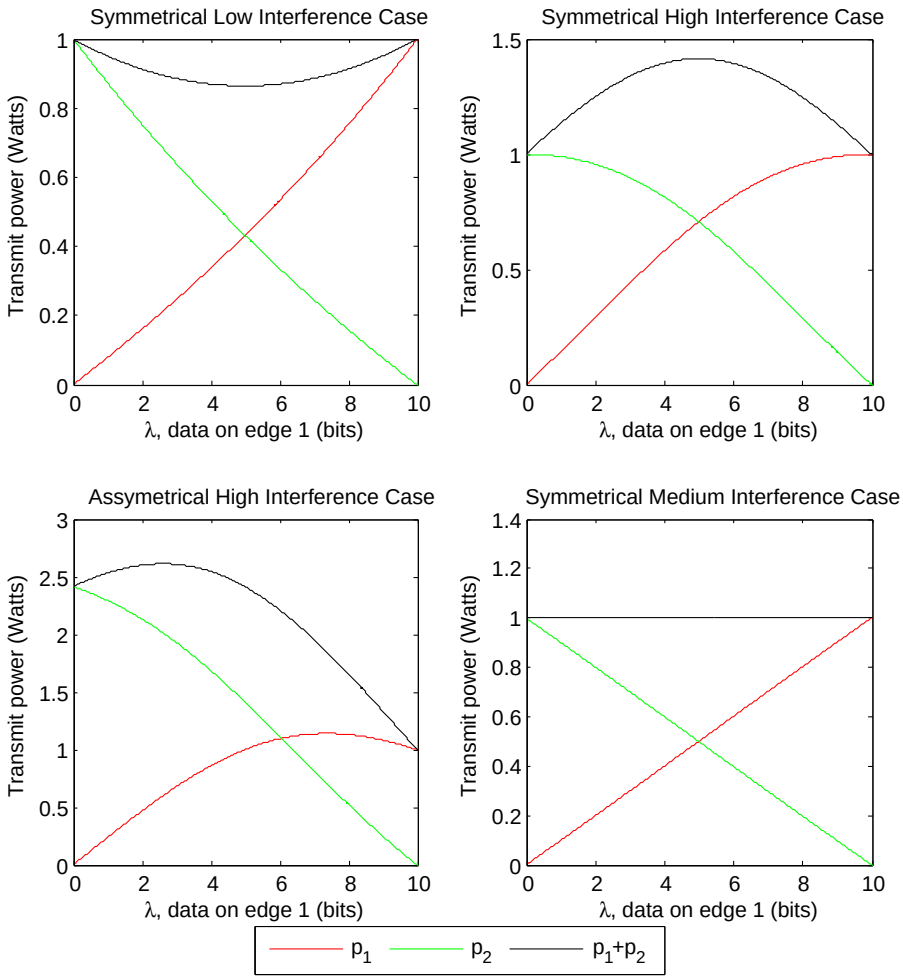

Figure 2.3: *Graph of λ against transmit power in the four qualitatively different situations*

the gain matrix fixed at $a_{11} = a_{22} = 1$, $a_{12} = a_{21} = 0.1$. We fix $\sigma^2 = 1$ and $\upsilon = 10$, and vary the message size S. From the previous section we would expect a unique solution for $S < 2\upsilon \log_2(\frac{a_{11}}{a_{12}} - 1) = 63.398$, infinitely many solutions when $S = 63.398$, and solutions at the boundary for $S > 63.398$. In Figure 2.4 we see exactly this behaviour.

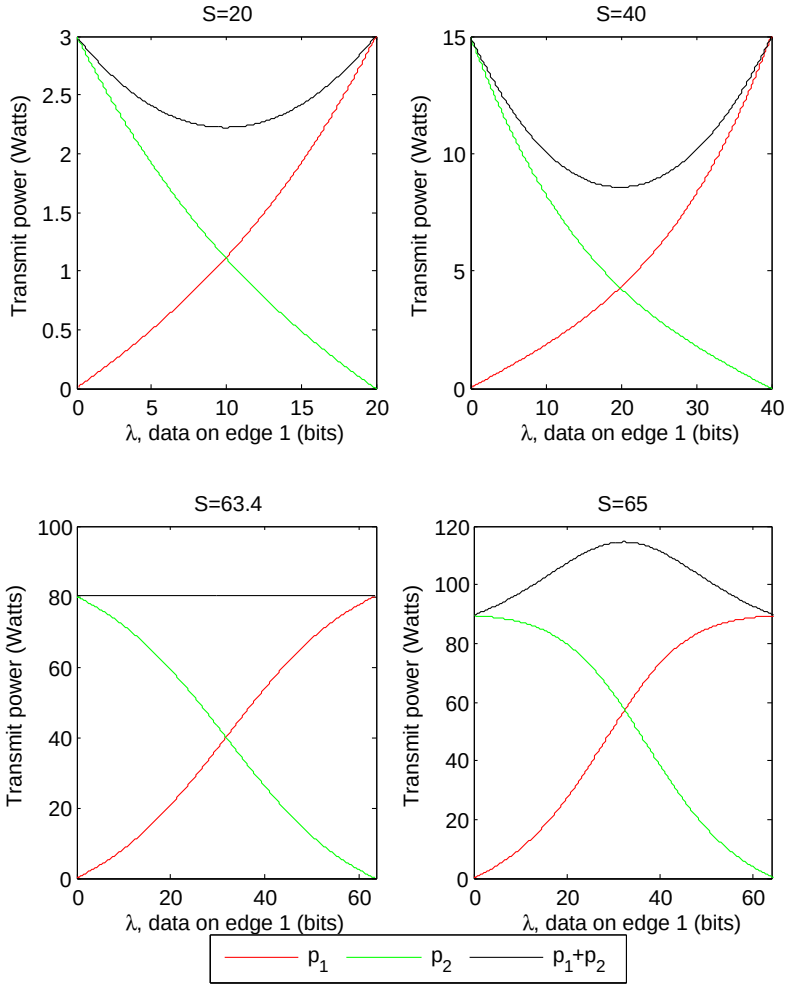

Figure 2.4: *Graph of λ against transmit power for increasing message size S*

In this chapter we have introduced a mathematical framework and nonlinear optimization problem to model the transmission of unicast messages through a wireless multihop network using multipath routing. Our optimization problem seeks solutions which are optimal over both communication resource variables and network flow variables.

By studying the simplest possible network which suffers from cross-channel interference we found conditions on the problem parameters which lead to the problem having a unique solution. Although unable to expand this result to a more general network, we still gain valuable insight into the effects changes of parameter values have on the solutions to the problem.

In the next chapter we will expand our model to enable it to handle a more general class of message transmissions.

Chapter 3

Expanding the Model to Handle Multicasts

The model we detailed in the previous chapter was built to handle the transmission of data through a wireless network from a single source node to a single receiver node. A generalization of this is to consider transmitting the same piece of data from a single source to a set of receiver nodes.

Definition 3.1 *Consider a set of V communication nodes, with $s \in V$ a source node and $D \subset V$ a set of receiver nodes. We assume $s \notin D$.*

If D consists of exactly one node, we term this transmission a ***unicast****.*

If $D \cup s = N$, we term this transmission a ***broadcast****.*

Otherwise, we term the transmission a ***multicast****.*

In this chapter we investigate ways in which we can expand our model to handle multicasts, and since broadcasts and unicasts are special cases of multicasts, our model will still be able to handle those too.

One major difference between unicasts and multicasts is that in multicasts, it is natural to allow intermediate nodes to store and replicate incoming data packets, that is to say a single packet received by an intermediate node can be transmitted along many different

outgoing edges or even in different timeslots in order to send it to the various multicast destinations. This means that the flow conservation laws fundamental to the model developed in the previous chapter need not necessarily still apply.

There are three main approaches to solving the multicasting problem we wish to consider, each with positive and negative attributes to weigh up. The first approach we will term the "naive" approach, and here we basically ignore the fact that we are sending the same data to multiple destinations, and treat a multicast as a set of simultaneous unicasts with the same source node but different destination nodes.

The second approach relies on ideas from graph theory and is the most common method for considering multicasts in networks. The idea is to consider a tree rooted at the source node, and with leaves at the destination nodes, and then linearly combine weighted combinations of these trees to make the best use of the available network resources.

With this approach we do not disregard the fact that the destinations all require the same data and so should get better solutions than those available through the naive approach. The problem with this tree finding approach is the need to formulate the problem as a combinatorial optimization problem, and it is therefore computationally very expensive as the size of the graph grows. It can also be shown that for a network with fixed capacity, this approach cannot necessarily find a maximal flow, suggesting it may also not find the best possible solutions to our problem.

The third approach is based on a recent idea known as network coding. Here, intermediate nodes are given the ability to perform elementary operations on incoming data packets. Unlike the tree finding approach, this method requires no combinatorial optimization, and can also be shown to find maximal multicast flows in networks of fixed capacity. All this suggests that it should be a promising technique for us.

3.1 The Naive Approach

The simplest way to treat a multicast is to consider it as a set of independent unicasts. In this way, no adaptation of our model is required: Suppose we have a network of nodes $V = v_1, \ldots, v_k$ and wish to have a unicast from v_i to v_j, and a multicast from v_m to $v_{p_1}, \ldots, v_{p_q}$. Then, by this naive approach we just treat this as $q+1$ unicasts and can feed it immediately into our optimization framework of the previous chapter.

Clearly for some networks this approach will work optimally, or near optimally, whereas for others it can be shown to perform arbitrarily badly. One can illustrate both cases by considering a very simple graph.

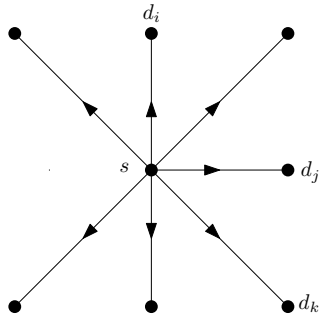

Figure 3.1: *An example of a network where the naive approach to multicasting is also the best approach.*

Definition 3.2 *A **star** is a graph where every edge shares a common node. This common node we will call the **central node**, all other nodes will be referred to as **non-central**.*

Consider a star on k nodes, and suppose we wish to multicast from the central node to any subset of the other nodes, see Figure 3.1. Clearly there is no choice but to send the same data along each edge between the source and a destination, and so treating the multicast as independent unicasts is harmless and indeed optimal.

Suppose now that the source is one of the non-central nodes, and the destinations are a set of j other non-central nodes, see Figure 3.2. For illustrative purposes we also

assume we are limited to two timeslots. By treating the multicast as a set of independent unicasts, we will have to send the same data j times along the edge from the source to the central node, all in the same timeslot.

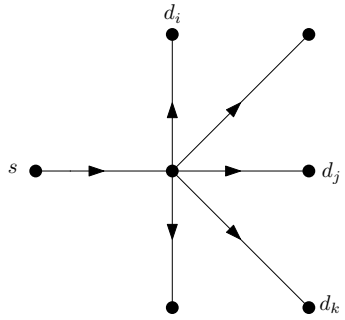

Figure 3.2: *An example of a network where the naive approach to multicasting is clearly suboptimal*

Assuming zero interference and identical channel conditions on each link, and setting all parameters to one, we see the power required for the first timeslot satisfies:

$$jM = \log_2(1 + p)$$

or

$$p = 2^{jM} - 1.$$

The power required along each edge used in the second time slot satisfies

$$M = \log_2(1 + p)$$

or

$$p = 2^M - 1.$$

The total power used in the naive case is therefore:

$$p_{\text{naive}} = j2^M + 2^{jM} - j - 1.$$

In such a simple case, one can see that the optimal multicast solution is to send the data M only once along the edge from the source to the central node, and then in the second timeslot to send the data along all edges to destination nodes. The total power required is thus

$$p_{\text{opt}} = (j+1)2^M - j - 1$$

and so

$$p_{\text{naive}} - p_{\text{opt}} = 2^{jM} - 2^M$$

which clearly tends to infinity as j increases. Further, the relative difference in solutions is:

$$\frac{p_{\text{naive}} - p_{\text{opt}}}{p_{\text{opt}}} = \frac{2^{jM} - 2^M}{(j+1)(2^M - 1)}$$

which also tends to infinity as j increases. We therefore see that in networks with a "trunk" edge, along which multicast data for multiple destinations needs to travel, this naive approach provides bad solutions. Figure 3.3 shows how it is easy to construct examples where this naive approach performs arbitrarily badly.

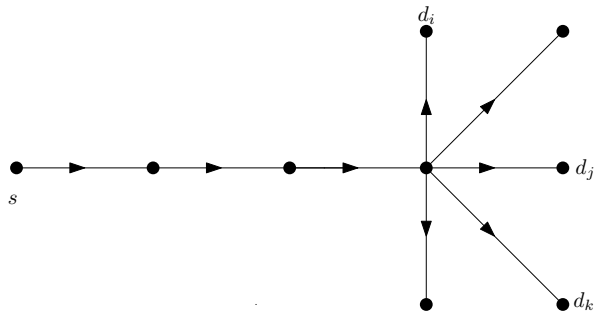

Figure 3.3: *An example of how we can construct a network in which the naive approach performs arbitrarily badly.*

3.2 Tree Finding Approach

To fully explain this approach we will require some basic concepts from graph theory.

Definition 3.3 *Consider a graph or digraph. A* ***tree*** *on the graph is a subgraph containing no cycles. A* ***spanning tree*** *is a tree which includes all the vertices of the underlying graph. A* ***Steiner tree*** *is a tree which includes a particular subset of the vertex set.*

Clearly, if we find a Steiner tree where the vertices we insist on including are the source vertex and destination vertices of our multicast problem, and each edge of the Steiner tree has capacity M, then by sending data along all the edges of the tree, we have found a feasible multicast. One can use multiple Steiner trees to achieve a higher throughput, or in our power minimization problem, to offer a larger set of feasible multicast routing solutions to choose an optimal solution from. Obviously, we can only use multiple trees if the total capacity requirements of all the trees stacked on top of each other does not violate the capacity of the underlying graph.

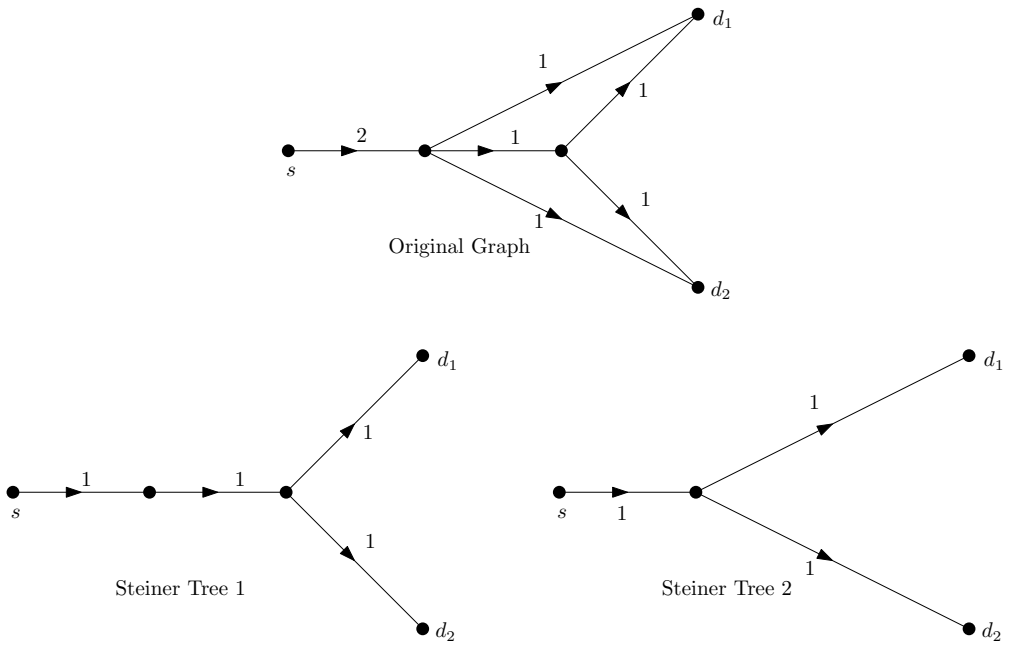

Figure 3.4: *An example of achieving maximal multicast throughput by packing Steiner trees.*

For example, Figure 3.4 shows how we can pack two trees into a graph to achieve a multicast throughput of 2. If however the left most edge in the original graph in Figure

3.4 had unit capacity, we could achieve a multicast throughput of no more than 1. The problem of optimally packing Steiner trees to find maximum flow is NP-hard. Since our problem is very much related to the problem of achieving maximum throughput in a fixed capacity graph (for a fixed power configuration our network has fixed capacity) as well as the problem of finding a minimum cost flow (in our case the costs are non-linear functions of the data to be sent), it is worth noting this and other results relating to these problems.

For unicast it is a well established classical result, the "max-flow min-cut theorem" that the maximum unicast flow achievable equals the minimum total weight of edges which needs to be removed from the network to mean the source is disconnected from the destination, see for example [38].

Definition 3.4 *We define the* ***minimum cut*** *or* ***mincut*** *between two vertices of a graph as follows:*

$$\text{mincut}(u,v) = \left\{ \min \sum_{e \in C} w_e \mid u \text{ is disconnected from v in } G \backslash C \right\}$$

Then for unicast we have $\text{maxflow}(u,v) = \text{mincut}(u,v)$. For broadcast or multicast on a fixed capacity network clearly an upper bound on the achievable broadcast rate is the maximum rate at which the source could hold a unicast with every destination independently.

Definition 3.5 *Let $R(u,D)$ be the* ***maximum achievable transmission rate*** *when multicasting from source u to destination set D in a fixed capacity network.*

Then $R(u,D) \leqslant \min_{d \in D} \text{mincut}(u,d)$ since if a given rate is unachievable as a unicast between the source and a given destination, it certainly is not possible as part of a multicast.

By Edmond's theorem of 1972, [13], this rate is actually achievable for all broadcasts by packing Steiner trees.

$$R(u, V\backslash u) = \min_{v \in V\backslash u} \text{mincut}(u, v).$$

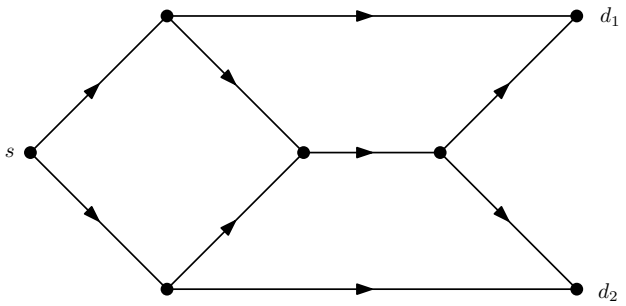

Figure 3.5: *The butterfly graph, a network where Steiner tree packing cannot achieve the theoretical maximum multicast flow*

For a general multicast the bound cannot in general be met by packing Steiner trees. For example, consider the graph in Figure 3.5 where each edge has unit capacity. Now $\text{mincut}(s, d_1) = \text{mincut}(s, d_2) = 2$ but since none of the Steiner trees of (s, d_1, d_2) are edge disjoint, as is shown in Figure 3.6 we see that a multicast flow of rate two cannot be achieved. In fact we can achieve a flow of rate 1.5 by packing one half of each of Steiner trees 1, 6 and 7 in Figure 3.6.

The upshot of all this theory is that finding optimal Steiner tree solutions is computationally expensive since the problem is NP-complete, and even if we go to the effort of finding a solution it will not necessarily achieve the min-mincut bound. Additionally, it is not immediately clear how we would incorporate Steiner tree solutions into our existing unicast optimization framework and how the solutions could be implemented in a distributed manner. We therefore seek an alternative method through the recently established approach of network coding.

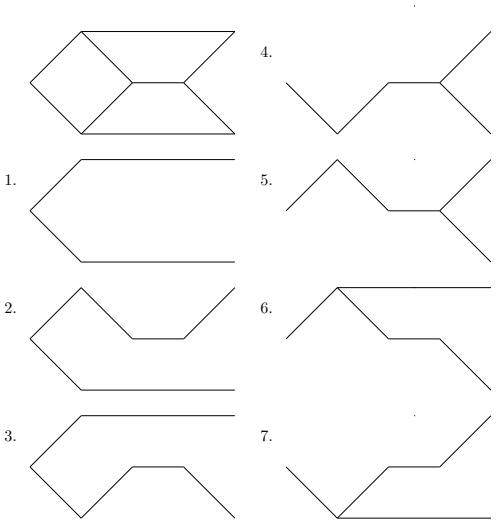

Figure 3.6: *All possible Steiner trees which include nodes* s, d_1 *and* d_2 *of the butterfly graph*

3.3 Motivating Network Coding

The third possible approach for incorporating multicast transmissions in our model is to use Network Coding.

In traditional routing, intermediate nodes are only capable of storing, forwarding and replicating any packets they receive. The principle of Network Coding is to open a whole new set of operations at each node. As well as storing, forwarding and replicating, the nodes are given the ability to combine incoming packets, thus packets on a nodes outgoing edges can be formed of a mixture of packets that previously arrived on incoming edges. Rather than the original required packets, a receiving node is presented with a set of recombined packets of information as well as the information required for retrieving the original information.

The idea of Network coding is relatively new one, first suggested in 2000 by Ahlswede et al, [1]. Over recent years, much research has focussed on ways in which Network coding could be employed to improve many aspects of information networks, including improving throughput, reducing delays, and improving robustness. Good introductions to the theory

can be found in [18] for the general setting, and [12] and [10] for specific consideration of Network Coding in wireless environments.

Before going into the technical detail of how network coding would be utilized for multicasting in wireless networks, we present some motivating examples of how we can expect network coding to be beneficial when compared to standard routing.

In all the simple examples in this section, we wish to send information packets A and B, where $A = 101\ldots0111$ and $B = 001\ldots1100$ are binary strings. By summing A and B modulo 2, we get $A + B = 100\ldots1011$. The main principle in use is that if a destination node knows the contents of packets A and $A+B$, where $A = 100\ldots0111$ and $A + B = 001\ldots1100$ are binary strings, then by the principles of modular arithmetic, the destination is able to recover the contents of packet B, since $B = (A + B) - A$, or equivalently, $B = (A + B) + A$. All of the examples presented are well known in the literature, and specifically discussed in [10]. We feel, however, they are illuminating enough to include in the thesis.

3.3.1 Increased Throughput in Multicasts and Multiple Unicasts

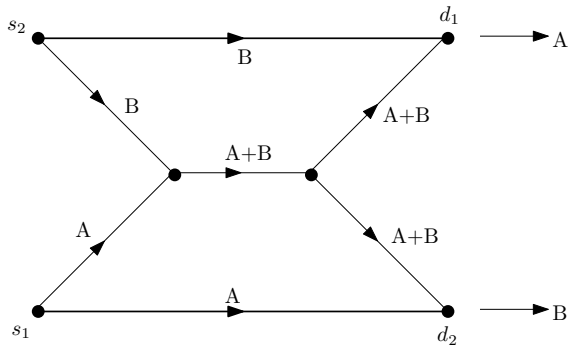

Figure 3.7: *Example of how network coding can be used to increase total throughput in a network carrying multiple unicast sessions*

A standard example already touched upon in the previous section of how network

coding can be utilized is the "butterfly graph", see Figures 3.7 and 3.8, which can be used to illustrate the throughput benefits network coding has over Steiner-tree based routing for both multicast and multiple unicast transmissions. Firstly, suppose s_1 wishes to send data to d_1, and s_2 wishes to send to d_2 in Figure 3.7, with unit capacity available on each link.

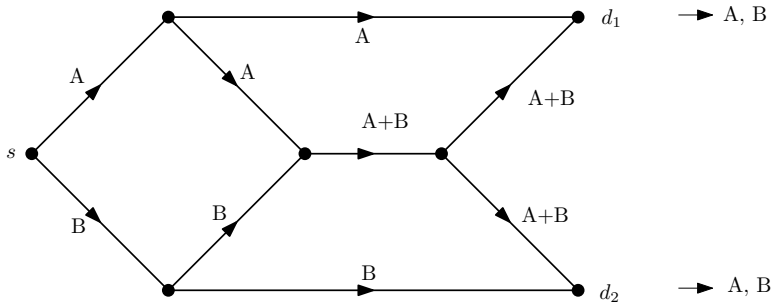

Figure 3.8: *Example of how network coding can be used to increase total throughput in a network carrying a single multicast session.*

With standard routing, the bottleneck middle link has to be shared by both communications, and so the total throughput available to senders s_1 and s_2 is limited to 1, and so in a fair system both sessions could achieve a rate of 1/2.

With network coding, we send a linear combination of the two messages down the bottleneck link. This linear combination can then be used by the two end links in combination with the messages sent down the side links to recover the required data. Specifically, d_1, requiring packet A receives packets $A + B$ and B, and then adds them to retrieve packet A. Similarly receiver d_2, requiring packet B receives packets $A + B$ and A, and then adds them to retrieve packet B. The individual throughputs of users s_1 and s_2 are now both limited separately by 1, and so the total throughput available is 2.

In the multicast case, see Figure 3.8, suppose s_1 wishes to send the same data to destinations d_1 and d_2. Using standard routing and spanning trees, it can be shown that the maximum rate that both destinations can receive together is 1.5, see Section 3.2, but with network coding, using the same principle as described above, both destinations can

enjoy a rate of 2, since both are able to retrieve packets A and B.

This simple example shows that situations exist where network coding is able to increase throughput when compared to routing in both multicast and multiple-unicast applications. In fact, it can be shown that using network coding, the min-mincut bound can always be achieved, which is not the case for Steiner tree based routing.

3.3.2 Decreasing Energy Consumption

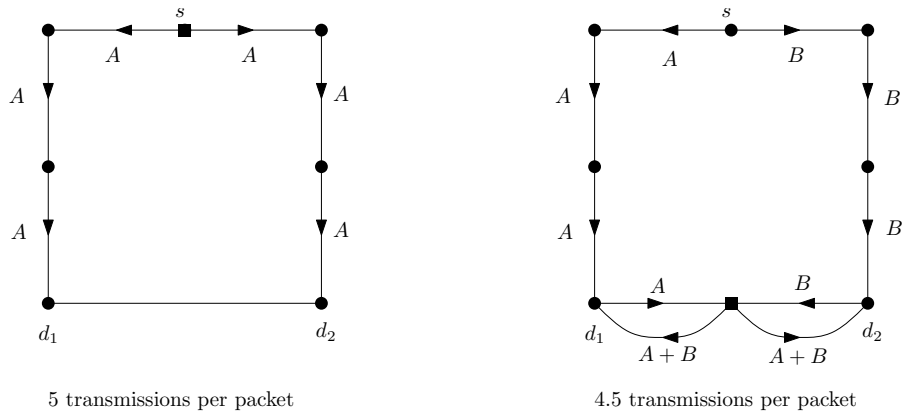

Figure 3.9: *Example of how network coding can be used to decrease energy required per packet sent in a network carrying a single multicast session.*

In [10], Chou and Wu show that, in a model of a wireless network where broadcast advantage is considered, but the energy required for transmission to either one or two destinations is considered constant, multicast can be achieved with less energy per packet sent if network coding is allowed as can be seen in Figure 3.9. If network coding is not used, the same packet is transmitted from the source to the top corner nodes at a cost of 1 unit. The 4 transmissions required down the sides of the square take the total cost to 5 units in order for both destinations to receive the one packet. With network coding 3 transmissions are required for d_1 to receive packet A, and 3 transmissions are required for d_2 to receive packet B. The bottom middle node then receives A and B at a cost of 2 further transmissions. $A + B$ is then broadcast back to d_1 and d_2 at a cost of 1

transmission. A total of 9 transmissions is thus required for both d_1 and d_2 to receive two packets. The cost per packet is therefore 4.5 units, a saving of 10% over routing.

The example considered is a very simple square network and provides encouragement that the benefits of network coding are not limited to carefully constructed academic examples. Since we do not currently consider broadcast advantage in our model, there is no difference between the minimal energy Steiner tree solution and the minimal energy network coding solution for this simple network.

3.3.3 Decreasing Delay

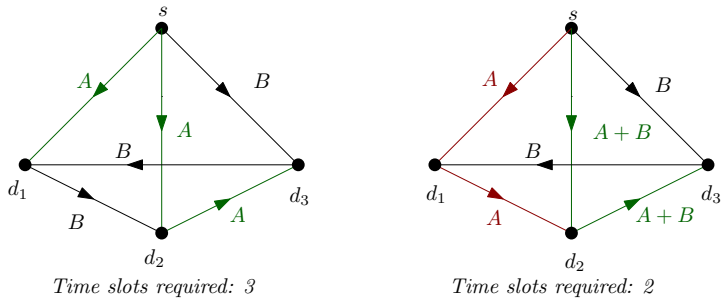

Figure 3.10: *Example of how network coding can be used to decrease total delay in a network carrying a single broadcast session.*

Network coding can also be used to decrease the time between the start of a multicast transmission and the time when all receivers have all the information. This can be seen by considering Figure 3.10. Here, we are broadcasting, since the destination set is every node in the network other than the source. Notice that $\text{mincut}(s, d_1) = \text{mincut}(s, d_2) = \text{mincut}(s, d_3) = 2$ and so by Edmond's Theorem, a Steiner tree routing solution with rate 2 can be achieved. This is done by packing two edge disjoint Steiner trees. As is shown the left hand network of Figure 3.10, the tree carrying packet B will take three timeslots for the packet to reach destination d_2. By using network coding, three paths of length two are used and so all receivers can have all necessary information in two time slots, as can be seen in the right hand network of Figure 3.10. Since our model requires the

transmission of the entire message within a fixed number of time slots, this would also seem likely to be a beneficial property of network coding when applied to our problem. Having motivated the potential benefits of network coding to us, in the forthcoming pages we seek a deeper understanding of the technicalities involved.

3.4 Technical Aspects of Network Coding

The above examples illustrate some of the potential benefits to be found through the use of Network Coding. Alone, these examples are not sufficient to fully understand how network coding can be performed. We will now go into more detail on how encoding and decoding can be performed in practice.

3.4.1 Encoding

Suppose a number of packets, $M^1, \ldots, M^n$, each containing L bits are being fed into a network. We break down each packet into $L/s = k$ strings of length s. Each string of length s can be seen as a symbol over the finite field $\mathbb{F}_{2^s}$. Network coding involves combining these symbols in various ways. We shall focus on linear network coding, since it is well studied, and is sufficiently flexible for the application of multicasting, as we shall see later.

With linear network coding, the outgoing packets from a node are formed of linear combinations of the incoming packets, where addition and multiplication are formed over $\mathbb{F}_{2^s}$. (The mechanics of arithmetic over finite fields is discussed in Section 3.4.3).

It is worth explicitly mentioning that by adding two packets together we do *not* mean concatenation. A linear combination of packets of length L will be a packet of length L. Subsequently, each packet contains only a part of the information from each original packet, and in some sense we are spreading out the information from the various packets.

Any packet, X, in the network is therefore a linear combination of the original n

packets, $X = \sum_{i=1}^{n} g_i M^i$, where $g_i \in \mathbb{F}_{2^s}$. These linear combinations are performed element wise for each symbol in each packet, that is to say if $M^i = (M_1^i, \ldots, M_k^i)$, then the individual symbols $X_1, \ldots, X_k$ of the encoded packet are calculated as$X_j = \sum_{i=1}^{n} g_i M_j^i$ for $j = 1 \ldots k$. Each encoded packet is thus associated with a coding vector, $(g_1, \ldots, g_n) \in \mathbb{F}_{2^s}^n$. As will be shown later, a set of encoded packets along with their encoding vectors can be used to decode and obtain the original packets.

Suppose incoming packets to a node, $X^1, \ldots, X^m$ are already encoded, with corresponding encoding vectors $g^1, \ldots, g^m$, where $g^j = (g_1^j, \ldots, g_n^j)$. An outgoing packet, Y, from this node would therefore be a linear combination of linear combinations of the original packets, which is of course again a linear combination of the original packets. Letting $h_j \in \mathbb{F}_{2^s}$ be the encoding coefficients for Y with respect to the X packets:

$$\begin{aligned}
Y &= \textstyle\sum_{j=1}^{m} h_j X^j \\
&= \textstyle\sum_{j=1}^{m} h_j \sum_{i=1}^{n} g_i^j M^i \\
&= h_1(g_1^1 M^1 + g_2^1 M^2 + \ldots g_n^1 M^n) \\
&\quad + h_2(g_1^2 M^1 + g_2^2 M^2 + \ldots + g_n^2 M^n) \\
&\quad + \ldots + h_m(g_1^m M^1 + g_2^m M^2 + \ldots g_n^m M^n) \\
&= \left(\textstyle\sum_{j=1}^{m} h_j g_1^j\right) M^1 + \ldots + \left(\textstyle\sum_{j=1}^{m} h_j g_n^j\right) M^n \\
&= \textstyle\sum_{i=1}^{n} \left(\sum_{j=1}^{m} h_j g_i^j\right) M^i
\end{aligned}$$

And so, we have

$$Y = \sum_{i=1}^{n} g_i' M^i,$$

where

$$g_i' = \sum_{j=1}^{m} h_j g_i^j.$$

3.4.2 Decoding

Suppose a destination node receives a set of encoded packets, $X^1, \ldots, X^m$, along with their encoding vectors $g^1, \ldots, g^m$, and wishes to retrieve the original message, that is the original n packets, $M^1, \ldots, M^n$. The destination node can construct the following system of linear equations:

$$\begin{aligned} g_1^1 M^1 + g_2^1 M^2 + \ldots g_n^1 M^n &= X^1 \\ g_1^2 M^1 + g_2^2 M^2 + \ldots g_n^2 M^n &= X^2 \\ \vdots \qquad\qquad\qquad\qquad\quad &\quad \vdots \\ g_1^m M^1 + g_2^m M^2 + \ldots g_n^m M^n &= X^m \end{aligned}$$

or,

$$\begin{pmatrix} g_1^1 & \cdots & g_j^1 & \cdots & g_n^1 \\ \vdots & \ddots & & & \\ g_1^i & & g_j^i & & \\ \vdots & & & \ddots & \\ g_1^m & & & & g_n^m \end{pmatrix} \begin{pmatrix} M^1 \\ \vdots \\ M^j \\ \vdots \\ M^n \end{pmatrix} = \begin{pmatrix} X^1 \\ \vdots \\ X^i \\ \vdots \\ X^m \end{pmatrix}$$

In order to retrieve the original message, we clearly need that $m \geqslant n$. Even in this case, it will only be possible for the system of equations to be solved if at least n of the encoding vectors received at the destination are linearly independent.

There are two ways to try and ensure that this condition holds:

Deterministic Linear Coding

The most obvious approach is to tell every node exactly what linear combinations of incoming packets to make. The advantage is that this way it should be easy to guarantee that each destination has enough linearly independent encoding vectors, and encoded

packets to recover the original message. Also, it would not be necessary to attach the encoding vectors with the encoded packets, as the destination nodes would already know how the encoded packets were constructed.

The main drawback to a deterministic approach is the obvious problem of deciding how to construct a suitable network code. For arbitrary networks, there exist polynomial time coding design algorithms, currently requiring a centralized strategy, [41] however for certain special networks, decentralized algorithms exist, [19].

Random Linear Coding

Another approach, first investigated by Ho, Medard et al in [24] and [25] is to allow each node to uniformly, at random, select the coefficients for all linear combinations of incoming packets to form outgoing packets. The main advantage of this approach is that it can be performed in a completely distributed manner, each node needs no knowledge of an over-arcing coding strategy.

The first possible drawback to Random Linear Network Coding is that, over a finite field, randomly selecting elements of a matrix will result in a singular matrix, with a probability that decreases as the size of the field increases, but increases as the number of nodes increases. It has been shown in [24] that the probability of a successful decoding is at least $(1 - d/q)^{\nu}$, where d is the number of destinations, q the size of the finite field, an ν the number of edges in the network. In [49], simulation shows that with fields as small as $\mathbb{F}_{2^8}$, the actual probability of decoding failure becomes negligible.

The second possible drawback is that the information of how the encoding has been performed needs to be carried along with the encoded packet, that is to say the encoding vector needs to be carried with each packet. As was shown in Section 3.4.1, the operation of updating the encoding vectors is a simple one. This leaves the problem of carrying the extra information through the network. Suppose we have 50 packets, each consisting of 1000 symbols. This is reasonable if we are working with packets of size $\sim$ 1 kilobyte

over the field $\mathbb{F}_{2^5}$. In this case, each encoding vector will consist of 50 symbols, and each packet of 1000 symbols, and so the extra capacity required is $50/1000 = 5\%$. Working with larger packets, or a smaller finite field would decrease this overhead, but even at 5% it would seem to be a reasonable cost to pay compared with the price of gathering the network structure, determining a coding strategy, and then communicating the strategy to every node in the network. We therefore see random linear network coding as the more promising approach. Since network coding involves adding and multiplying elements of a finite field, it will be useful to have an understanding of these mechanisms.

3.4.3 Arithmetic Over Finite Fields

When describing network coding we talk about representing a packet of data as a string of symbols from a finite field, and then forming linear combinations of these symbols. In this section we describe how the required operations of multiplication and addition are performed over a finite field. The number of elements in a finite field is always equal to p^n, for some prime p and positive integer n. Any two finite fields with the same number of elements are isomorphic, that is to say, by renaming the elements of one, the addition and multiplication tables for both are identical. Providing the number of elements of a finite field is therefore enough to describe it completely, and thus the finite fields can be named $\mathbb{F}_{p^n}$, and so $\mathbb{F}_{2^s}$ is the unique finite field (up to isomorphism) with 2^s elements.

Finite Fields of Prime Size

The finite field with p elements can be represented as the ring $\mathbb{Z}/p\mathbb{Z}$. Here, arithmetic is performed modulo p, so for example if $p = 3$, $\mathbb{F}_3$ contains the elements $\{0, 1, 2\}$, with addition and multiplication as shown in Table 3.1.

Finite Fields of Non-Prime Size

We will now describe how to construct a field of size p^n for $n > 2$. First, one needs to consider $\mathbb{F}_p[T]$, that is the ring of all polynomials in T with co-efficients in $\mathbb{F}_p$, and find a

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Table 3.1: *Addition and multiplication tables for the finite field* $\mathbb{F}_3$.

$+$	0	1	T	$1+T$
0	0	1	T	$1+T$
1	1	0	$1+T$	T
T	T	$1+T$	0	1
$1+T$	$1+T$	T	1	0

$\times$	0	1	T	$1+T$
0	0	0	0	0
1	0	1	T	$1+T$
T	0	T	$1+T$	1
$1+T$	0	$1+T$	1	T

Table 3.2: *Addition and multiplication tables for the finite field* $\mathbb{F}_4$.

monic, irreducible polynomial of degree n. For example, a monic, irreducible polynomial of degree 3 in $\mathbb{F}_2[T]$ is $T^3 + T + 1$, since it has no factors with coefficients in $\{0, 1\}$. $T^3 + T + 1$ is not however irreducible in $\mathbb{F}_3[T]$ since $1^3 + 1 + 1 = 0$ and so $(T - 1)$ is a factor, indeed $T^3 + T + 1 = (T + 2)(T^2 + T + 2)$. It is known that a monic polynomial of degree n always exists in $\mathbb{F}_p[T]$ for all prime p and all positive integer n.

With this monic polynomial, say $f(T)$ found, we construct a finite field $\mathbb{F}_p[T]/(f(T))$. The elements of the field will be of the form $a_0 + a_1T + \ldots a_{n-1}T^{n-1}$, where $a_i \in \mathbb{F}_p$. Clearly there are p possible values for each co-efficient, and n co-efficients and so we have p^n field elements. Addition is performed modulo p and multiplication is performed by using the fact that $f(T) = 0$.

Example 3.6 *We now explicitly construct the finite field of size* $4 = 2^2$*. We have that* T^2+T+1 *is a monic irreducible quadratic in* $\mathbb{F}_2[T]$*, and so* $\mathbb{F}_2[T]/(T^2+T+1)$ *is a finite field of size 4, with elements* $\{0, 1, T, T + 1\}$*. Using the fact that* $T^2 + T + 1 = 0$*, or* $T^2 = T + 1$*, we are able to perform addition and multiplication as displayed in Table 3.2 and see that all the required properties of a field are satisfied.*

Converting a String of Bits into a String of Symbols

Having seen how we can represent the elements of a finite field of size 2^s as polynomials of degree $s-1$ with binary coefficients, it is a simple task to convert a string of L bits into L/s elements of $\mathbb{F}_{2^s}$: We break the string into L/s substrings of length s, and then these s binary digits can simply be considered as the s coefficients of a polynomial. We consider it informative to present a simple example of the encoding and decoding operations at a binary level.

Example 3.7 *Suppose at a node we have three incoming packets,* $P^1 = 1100$*,* $P^2 = 0111$ *and* $P^3 = 1011$*, and we wish to generate three outgoing packets encoded using random linear network coding over the field* $\mathbb{F}_{2^2}$*. We break each packet down into two strings of length two, and then interpret each string of length two as an element of* $\mathbb{F}_{2^2}$*. Then* $P^1 = [P_1^1 P_2^1] = [1+T, 0], P^2 = [T, 1+T], P^3 = [1, 1+T]$*. Suppose our encoding vectors are chosen as* $g^1 = [g_1^1, g_2^1, g_3^1] = [T, 0, 1+T], g^2 = [T, 1+T, 1+T], g^3 = [1, 1, T]$*, then we generate our encoded packets one symbol at a time:*

$$Q_1^1 = \sum_{i=1}^{3} g_i^1 P_1^i = T.(1+T) + 0.T + (1+T).1 = T^2 + 1 = T$$

and

$$Q_2^1 = \sum_{i=1}^{3} g_i^1 P_2^i = T.0 + 0.(1+T) + (1+T).(1+T) = 1 + T^2 = T,$$

so $Q^1 = [T, T]$*. Similarly,* $Q^2 = [1+T, 0]$*,* $Q^3 = [0, T]$*. Suppose a destination node receives the three encoded packets and the three encoding vectors. As long as the three encoding vectors are linearly independent, the destination node simply needs to solve a linear system of equations for each symbol. The systems for different symbols have identical left hand sides, and so solving many requires very little more effort than solving one. For the first*

symbol we have:

$$\begin{pmatrix} T & 0 & 1+T \\ T & 1+T & 1+T \\ 1 & 1 & T \end{pmatrix} \begin{pmatrix} P_1^1 \\ P_1^2 \\ P_1^3 \end{pmatrix} = \begin{pmatrix} Q_1^1 \\ Q_1^2 \\ Q_1^3 \end{pmatrix} = \begin{pmatrix} T \\ 1+T \\ 0 \end{pmatrix}$$

Through a series of elementary row operations, we obtain the system:

$$\begin{pmatrix} T & 0 & 1+T \\ 0 & 1+T & 0 \\ 0 & 0 & 1+T \end{pmatrix} \begin{pmatrix} P_1^1 \\ P_1^2 \\ P_1^3 \end{pmatrix} = \begin{pmatrix} T \\ 1 \\ 1+T \end{pmatrix},$$

from which we see that $(1+T)P_1^3 = (1+T)$*, and so* $P_1^3 = 1$*. Similarly,* $(1+T)P_1^2 = 1$*, and so* $P_1^2 = T$*. Finally,* $TP_1^1 + (1+T)P_1^3 = TP_1^1 + 1 + T = T$*, and so* $P_1^1 = (1+T)$*. For the second symbol, we perform the same row operations, and thus attain the same left hand side with a new right hand side. Doing this we find* $P_2^1 = 0, P_2^2 = (1+T), P_2^3 = (1+T)$*, and finally by converting our symbols in* $\mathbb{F}_2^2$ *back to binary digits, we find that we have recovered the original message.*

3.4.4 Important Results

Having demonstrated the principle of network coding and given examples of how it can be used in both the case of multiple unicasts and multicasts to improve network utility, we now wish to apply network coding within our existing optimization framework. In order to do so, we need to make use of some theorems concerning the use of network coding in multicasting.

Theorem 3.8 *For a fixed capacity network using linear network coding, a multicast flow can be found which satisfies the min-min-cut bound. That is to say* $R(s, D) = \min_{d \in D} \text{mincut}(s, d)$.

Proof. See [31]. □

Theorem 3.9 *Consider a multicast from a source node s to a destination set $d_1, \ldots, d_k$ in a fixed capacity network. If the network is able to support a **unicast** at rate α from s to d_i $\forall$ $i = 1, \ldots k$, using traditional routing with no other traffic in the network, the network is able to support a **multicast** from s to $d_1, \ldots, d_k$ at rate α using linear network coding.*

Proof. See [29]. □

This theorem is clearly important to us because it means we can use our existing model to calculate independent unicasts, and then use the Network Coding principle to combine them into a multicast, an idea suggested in [53]. The proof of this theorem works on the basis that, unlike in routing, where each destination node requires all of the original packets sent by the source node, in network coding each destination node simply requires a given number of encoded packets, and so all encoded packets become equally useful for all destinations.

If, of course, it was difficult to calculate the specific Network Code needed to achieve this multicast rate, the computational effort and time required to find the solution might render the whole method counter productive for our purposes, but we have the following theorem about random network codes to greatly improve ease of implementation.

Theorem 3.10 *Over a large enough finite field, random linear network coding achieves the required multicast flow, with probability of a successful code transmission converging to one as the field size increases.*

Proof. See [24] □

With this theorem, intermediate nodes no longer need to be aware of an over-arcing network coding strategy and can simply form random linear combinations of any incoming

packets. This reduces communication overheads as well as cancelling the need for a centralized calculation of the network coding strategy.

The key thing to notice from these theorems is that flows for different destinations do not compete for communication variables when sharing an edge. This means that within our model, we can treat the multicast flow as a set of unicast flows, all of which satisfy the constraints of our existing model, but where separate unicast flows along the same edge do not compete for capacity. The actual number of coded packets that need to be transmitted down a given edge during a given time slot is simply the maximum of the number of packets that the separate unicast flows require. This is best explained through an example.

3.5 A Worked Example

Suppose we wish to multicast a message M through our network from source s to destinations d_1 and d_2 in Figure 3.8. We suppose for illustrative purposes that the message is split into 100 packets of equal size, $P^1, \ldots, P^{100}$. Figure 3.11 shows a unicast flow to each of the two destinations which both satisfy simple flow constraints. The coloured numbers represent the number of packets to be transmitted down each edge. We now look at three

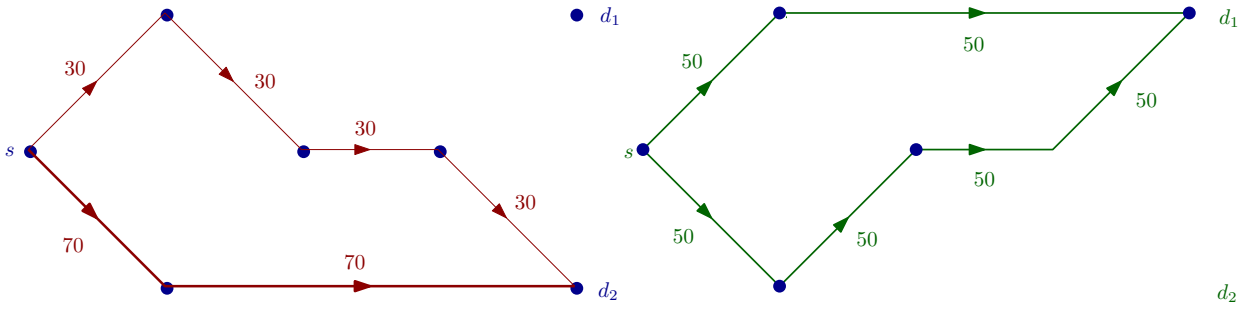

Figure 3.11: *Feasible unicast flows of 100 packets from s to d_1 and d_2.*

possible ways in which we can combine these two unicast flows into a multicast flow.

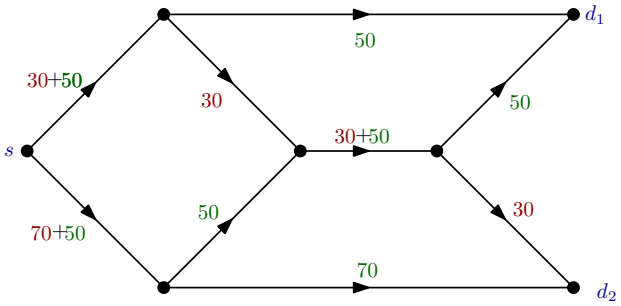

Figure 3.12: *A naive multicast solution, treating the two unicasts in figure3.11 as independent.*

3.5.1 Solving using the naive approach

The simplest way is to treat the two flows as completely independent, see Figure 3.12. Again, in the figure the coloured numbers represent the number of packets being sent along each edge. In this way, all edges which share the two unicasts will have to share their communications resources. This is clearly a suboptimal approach but at least it is very easy to implement as we simply use our existing model, and have two messages, with the same source but different destinations.

3.5.2 A More Intelligent Routing Solution

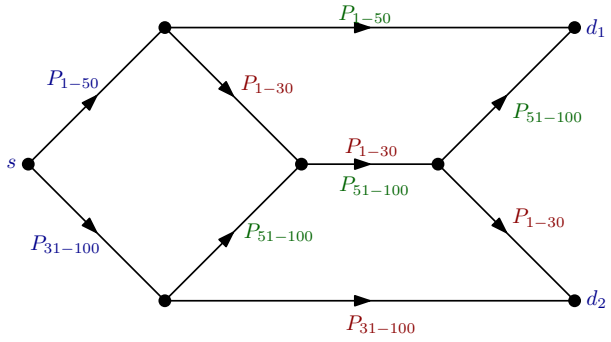

Figure 3.13: *A more intelligent routing strategy, using the fact that the same 100 packets are required at both destinations.*

Definition 3.11 *In figures (3.13 and 3.14) we make use of notation P_{i-j} to mean packets $\{P^i, \ldots, P^j\}$.*

A second approach is to notice that the same 100 packets are needed at both destinations, and thus in the naive approach above, the same packets are most likely being sent down the same edge twice. A more intelligent routing strategy is demonstrated in Figure 3.13. Here the blue numbers indicate packets sent along an edge which will be sent to both of the destinations. With this strategy, the only edge carrying different packets for the two destinations is the bottle neck edge. Two edges carry less packets than in the naive solution, and no edge carries more packets than in the naive solution, and therefore this is an improved solution. The problem with this approach is that an over-arcing routing strategy would need to be known by each intermediate node, telling them which individual packets need to be sent along which edges. Also, in general, calculating the set of edge disjoint multicast trees which achieve the optimal multicast rate for given capacity constraints is NP-complete, so incorporating this into our model where we have continuously variable capacities would seem unviable.

3.5.3 A Network Coding Solution

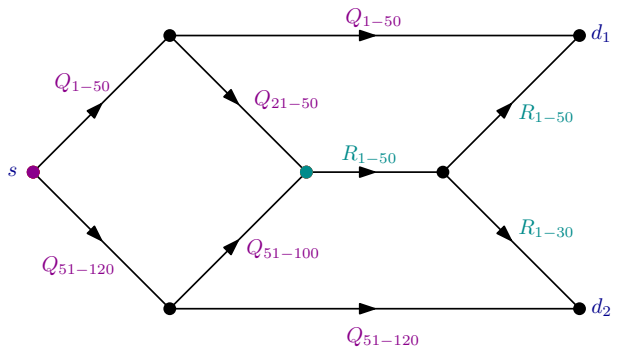

Figure 3.14: *A Network Coding Solution, random linear encoding is performed at the source and the central node.*

The third approach is to use random linear network coding. Here, instead of the separate packets P^1 to P^{100} leaving the source node, each packet that leaves is a random linear combination of the packets $P^1, \ldots P^{100}$, where $Q^j = \sum_{i=1}^{100} g_j^i P^i \;\; j = 1, \ldots, 120$. At each intermediate node, if there is more than one incoming edge, incoming packets are

once again formed into random linear combinations to be transmitted, as in Figure 3.5.3. We see that in this example, the only other node where further encoding is required is the central node with two incoming edges. Here, we perform the encoding operation $R^k = \sum_{j=21}^{100} h_k^j Q^j \;\; k = 1, \ldots, 50$. This way, every packet contains data useful to every destination, and communication resources no longer have to be shared. In the example in the diagram, both d_1 and d_2 are supplied with 100 encoded packets, as well as the encoding coefficients required to decode them. Each destination is able to decode the original message as long as the 100×100 matrix of coding coefficients is non-singular. The probability of these matrices being non-singular increases as we increase the size of the field over which we choose the coefficients, and obviously decreases as we increase the number of destinations and hence matrices required to be non-singular.

3.6 Using Network Coding to Extend our Model to Multicast

We see from Theorem 3.9 and Section 3.5.3 that we can treat a multicast as a set of unicasts which do not compete for channel capacity, this is because the encoded packets being transmitted along a given edge in a given time slot are potentially useful for any destination node.

We thus proceed to adapt the existing model of chapter two for multiple unicasts to a model for a single multicast as follows:

3.6.1 Variables

We still require power variables, $p_{e,t}$ and physical flow and buffer variables, $f_{e,t}$, $b_{v,t}$, representing the amount of encoded data transmitted along a given edge in a given time slot, and the amount of encoded data stored at a given buffer in a given time slot. Since we are considering only one multicast message, we no longer need a message index on

these variables.

We do, however, need new variables to model the non-competing unicast flows we use to construct the multicast. Since these do not represent the transfer or storage of physical data, we will adopt the convention of [53], and refer to them as **conceptual**. We therefore introduce **conceptual flow variables**:

$c^d_{e,t}$: The amount of data transmitted along edge e in timeslot t forming part of the conceptual flow from the multicast source to destination d.

and **conceptual buffer variables**:

$d^d_{n,t}$: The amount of data stored at node n in timeslot t forming part of the conceptual flow from the multicast source to destination d.

3.6.2 Objective Function

Since an objective function involving variables which do not relate to the physical transfer of data does not make physical sense, we restrict ourselves to objective functions of the form $\Phi(\mathbf{b}, \mathbf{f}, \mathbf{p})$. We still assume the objective is convex and monotone increasing in $\mathbf{p}$.

3.6.3 Constraints

Since we are still considering the same physical network, we retain the same constraints on the power variables as in the original model.

As we are interested in combining unicast flows to create our multicast flow, we require that the conceptual flow and buffer variables for each destination satisfy the flow constraints, that is the modified Kirchoff constraints and the initialisation constraints.

The physical flow variables need to satisfy the scheduling constraints, and are coupled to the conceptual flow variables through the requirement that each actual flow along a given edge in a given time slot must carry enough data to support each of the conceptual flows, i.e.

$$f_{e,t} = \max_{d \in D} c^d_{e,t} \; e \in E, t \in T \tag{3.1}$$

or equivalently,

$$f_{e,t} \geqslant c^d_{e,t} \; d \in D, e \in E, t \in T.$$

Similarly, at each intermediate node, one needs to assure that there are enough physical data bits stored for each conceptual flow to use, and so

$$b_{n,t} = \max_{d \in D} d^d_{n,t} \; n \in N, t \in T \tag{3.2}$$

or equivalently,

$$b_{n,t} \geqslant d^d_{n,t} \; n \in N, t \in T, d \in D.$$

These max operations stem from the fact that any packet of encoded data can be used by any multicast destination, and so we need that there are enough physical bits to satisfy each conceptual unicast flow along each edge and each conceptual buffer at each node.

The main difference is that the actual amount of data transmitted down any edge in a given time slot, need only be the **maximum** of any conceptual flows along that edge in that time slot. And the actual amount of data stored in a buffer at the start of a given time slot need only be the **maximum** of any conceptual buffers stored there at the beginning of the time slot. This is because encoded packets travelling along any edge are useful to **all** destinations, and similarly encoded packets stored in a buffer can be useful to **any** destination.

Finally, we need to consider the coupling constraints between power variables and physical flow variables. Since, with each packet of data transmitted we also need to transmit its encoding vector, we need to build in the overhead cost by ensuring that the total size of the transmitted data, plus encoding vector is no greater than the physical channel allows, that is to say if κ :=(size of encoding vector)/(size of packet), is the

encoding information overhead then

$$(1+\kappa)f_{e,t} \leqslant B\tau \log_2\left(1+\frac{a_{e,e}p_e}{\sum_{l\neq e} a_{l,e}p_l+\sigma^2}\right) \tag{3.3}$$

3.6.4 The Complete Multicast Program

Combining our constraints, and objectives, we construct our nonlinear programming problem for multicasting in networks with multiple access interference (eg. CDMA) using network coding:

$$\begin{array}{rcll}
 & \text{minimize} & \Phi(\mathbf{p},\mathbf{f}) & \\
 & \text{s.t.} & & \\
\mathbf{p},\mathbf{f},\mathbf{c},\mathbf{b},\mathbf{d} & \geqslant & 0. & \\
p_{e,t} & \leqslant & P_e^{\max} & e\in E, t\in T \\
\sum_{e\in E^+(v)} p_{e,t} & \leqslant & P_v^{\max} & v\in V \forall t\in T \\
d^d_{s,1} & = & S & d\in D \\
d^d_{d,t_{\max}+1} & = & S & d\in D \\
b_{v,1} & = & 0 & v\in V\backslash s \\
f_{e,t} & = & 0 & \text{whenever } \mathrm{col}_E(e)\neq \mathrm{col}_T(t) \\
 & & & d\in D, e\in E, t\in T \\
d^d_{v,t+1}-b^d_{v,t} & = & \sum_{e\in E^-(v)} c^d_{e,t}-\sum_{e\in E^+(v)} c^d_{e,t} & v\in V, d\in D, t\in T \\
b_{v,t} & \leqslant & B_v^{\max} & v\in V, t\in T \\
\max_{d\in D} c^d_{e,t} & \leqslant & f_{e,t} & e\in E, t\in T \\
\max_{d\in D} d^d_{v,t} & \leqslant & b_{v,t} & v\in V, t\in T \\
(1+\kappa)f_{e,t} & \leqslant & B\tau\log_2(1+\mathrm{SINR}_{e,t}) & e\in E, t\in T
\end{array} \tag{3.4}$$

Observation 3.5 *If we replace the maximum operation with a sum operation in constraints (3.1) and (3.2) we simply have a reformulation of our original unicast problem, but with all messages originating from the same source node. This is equivalent to the naive routing detailed in Section 3.1.*

This maximum operation rather than a sum operation in constraints (3.1) and (3.2) clearly opens up a larger feasible region. What is unknown without experimentation is to what extent this increased feasible region will lead to an improved optimal objective function. In the final section of this chapter, we experiment with various multicasts in various network environments to see what performance gains can be gained in practice when using network coding over naive routing.

3.7 A Model for Multiple Unicasts, Multicasts, and Broadcasts

Having developed a framework for multiple unicasts, and one for a single multicast, it is now relatively straightforward, with the aid of multiple indices, to consider the general problem of wishing to send multiple unicasts, multicasts and broadcasts through a network.

Consider our usual network, consisting of a node set N, and an edge set E.

Suppose, as in the first chapter, we have a message set $M = \{1, \ldots, k\}$ to be transmitted through our network, and with each message, $m \in M$ we associate a source node $s^m \in V$, and a destination set $D^m \subset V$, as well as a message size S^m. We will continue with the convention of listing edge, or node, and timeslot as subscripts to the right, and placing the message and destination indices as superscripts.

3.7.1 Variables

We still need one power variable for each edge and each time slot, $p_{e,t}$, but now we need a different physical flow variable, $f_{e,t}^m$ and physical buffer variable, $b_{n,t}^m$ for each message, and each edge or node and each time slot. Finally, we need conceptual flow and conceptual buffer variables for each unicast flow within each multicast flow, $c_{e,t}^{m,d}$, and $d_{e,t}^{m,d}$.

3.7.2 New Parameters

Since each multicast message could potentially be encoding over different finite fields, and using different size packets, and the unicast messages require no encoding, with each message we will associate a different encoding information overhead, κ^m. For unicast messages this will be zero, and for multicast messages it would typically be around 0.05. Since data from different messages competes for bandwidth, we have that

$$\sum_{m \in M} (1 + \kappa^m) f_{e,t}^m \leqslant B\tau \log_2 (1 + \mathrm{SINR}_{e,t}) \, \forall e \in E, t \in T, m \in M. \tag{3.6}$$

Wit these alterations, we are now able to present the complete nonlinear optimization problem for transmitting multiple unicast and multicast messages through a multihop wireless network.

3.7.3 The General Model

$$
\begin{array}{rcll}
 & \text{minimize} & & \Phi(\mathbf{p},\mathbf{f},\mathbf{b}) \\
 & \text{s.t.} & & \\
\mathbf{p},\mathbf{f},\mathbf{c},\mathbf{b},\mathbf{d} & \geqslant & 0 & \\
p_{e,t} & \leqslant & P_e^{\max} & e\in E, t\in T \\
\sum_{e\in E^+(v)} p_{e,t} & \leqslant & P_v^{\max} & v\in V \forall t\in T \\
d_{s^m,1}^{m,d} & = & S^m & m\in M, d\in D^m \\
d_{d,t_{\max}+1}^{m,d} & = & S^m & m\in M, d\in D^m \\
b_{v,1}^m & = & 0 & m\in M, v\in V\backslash\{s^m\} \\
f_{e,t}^m & = & 0 & \text{whenever } \mathrm{col}_E(e)\neq \mathrm{col}_T(t) \\
 & & & m\in M, e\in E, t\in T \\
d_{v,t+1}^{m,d}-d_{v,t}^{m,d} & = & \sum_{e\in E^-(v)} c_{e,t}^{m,d}-\sum_{e\in E^+(v)} c_{e,t}^{m,d} & \forall v\in V, d\in D, t\in T, m\in M \\
\sum_{m\in M} b_{v,d,t} & \leqslant & B_v^{\max} & v\in V, t\in T, m\in M \\
c_{e,t}^{m,d} & \leqslant & f_{e,t}^m & e\in E, t\in T, m\in M, d\in D \\
d_{v,t}^{m,d} & \leqslant & b_{e,t}^m & v\in V, t\in T, m\in M, d\in D \\
\sum_{m\in M}(1+\kappa^m) f_{e,t}^m & \leqslant & B\tau\log_2(1+\mathrm{SINR}_{e,t}) & e\in E, t\in T, m\in M.
\end{array}
\tag{3.7}
$$

In Theorem 2.5 we proved that at a locally optimal solution of the unicast problem, all coupling constraints are active. This will later prove to be a very useful result, and so we state it again here for the general problem incorporating unicasts and multicasts as detailed above in (3.7).

Theorem 3.12 *Suppose that the objective function $\Phi : (\mathbf{p},\mathbf{b},\mathbf{f}) \mapsto \Phi(\mathbf{p},\mathbf{b},\mathbf{f})$ is strictly monotone in $\mathbf{p}$, and that we want to solve the optimization problem detailed in equation (3.7). Then all constraints (3.6) are active at each locally optimal solution of this problem.*

Proof. Although we have more variables than in the unicast case, the proof is identical

to that of Theorem 2.5. In identical fashion it can be shown that the feasible descent direction of that proof is again a feasible descent direction for this problem, and therefore any point for which not all coupling constraints are active cannot be a local optimal. □

Having developed the optimization problem for transmitting multicast messages using network coding and observed the similarity to the optimization problem for transmitting multicast messages using naive routing, we shall compare the solutions to these problems numerically in the final section of this chapter.

3.8 Computational Results Comparing Network Coding to Naive Routing

Before going on in the second half of this thesis to investigate solution methods tailored towards the specific problems discussed and modelled up to now, it is useful to use a standard black box solver to give us a quantitative idea of the sort of performance gains we can expect through using the network coding approach when compared to naive routing. Related work has been done by [40], in which joint network coding and scheduling is considered in wireless networks and compared to joint routing and scheduling. They find encouraging results finding network coding to provide much more energy efficient multicasts than traditional routing. In [49], Wu and Chou compare the multicast throughputs of network coding and Steiner tree multicasting in fixed capacity networks and find very little difference, but highlight the fact that computationally, network coding is much cheaper and can be achieved in a distributed manner. In [53], the authors introduce "conceptual flows" which satisfy flow constraint laws but which do not compete for channel capacity. This is included in a joint power control and routing optimization problem on a non-timeslotted network by setting the actual flow along an edge as the maximum of all conceptual flows along that edge. The authors then go on to solve this problem using

a dual decomposition approach similar to [27].

For ease of implementation, and due to the combinatorial difficulty of Steiner tree based solutions, we shall compare network coding to naive routing only, and not seek to find the optimal solutions a Steiner tree based approach could give us at this time.

Our investigations focus on two network structures, the butterfly network as displayed in Figure 3.8, and the newly introduced multihop backhaul network, see Figure 3.15. The backhaul network is a commonly used wireless network structure in situations where installing a wireline network would prove too costly or impractical, and allows the linking of a grid of wireless transmitters back to the wireline network. For these networks we seek the minimal total energy required to transmit a single multicast from the source to the specified destinations under various interference conditions and for various parameter choices, and compare the results from naive routing and using network coding. In each case, we feed the problem in its entirety into the NAG sparse non-linear programming solver, [35], and in the non-convex problem instances we run each problem twenty times from random starting points and take the minimum value.

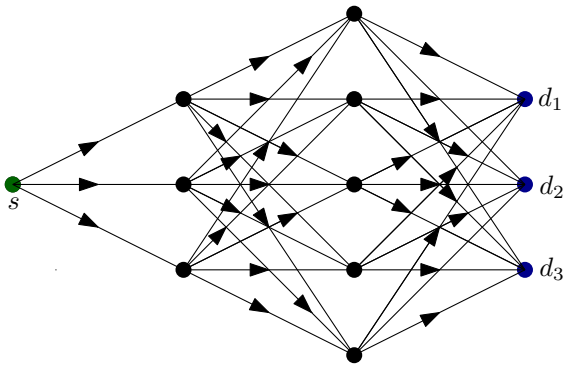

Figure 3.15: *A 1-3-5-3 Backhaul network.*

3.8.1 Parameter Values

Given the directed graph of the network, and the source and destination nodes, nearly all aspects of the optimization problem are fixed. The only choices we need to make are the values to assign to the parameters $B, S, \kappa, \tau, \sigma^2$, and the gain matrices A. For ease of implementation in the butterfly case, all parameters were given simple values, and the gain matrix was calculated such that $a_{i,j}$ was inversely proportional to the distance squared between the source node of edge i and the destination node of edge j.

For the backhaul network, all parameter values, as well as the gain matrix are given realistic physical values. See appendix D for the actual values used.

3.8.2 Results

In all the experiments undertaken, Network Coding significantly outperforms naive routing as expected. We first compare the total transmit energy required to transmit messages of varying size over fixed channel conditions, with and without interference, assuming the encoding vector overhead to be negligible. As can be seen in Figure 3.16, in the butterfly network with or without cross channel interference almost twice as much energy is required for the multicast transmission when using naive routing when compared to network coding. This difference remains fairly consistent as the message size is varied.

In 3.17 we compare the minimum energy consumption required for a multicast transmission in the 1-3-5-3 backhaul network using network coding or naive routing. Again, the reduction in energy consumption is significant and consistent over a wide range of message sizes and various levels of interference, that is zero interference, full interference, or interference between different rings only.

It is also worth nothing that the energy reduction through the use of network coding is greater than the energy reduction through eliminating interference. It is important to stress that although these results look impressive, it is to be expected that naive routing

performs badly as was shown in the examples in Section 3.1. Although we have not directly compared network coding to Steiner tree packing, we know that Steiner tree packing cannot provide better solutions than network coding, in some situations will return worse solutions, and is computationally expensive to implement.

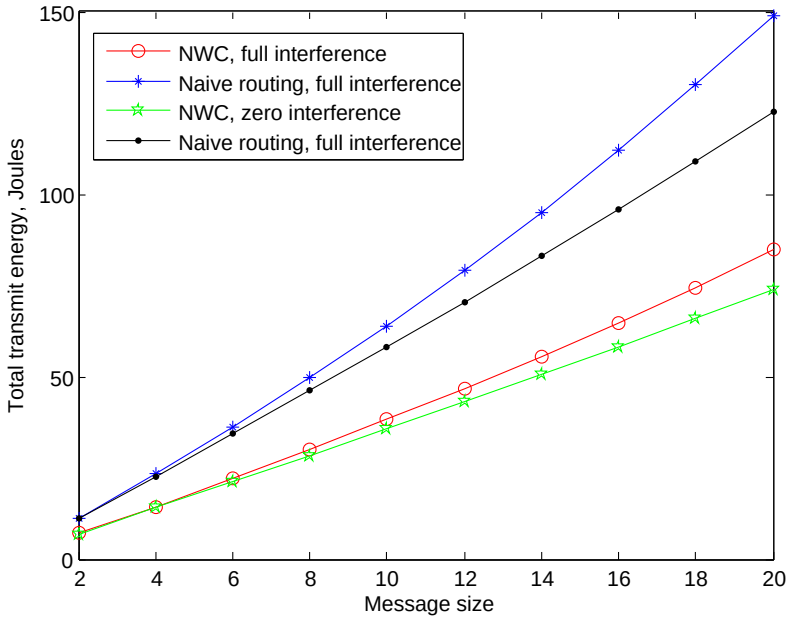

Figure 3.16: *Total multicast transmit energy for varying message sizes in Megabits, with or without interference in the butterfly network over 20 timeslots.*

Having shown that, with encoding vector overhead assumed negligible, network coding greatly outperforms naive routing, another important question is the impact that these overheads have on the solution to the optimization problem. To this end, we ran the optimization problem for fixed message size and channel conditions, whilst varying κ, the encoding co-efficient overhead, in constraint (3.3) between 0 and 0.125. In both networks, network coding still outperforms naive routing, even with encoding vectors 1/8 the size of the packets.

In Figure 3.18 we see that with full or zero interference in the butterfly network, when transmitting a message of size 10 Megabits, the carrying of the encoding coefficients does

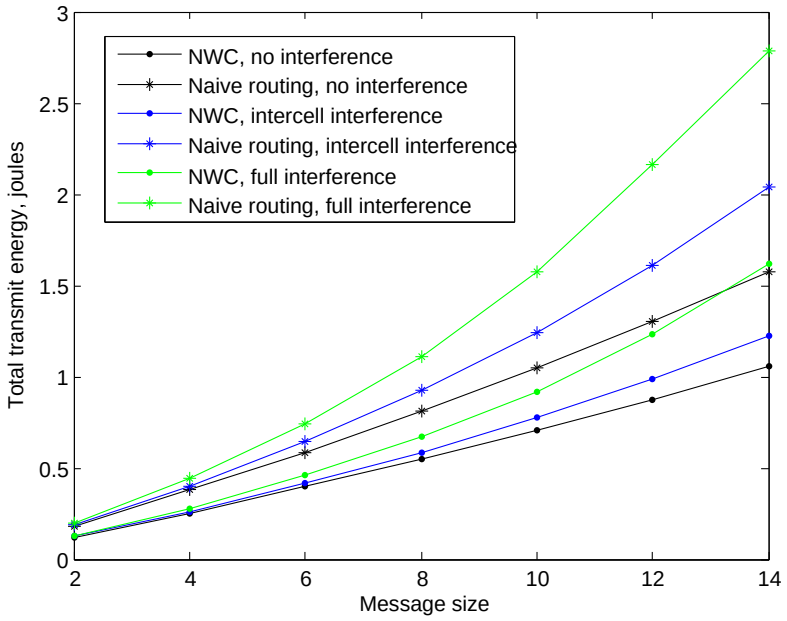

Figure 3.17: *Total multicast transmit energy for varying message sizes in Megabits, with various interference levels in the backhaul network over 20 timeslots*

not result in significantly increased energy consumption, and in Figure 3.19 we see the same for the 1-3-5-3 backhaul network.

In this chapter we have investigated various methods for incorporating multicast transmissions into our optimization problem and settled on random linear network coding as the best approach due to its ease of implementation and relative simplicity as compared to the Steiner tree packing approach, and the improved solutions to the optimization problem as compared to naive routing.

For the remainder of the thesis we will focus on algorithmic methods for solving the optimization problems discussed thus far.

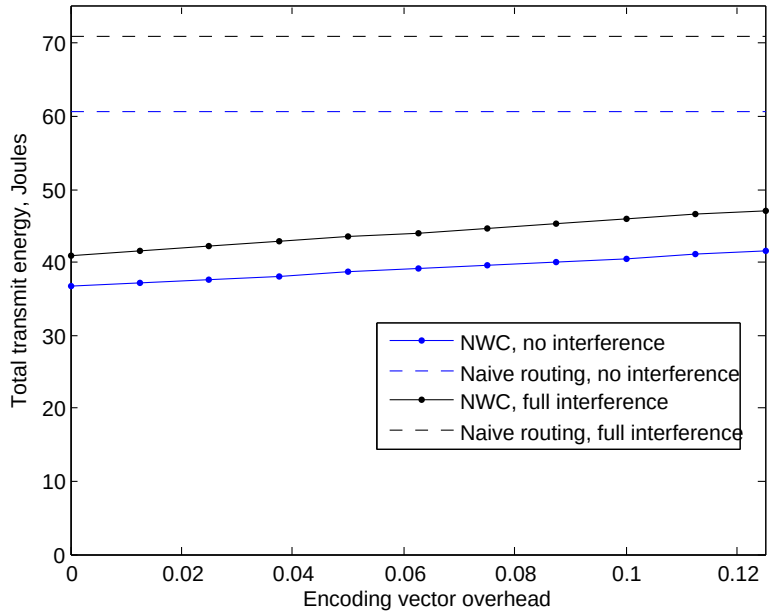

Figure 3.18: *Total multicast transmit energy for varying encoding coefficient overheads, with or without interference in the butterfly network over 20 timeslots, with message size 20.*

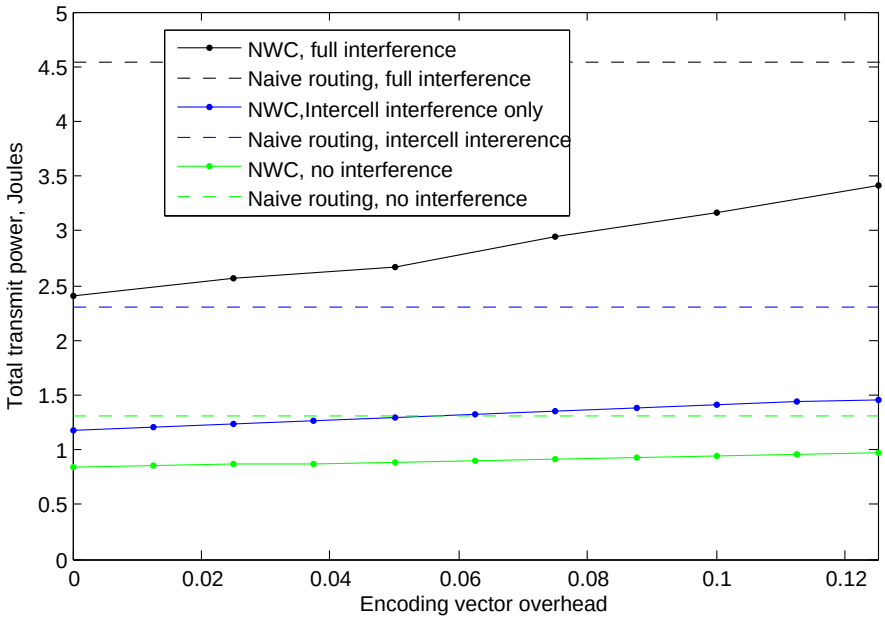

Figure 3.19: *Total multicast transmit energy for varying encoding coefficient overheads, with various interference levels in the backhaul network over 20 timeslots, with message size 20.*

Chapter 4

Solution Methods for the General Problem

The remainder of this thesis is focused on algorithms for solving the optimization problems developed in chapters two and three. We will look at ways in which the non-convexity first discussed in section (2.4.3) can be worked around as well as questioning to what extent it is necessary to worry about the non-convexity. In this chapter we focus on a dual decomposition algorithm which has the benefit of being able to exploit the split level structure of the problem in hand, as well as having the potential to be performed in a distributed way. We will present numerical results of the application of the dual decomposition algorithm, and address its main drawback, its slow convergence rate, through a convergence acceleration technique, [15]. In the next chapter, we will look at a primal co-ordinate descent approach first introduced in [17], exploring its possible benefits as well as short comings and numerically testing it for a range of network environments. Before looking into these algorithmic approaches, it is useful to consider the way the non-convexity of the problem may affect the solutions returned by an algorithm.

4.1 Tackling the Non-Convexity

A convex optimization problem has a unique global solution, or a connected set of global optimal solutions, it has no locally optimal solutions which are not globally optimal. Therefore, if we have an algorithm which guarantees convergence to a local minimizer and we apply it to a convex problem, we are guaranteed to find a global minimizer. If the problem we are dealing with is not convex, i.e. it has either a nonconvex objective or a nonconvex constraint, then we can not guarantee that the problem has a unique global solution, and it is often impossible to tell whether the solution we have reached is truly the global optimum, or just locally optimal. It is therefore very desirable to have a convex program, or to be able to change a non-convex problem into a convex one. Indeed, Mung Chiang suggests in [9] that there are three broad angles of attack for handling non-convexity:

- Solve the difficult nonconvex problem. This can be done by exploiting some specific structural properties of the problem in hand, telling us that although the problem is non-convex, we can guarantee that the solution we reach is a globally optimal one. This approach requires deep understanding of the constraints and objective of the problem in hand.

- Avoid solving the nonconvex problem. This can sometimes be done by a change of variables, turning a seemingly nonconvex problem into a convex one. Alternatively, we can approximate our nonconvex problem with a convex one, or find conditions under which the problem becomes convex. This is similar to what we did in solving the simplest case in Section 2.5.

- Reformulate the original problem in such a way that the nonconvexity is no longer present. Optimization problems are built up on physical assumptions and design

criteria. To alter the structure in this way does not lie within mathematical optimization but is rather an engineering exercise in this case where we are dealing with complicated relationships between communication resources and data capacities.

It is worth reminding ourselves here that although convexity guarantees a unique global minimum, it is not necessary to be convex in order to have a unique global minimum.

4.1.1 Convex Approximation

In this section we combine two of the three techniques discussed in the previous section to transform our problem into a convex optimization problem. We shall make an approximation and then also make a change of variables to end up with convex constraints. Consider the coupling capacity constraint presented in the second chapter:

$$\sum_{m\in M} f_{e,t}^{m} \leqslant B\tau \log_2\left(1+\frac{a_{e,e}p_{e,t}}{\sum_{l\in E\setminus e} a_{l,e}p_{l,t}+BN_0}\right)\ e\in E, t\in T.$$

It is necessary now to look at the properties of the log function in order to understand how one might approximate our constraints. It is easy to show that

$$\lim_{x\to+\infty}\left(log_2(1+x)-log_2(x)\right)=0,$$

and

$$\lim_{x\to 0}\left(log_2(1+x)-\frac{x}{\ln 2}\right)=0,$$

With these simple results, it would seem that if $\mathrm{SINR_{e,t}} \gg 1$, remembering that

$$\mathrm{SINR_{e,t}} := \frac{a_{e,e}p_{e,t}}{\sum_{l\in E\setminus e} a_{l,e}p_{l,t}+BN_0}$$

we can approximate our coupling constraints with

$$\sum_{m \in M} f_{e,t}^{m} \leqslant B\tau \log_2(SINR_{e,t})$$

This approximation is suggested in [27]. It would only seem reasonable to use this approximation if $\mathrm{SINR_{e,t}} \gg 1 \ \forall e \in E$, which would be the case if $a_{e,e} \gg a_{l,e} \ \forall e \in E, \ l \neq e \in E$, i.e. if the Gain matrix A is strongly diagonally dominated. It is also worth pointing out that this approximation is only valid on a given edge if the power being allocated to that edge is non-zero. This holds for all the edges in the network and suggests that this approximation is only valid if all the communication links in consideration have at least some power allocated to them. Even with this approximation, the coupling constraints are still not convex, since if

$$\widetilde{\Psi}_e(\mathbf{p}) = B\tau \log_2(SINR_e) = B\tau \log_2(\frac{a_{e,e}p_{e,t}}{\sum_{l \in E \setminus e} a_{l,e}p_{l,t} + BN_0})$$

we have that

$$\nabla^2 \widetilde{\Psi}_e(\mathbf{p})_{e,e} = \frac{\ln 2B\tau}{(\sum_{l \neq e} a_{l,e}p_{l,t} + BN_0)^2} A_e A_e^T$$
$$\nabla^2 \widetilde{\Psi}_e(\mathbf{p})_{e} = (0, \ldots, -\frac{\ln 2B\tau}{p_{e,t}}, \ldots, 0)^T$$

Since the negative of this matrix is not positive semi definite, we have that $\widetilde{\Psi}_e$ is not concave, and thus the approximated coupling constraint is not a convex constraint. Having applied the approximation we now need to make use of a change of variables common in geometric programming in order to make our capacity constraints convex.

Change of Variables

Since $log(x) = -log(1/x)$, we can re write our approximated capacity functions $\widetilde{\Psi}_e(\mathbf{p})$ as

$$\begin{aligned}\widetilde{\Psi}_e(\mathbf{p}) &= -B\tau \log_2 \left(\frac{\sum_{l \in E \setminus e} a_{l,e} p_{l,t} + BN_0}{a_{e,e} p_{e,t}} \right) \\ &= -B\tau \log_2 \left(\sum_{l \in E \setminus e} a_{l,e} a_{e,e}^{-1} p_{l,t} p_{e,t}^{-1} + BN_0 a_{e,e}^{-1} p_{e,t}^{-1} \right).\end{aligned}$$

We now make the change of variables $q_{e,t} = \ln(p_{e,t})$, so $p_{e,t} = e^{q_{e,t}}$, and define $\widehat{\Psi}_e(\mathbf{q}) := \widetilde{\Psi}_e(e^{\mathbf{q}})$. Writing this out in full, we have that

$$\widehat{\Psi}_e(\mathbf{q}) = -B\tau \log_2(\sum_{l \in E \setminus e} a_{l,e} a_{e,e}^{-1} e^{(q_{l,t} - q_{e,t})} + BN_0 a_{e,e}^{-1} e^{-q_{e,t}}).$$

We will now show that with this change of variables, our constraints become convex. This change of variables is again suggested in [27] and is a trick often used in Geometric Programming to turn a non-convex function into a convex function, see [5].

Lemma 4.1 *Any function of the form* $\log_2 \left(\sum_{i=1 \ldots N} a_i e^{(x_i - x_{N+1})} + a_{N+1} e^{-x_{N+1}} \right)$ *is convex.*

In order to present the proof, we need to define some notation for various matrices. The proof uses ideas from [6].

Definition 4.2 *Let* $\mathbf{a} = (a_1, a_2, \ldots, a_n)^T \in \mathbb{R}^n$. *Then*

$$\text{diag}(\mathbf{a}) := \mathbf{a}^T I,$$

where I *is the* $n \times n$ *identity matrix. We also define the vector of ones, whose dimension*

is clear by the context it is used in;

$$e := (1, 1, \ldots, 1)^T.$$

Finally, specific to this proof, we define $\mathbf{z} := (a_1 e^{x_1}, a_2 e^{x_2}, \ldots, a_N e^{x_N}, 0)^T$.

Proof (Proof of Lemma 4.1). Let

$$\begin{aligned} f(\mathbf{x}) &= \log_2 \left(\sum_{i=1\ldots N} a_i e^{(x_i - x_{N+1})} + a_{N+1} e^{-x_{N+1}} \right) \\ &= \log_2 \left(e^{-x_{N+1}} \left(\sum_{i=1\ldots N} a_i e^{x_i} + a_{N+1} \right) \right). \end{aligned}$$

Then,

$$\begin{aligned} \frac{\partial f}{\partial x_j} &= \frac{\ln 2 a_j e^{x_j}}{\sum_{i=1}^N a_i e^{x_i} + a_{N+1}} \text{ for } j \in \{1 \ldots N\} \\ \frac{\partial f}{\partial x_{N+1}} &= -\ln 2 \\ \frac{\partial^2 f}{\partial x_j^2} &= \ln 2 \frac{a_j e^{x_j} (\sum_{i=1}^N a_i e^{x_i} + a_{N+1}) - a_j^2 e^{2x_j}}{(\sum_{i=1}^N a_i e^{x_i} + a_{N+1})^2} \text{ for } j \in \{1 \ldots N\} \\ \frac{\partial^2 f}{\partial x_j \partial x_j} &= -\ln 2 \frac{a_j a_k e^{x_j} e^{x_k}}{(\sum_{i=1}^N a_i e^{x_i} + a_{N+1})^2} \text{ for } j \in \{1 \ldots N\}, k \in \{\{1 \ldots N\} \backslash j\}. \\ \frac{\partial^2 f}{\partial x_{N+1} \partial x_j} &= 0 \text{ for } j \in \{1 \ldots N+1\}. \end{aligned}$$

Combining all this, and using the definitions from above, we have that

$$\nabla^2 f(\mathbf{x}) = \frac{\ln 2}{(e^T z + a_{N+1})^2} \left(\left(e^T z + a_{N+1} \right) (diag)(\mathbf{z}) - \mathbf{z}\mathbf{z}^T \right).$$

Finally, we need to show that this Hessian is positive semidefinite. Consider an arbitrary $\mathbf{v} \in \mathbb{R}^{N+1}$. We need to show that $\mathbf{v}^T \nabla^2 f(\mathbf{x}) \mathbf{v} \geqslant 0$. We can ignore the constant $\frac{\ln 2}{(e^T z + a_{N+1})^2}$

since it is always positive.

$$\begin{aligned}\mathbf{v}^T\left((e^T z + a_{N+1})(diag)(\mathbf{z}) - \mathbf{z}\mathbf{z}^T\right)\mathbf{v} &= \left(\sum_{i=1}^{N} z_i + a_{N+1}\right)\left(\sum_{i=1}^{N+1} v_i^2 z_i\right) - \left(\sum_{i=1}^{N+1} v_i z_i\right)^2 \\ &\geqslant \left(\sum_{i=1}^{N+1} z_i\right)\left(\sum_{i=1}^{N+1} v_i^2 z_i\right) - \left(\sum_{i=1}^{N+1} v_i z_i\right)^2.\end{aligned}$$

This inequality holds since $a_{N+1} > 0$ by assumption, and $z_{N+1} = 0$. Since $z_i = a_i e^{x_i}$, and $a_i > 0$, $z_i > 0$. Letting $s_i = v_i z_i^{\frac{1}{2}}$, $t_i = z_i^{\frac{1}{2}}$ and applying the Cauchy-Schwartz inequality to $\mathbf{s}$ and $\mathbf{t}$ we get the desired result. □

With this Lemma proved, it becomes evident that we have transformed our coupling constraints into convex constraints. We now need to check that the convexity of the other constraints and objective is not compromised by the change of variables. The objective function becomes a sum of exponentials, which is evidently convex. The linear constraints become of the form

$$a_1 e^{q_1} + a_2 e^{q_2} + .. + a_k e^{q_k} - K \leqslant 0,$$

which are clearly also convex. The final constraints to be considered involving the power variables are the non-negativity constraints. Luckily these constraints become redundant in the $\mathbf{q}$ variables since requiring that $\mathbf{p} > 0$, means that $\mathbf{q} = \ln(\mathbf{p})$ can take any real value.

We have now shown that we can construct a convex optimization problem, guaranteed to have a connected set of global minimizers. This is good in many respects as it means that if an algorithm we design to solve this problem reaches a minimizer, we know it has found a global minimizer. It also means, as we shall see shortly, that there is no duality gap for this problem. The downside is that we do not know exactly how the approximation has affected the problem and indeed how close the minimizer of the approximated and

convexified problem is to the solution of the original problem. In order to investigate this problem we will go back to the simple problem of Section 2.5 of which we have a good understanding.

4.1.2 Comparison of Approximation with Original Problem for Simplest Case

Looking at the simple problem discussed in section 2.5 and applying the approximation and change of variables discussed previously in this chapter, we are left with the following convex optimization problem:

$$\begin{aligned}
\text{minimize} \quad & e^{q_1} + e^{q_2} \\
\text{such that} \quad f_1 + f_2 \quad & = C, \\
e^{q_1} + e^{q_2} \quad & \leqslant P_{\max} \\
f_1 \quad & \leqslant B\tau \log_2\left(\frac{a_{11}e^{q_1}}{a_{12}e^{q_2} + \sigma^2}\right), \\
f_2 \quad & \leqslant B\tau \log_2\left(\frac{a_{22}e^{q_2}}{a_{21}e^{q_1} + \sigma^2}\right).
\end{aligned}$$

Applying the same principles as applied in the previous chapter and again letting $\upsilon := B\tau$, we reach that:

$$\frac{1}{\sigma^2}(e^{q_1} + e^{q_2})(\lambda) = \frac{a_{22}2^{\frac{\lambda}{\upsilon}} + (a_{12} + a_{21})2^{\frac{C}{\upsilon}} + a_{11}2^{\frac{C-\lambda}{\upsilon}}}{a_{11}a_{22} - a_{12}a_{21}2^{\frac{C}{\upsilon}}}$$

The easiest way to compare the original problem with the approximated problem is to compare the graphs of $(e^{q_1} + e^{q_2})(\lambda)$ and $(p_1 + p_2)(\lambda)$ for the various different cases and see how the locations and function values for the global optima compare. For simplicity, we fixed $\upsilon = 10$, $\sigma^2 = 1$, $C = 50$ and considered the completely symmetrical case, setting $a_{11} = a_{22} = 1$ and varying $a_{12} = a_{21}$. In figures (4.1) we show the graphs of $(p_1 + p_2)(\lambda)$

for the original problem, and $(e^{q_1} + e^{q_2})(\lambda)$ in the convexified version. The first graph shows that in the case where we have a unique global minimizer and a high SINR on both links at the solution due to zero cross channel interference, the convex approximation is reasonably accurate, but slightly overestimates the required powers. The remaining three graphs show that as the interference is increased, the convex approximation becomes less accurate, until finally the minimizing solution to the original problem is to route all the data down one edge at a cost of 31 Watts, whereas the minimizing solution to the convexified problem is to route half of the data down each edge at a total cost of 295 Watts. The conclusion we make is that this approximation approach is only valid in situations where there is very low interference, and in these situations the original problem already has a unique solution and so the change of variables and approximation is unnecessary. In proceeding to solve the problem therefore, we see no point in attempting to remove this non-convexity, and will instead attempt to solve the original problem.

4.2 Numerical Solution techniques

We now focus on computational methods for solving this problem. We have several requirements for an algorithm for solving our optimization problem:

- **Distributivity**

  As we have that each node in the network is capable of computation, it would be sensible to make use of this spread computational power by applying the algorithm in a distributed manner. Since in wireless networks we have limited bandwidth availability it would be preferable for each node to have to communicate with other nodes a minimum amount in order for it to know what action it needs to perform, that is how much power to allocate to each link and how much of each message to send down each link in a given time step. In other words we would like each node to only need locally measurable or locally attainable information to perform the

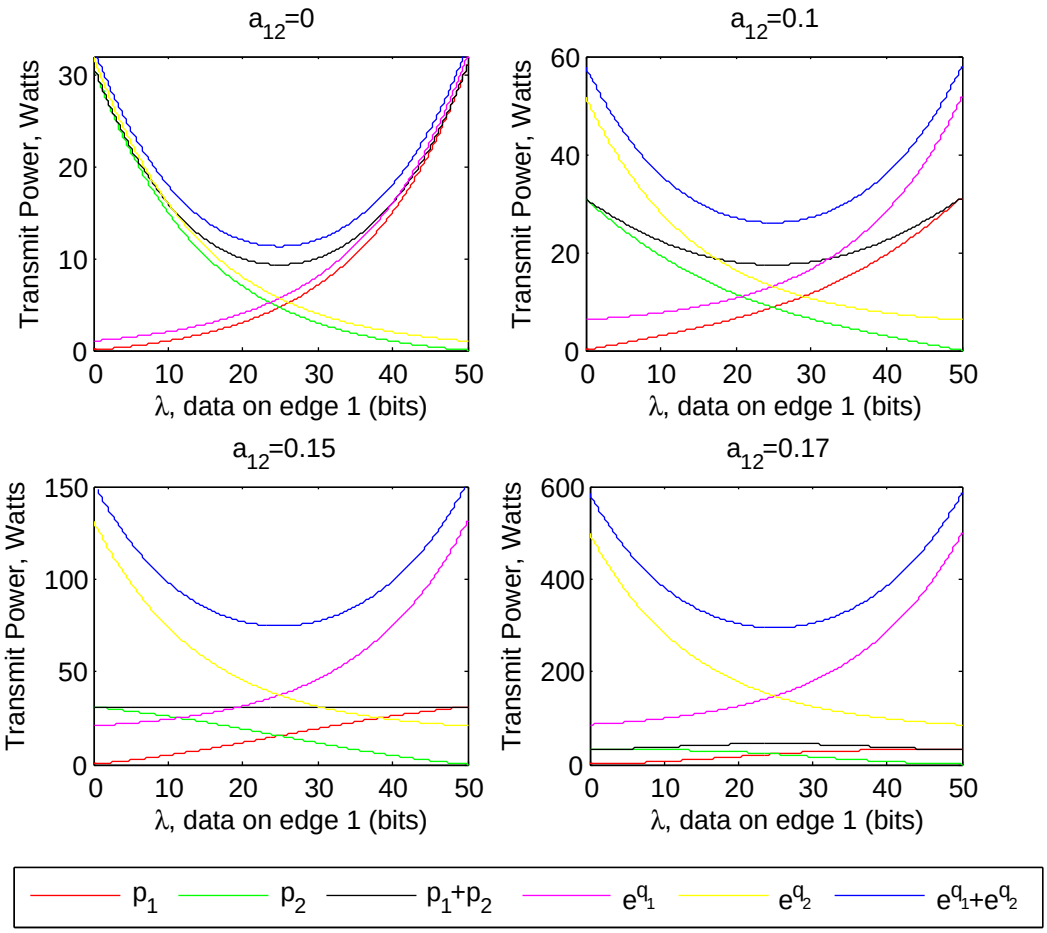

Figure 4.1: *Comparison of Original problem to Convexified Problem for varying interference levels*

algorithm.

- **Simplicity**

  Ideally, the algorithm would make use of existing techniques available for solving network flow problems and power control problems. Since the network variables and power variables are only linked through the link capacity constraints, this points us towards a decomposition approach.

- **Speed**

  It is desirable that the algorithm reaches a minimizer quickly.

- **Optimality**

  Ideally, the algorithm would converge to a global minimizer.

One way to develop an algorithm to fulfil these requirements, and exploit the structure of our problem, is to use the theory of duality.

4.2.1 Duality

We will briefly introduce the concepts of duality as applied to a mathematical program. Consider the problem,

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{such that} \quad & x \in X, \\ & g_j(x) \leqslant 0, \quad j = 1, \ldots, r, \end{aligned} \tag{4.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}, g_j : \mathbb{R}^n \mapsto \mathbb{R}$ are given functions, $X \subseteq \mathbb{R}^n$. We refer to this problem as the **primal problem**. We will denote the minimal function value of this problem by f^*,

and the global minimizer as x^*:

$$f^* := \inf_{x \in X, g_j(x) \leqslant 0, j=1,\ldots,r} f(x) =: f(x^*).$$

For a primal problem we can define the **Lagrangian**. This is an important concept in constrained optimization.

Definition 4.3 *The **Lagrangian function** for an inequality constrained optimization problem, (4.1) is* $L : \mathbb{R}^{n+r} \mapsto \mathbb{R}$,

$$L(x, \mu) := f(x) + \sum_{j=1}^{r} \mu_j g_j(x) = f(x) + \mu^T g(x),$$

where $\mu = (\mu_1, \ldots, \mu_r)^T \in \mathbb{R}^r_+$.

For any primal problem we can define its **dual function**, $q : \mathbb{R}^r \mapsto \mathbb{R}$ by

$$q(\mu) = \inf_{x \in X} L(x, \mu).$$

We can then define the **dual problem** as

$$\begin{aligned} \text{maximize} \quad & q(\mu) \\ \text{subject to} \quad & \mu \geqslant 0, \end{aligned}$$

and the maximal objective function value of this problem by q^*, with corresponding maximizing dual variables μ^*:

$$q^* = \sup_{\mu \geqslant 0} q(\mu) =: q(\mu^*). \tag{4.2}$$

We now present two well known important results concerning duality. For a thorough introduction to the topic see [3] and [33]:

Proposition 4.4 *The dual function q is concave over the domain $D = \{\mu \mid \mu \geqslant 0, q(\mu) > -\infty\}$.*

Proof. See [3]. □

Theorem 4.5 (Weak Duality Theorem) *For any constrained optimization problem, $q^* \leqslant f^*$.*

Proof. See [3] □

With these two results we see that the dual problem is always a convex optimization problem, it is not guaranteed however that the objective will be differentiable. If $q^* = f^*$ we say there is no **duality gap**, and in this case solving the dual problem is often as valuable to us as solving the primal problem. Even in the case where there is a duality gap, solving the dual problem gives us a lower bound on the globally minimal objective function value and is therefore useful also. Sometimes even having zero duality gap does not guarantee that we are able to retrieve a minimizer for the primal problem from a maximizer for the dual problem. Typically, to guarantee zero duality gap we need to assume convexity along with other constraint qualifications, but in practice for our non-convex problem we would be happy if we can develop a heuristic enabling us to retrieve primal values which are near a minimizer from dual values which are near a maximizer.

One important question in the theory of duality is as follows: Suppose we can calculate μ^* and $q(\mu^*)$. Under what conditions on the primal objective function and constraints is it possible to get back x^* using the formula

$$x^* \in \arg\min_{x \in X} L(x, \mu^*). \tag{4.3}$$

The answer to this question comes in the Strong Duality Theorem.

Theorem 4.6 (Strong Duality Theorem) *Consider the primal problem formulated in (4.1). Suppose the problem is feasible and its optimal value is finite. Suppose further that X is a convex subset, f and g_j are convex over X for all j. Finally suppose that there exists a $y \in X$ such that $g_j(y) < 0 \, \forall j$, that is to say the interior of the constraint set is non-empty.*

Then there is no duality gap and it is possible to obtain an optimal primal solution using (4.3).

Proof. See ([3],Chapter 5.3). □

Although we do not satisfy the conditions of this Theorem, we will still use the approach using (4.3) to retrieve primal variables from dual variables, and attempt to use some heuristic arguments to get as close to a primal minimizer as possible.

We will now introduce a specific type of constrained optimization problem considered in [3]. Our optimization problems (2.5) and (3.7) are specific cases of this type of problem.

Definition 4.7 *Suppose $\mathbf{x}$ can be broken down into m components, $\mathbf{x}_1, \ldots, \mathbf{x}_m$ of dimensions $n_1, \ldots, n_m$ respectively, and we have an inequality constrained optimization problem of the form*

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{m} f_i(\mathbf{x}_i) && (4.4) \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{m} g_{ij}(\mathbf{x}_i) \leqslant 0, \quad j = 1, \ldots, r, && (4.5) \\ & \mathbf{x}_i \in X_i, \qquad i = 1, \ldots, m, , && (4.6) \end{aligned}$$

where $f_i : \mathbb{R}^{n_i} \mapsto \mathbb{R}$ and $g_{ij} : \mathbb{R}^{n_i} \mapsto \mathbb{R}$ are given functions, and X_i are given subsets of $\mathbb{R}^{n_i}$. We call such a problem a ***Separable, Inequality constrained Optimization Problem.***

This type of problem is interesting to us since it is exactly what we are dealing with. Looking at the most general unicast and multicast problem (3.7) we see that it is in the form of a separable, inequality constrained optimization problem with $m = 2$, $\mathbf{x}_1 = (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$, the set of network and buffer variables, and $x_2 = \mathbf{p}$, the set of power variables. Further, we can say that $X_1 = \mathbf{C}_{\mathbf{b},\mathbf{d},\mathbf{c},\mathbf{f}}$ and $X_2 = \mathbf{C}_{\mathbf{p}}$. Letting j vary over all (e,t) pairs, we can set

$$g_{1j}(\mathbf{x}_1) = \sum_{m \in M} (1 + \kappa^m) f_{e,t}^m$$

and

$$g_{2j}(\mathbf{x}_2) = \Psi_{e,t}(\mathbf{p}).$$

We therefore see that as long as the objective function $\Phi(\mathbf{b}, \mathbf{f}, \mathbf{p})$ is of the form $\Phi_1(\mathbf{b}) + \Phi_2(\mathbf{f}) + \Phi_3(\mathbf{p})$, our optimization problem is of the correct form. This is the case in most commonly used objective functions, such as power minimization.

Having seen that separable, inequality constrained optimization problems are important to us, we now consider the dual problem for such a problem. The dual function can be written as:

$$q(\mu) = \sum_{i=1}^{m} \inf_{x_i \in X_i} \left\{ f_i(x_i) + \sum_{j=1}^{r} \mu_j g_{ij}(x_i) \right\} = \sum_{i=1}^{m} q_i(\mu),$$

where

$$q_i(\mu) = \inf_{x_i \in X_i} \left\{ f_i(x_i) + \sum_{j=1}^{r} \mu_j g_{ij}(x_i) \right\}, i = 1, \ldots, m.$$

The dual problem then becomes:

$$\text{maximize} \quad \textstyle\sum_{i=1}^{m} q_i(\mu) \tag{4.7}$$

$$\text{subject to} \quad \mu \succcurlyeq 0 \tag{4.8}$$

Although the original problem could not be broken up into smaller subproblems due to the coupling constraints being in both power and flow variables, the dual problem can be split into easier to manage subproblems. It has also been observed that even in the non-convex case, separable problems tend to have relatively small duality gaps, meaning one can often obtain a near-optimal primal solution from the dual solution, [3]. This should be useful in the situation we are dealing with.

4.2.2 Solving the Dual Problem

The next problem to be addressed is that of solving the non-smooth convex optimization problem, described in (4.7) and (4.8).

In order to do this we will need the very important concept of a subgradient, a commonly used object in non-smooth optimization.

Definition 4.8 *Let $f : U \mapsto \mathbb{R}$ be a convex function on a convex set U. Then s is a* ***subgradient of*** *f* ***at*** *x_0 if*

$$f(x) \geqslant f(x_0) + (x - x_0)^T s, \forall x \in U.$$

If f is a concave function, we say that s is a subgradient of f at x_0 if $-s$ is a subgradient of the convex function $-f$ at x_0. Or, in other words, for concave f, s is a subgradient of f at x_0 if

$$f(x) \leqslant f(x_0) + (x - x_0)^T s, \forall x \in U.$$

From the definition of $q(\mu)$ it is not immediately clear how one would find a subgradient of q. We shall present a well known result in the field of dual programming, found in [3] for finding subgradients of $q(\mu)$.

Lemma 4.9 *For a given $\mu \in \mathbb{R}^r$, suppose that x_μ minimizes the Lagrangian $L(x, \mu)$ over $x \in X$, that is*

$$x_\mu \in \arg\min_{x \in X} L(x, \mu) = \arg\min_{x \in X} \{f(x) + \mu^T g(x)\},$$

or, equivalently, for that fixed μ, $q(\mu) = L(x_\mu, \mu)$. Then $g(x_\mu)$ is a subgradient of the dual function q at μ, that is to say

$$q(\overline{\mu}) \leqslant q(\mu) + (\overline{\mu} - \mu)^T g(x_\mu), \forall \overline{\mu} \in \mathbb{R}^r.$$

Proof. See [3]. □

The important thing to notice with this is that in order to evaluate $q(\mu)$ for a given μ, we need to know x_μ. This means we can calculate a subgradient, $g(x_\mu)$ at almost no additional cost. Now we know how to calculate $q(\mu)$ and a subgradient, we look at a method for solving the dual problem, that is maximizing $q(\mu)$ over all μ in the non-negative orthant.

4.2.3 Subgradient Methods for Non-Smooth Optimization

We will use subgradient information to generate a sequence of dual feasible points by using the iteration:

$$\mu^{k+1} = [\mu^k + \alpha^k s^k]^+. \tag{4.9}$$

Here s^k is the subgradient $g(x_{\mu^k})$, α^k is a positive scalar stepsize, and $[.]^+$ denotes projection onto the non-negative orthant. It can be shown that under certain conditions on the stepsize, this sequence converges to μ^*.

Lemma 4.10 *If α^k satisfies $\sum_{k=0}^{\infty} \alpha^k = \infty$, and $\alpha^k \to 0$ then the subgradient method converges to a μ which is a maximizer of the dual problem, that is $\lim_{k \to \infty} \mu^k = \mu*$, and $q(\mu^*) = q^*$.*

Proof. See [3]. □

We will now use the subgradient method applied to the dual of our original primal optimization problem.

4.3 Application to Our Unicast Problem

First we will remind ourselves of the primal problem we wish to solve. For ease of explanation, we will focus on using the dual decomposition method to solve the unicast problem of chapter two rather than the generalized unicast and multicast problem of chapter three. This is due to the main points of interest where the dual problem is concerned are the coupling constraints. Studying the unicast problem is illuminating enough to show us how the nonconvexity in the coupling constraints affects the performance of the subgradient method. To further simplify the problem but retain the nonconvex coupling constraints we shall only consider one message, and not consider the problem of scheduling. We no longer consider individual timeslots and instead consider average powers and data flows, and do not consider buffering or individual time slots. This greatly decreases the number of primal and dual variables to be considered, but in terms of complexity, all we have lost are some linear and box constraints. If we can solve the simplified version we can solve the more complicated problem by identical methods, but for the remainder of this chapter we will only consider the simpler version. Suppose the message of size S is sourced at

vertex s and destined for vertex d. Our simplified problem becomes:

$$\begin{array}{rcll}
\text{minimize} & \sum_{e\in E} p_e & & \\
\text{such that} & & & \\
p_e, f_e & \geqslant & 0 & e \in E, \\
\sum_{e\in E^+(v)} p_e & \leqslant & P_v & v \in V, \\
p_e & \leqslant & P_e & e \in E, \\
\sum_{e\in E^+(s)} f_e & = & S, & \\
\sum_{e\in E^-(d)} f_e & = & S, & \\
\sum_{e\in E^+(v)} f_e - \sum_{e\in E^-(v)} f_e & = & 0 & v \in V, \\
f_e - B\tau \log_2 \left(1 + \frac{a_{e,e}p_e}{\sum_{l\neq e\in E^{c(e)}} a_{l,e}p_l + BN_0}\right) & \leqslant & 0 & e \in E.
\end{array} \tag{4.10}$$

or more succinctly:

$$\begin{array}{ll}
\text{minimize} & \Phi(\mathbf{p}, \mathbf{f}) = \sum_{e\in E} p_e \\
\text{such that} & \mathbf{f} \in \mathbf{C_f} \\
& \mathbf{p} \in \mathbf{C_p} \\
& f_e \leqslant B\tau \log_2(1 + \frac{a_{e,e}p_e}{\sum_{l\in E\setminus e} a_{l,e}p_l + BN_0})\ e \in E.
\end{array}$$

This problem concerns wireless routing and power control for a single unicast transmission, and does not consider scheduling into time slots or multiple unicast transmissions. We feel it is complex enough to illustrate the algorithm we are using.

In order to apply the dual decomposition approach, we need to decide which constraints to put into the Lagrangian and which to leave out. Since the link capacity constraints involve $\mathbf{f}$ variables as well as $\mathbf{p}$ variables, these constraints need to be included in the Lagrangian in order for us to decompose the problem into a subproblem in the $\mathbf{f}$ variables, and a subproblem in the $\mathbf{p}$ variables. We will leave the linear constraints out of the

Lagrangian, and the sets X_i discussed in the section on dual decomposition will be $\mathbf{C_f}$ and $\mathbf{C_p}$. With this in mind, our Lagrangian is:

$$L(\mathbf{f}, \mathbf{p}, \mu) = \sum_{e \in E} p_e + \sum_{e \in E} \mu_e \left(f_e - B\tau \log_2 \left(1 + \frac{a_{e,e} p_e}{\sum_{l \in E \setminus e} a_{l,e} p_l + BN_0} \right) \right),$$

and the dual function we need to consider is

$$q(\mu) = \min_{\mathbf{f} \in \mathbf{C_f}, \mathbf{p} \in \mathbf{C_p}} L(\mathbf{f}, \mathbf{p}, \mu).$$

We decompose the Lagrangian and dual function into two parts, one in the network variables $\mathbf{f}$ and one in the communication variables $\mathbf{p}$:

$$L_{\text{net}}(\mathbf{f}, \mu) = \sum_{e \in E} \mu_e f_e \tag{4.11}$$

$$L_{\text{comm}}(\mathbf{p}, \mu) = \sum_{e \in E} \left(p_e - \mu_e B\tau \log_2 \left(1 + \frac{a_{e,e} p_e}{\sum_{l \in E \setminus e} a_{l,e} p_l + BN_0} \right) \right) \tag{4.12}$$

$$L(\mathbf{f}, \mathbf{p}, \mu) = L_{\text{net}}(\mathbf{f}, \mu) + L_{\text{comm}}(\mathbf{p}, \mu). \tag{4.13}$$

With this in mind, we can now decompose our dual function,

$$q_{\text{net}}(\mu) = \min_{\mathbf{f} \in \mathbf{C_f}} L_{\text{net}}(\mathbf{f}, \mu), \tag{4.14}$$

$$q_{\text{comm}}(\mu) = \min_{\mathbf{p} \in \mathbf{C_p}} L_{\text{comm}}(\mathbf{p}, \mu), \tag{4.15}$$

and the dual problem we have to solve is:

$$\text{maximize} \quad q(\mu) := q_{\text{net}}(\mu) + q_{\text{comm}}(\mu) \tag{4.16}$$

$$\text{such that} \quad \mu \geqslant 0. \tag{4.17}$$

We now present a pseudo-code of the algorithmic method for solving our problem through dual decomposition.

4.4 Algorithm for Solution Through Dual Decomposition

In the psuedocode we make use of some shorthand notation, namely

$$\Theta_e(f_e, \mathbf{p}) = f_e - B\tau \log_2 \left(1 + \frac{a_{e,e}p_e}{\sum_{l \neq e \in E^{c(e)}} a_{l,e}p_l + BN_0}\right)$$

represents the difference between flow on a given edge, and capacity on that edge, and is used to calculate the subgradient.

$$\Lambda_{\mathbf{f}}^k = \min_{f \in \mathbf{C_f}} L_{\text{net}}(\mathbf{f}, \mu^{\mathbf{k}})$$

and

$$\Lambda_{\mathbf{p}}^k = \min_{p \in \mathbf{C_p}} L_{\text{comm}}(\mathbf{p}, \mu^{\mathbf{k}})$$

are the minimum objective function values of the two decomposed parts of the Lagrangian for a given μ^k. With this in mind we proceed to describe the algorithm:

1. Define parameters.
2. Set $k = 0$ and initialize $\mu^{\mathbf{0}}$ and $\mathbf{s^0}$
3. While stopping criteria not met:
    - Solve $\min_{\mathbf{f} \in \mathbf{C_f}} L_{\text{net}}(\mathbf{f}, \mu_{\mathbf{k}})$,
    - Set $[\mathbf{f^k}, \mathbf{\Lambda_f^k}] = [\arg\min_{\mathbf{f} \in \mathbf{C_f}} L_{\text{net}}(\mathbf{f}, \mu_{\mathbf{k}}), \min_{\mathbf{f} \in \mathbf{C_f}} L_{\text{net}}(\mathbf{f}, \mu_{\mathbf{k}})]$.
    - Solve $\min_{\mathbf{p} \in \mathbf{C_p}} L_{\text{comm}}(\mathbf{p}, \mu_{\mathbf{k}})$,

- Set $[\mathbf{p^k}, \mathbf{\Lambda_p^k}] = [\arg\min_{\mathbf{p}\in\mathbf{C_p}} L_{\text{comm}}(\mathbf{p}, \mu_\mathbf{k}), \min_{\mathbf{p}\in\mathbf{C_p}} L_{\text{comm}}(\mathbf{p}, \mu_\mathbf{k})]$.
- Note that $q((\mu^k)) = \mathbf{\Lambda_f^k} + \mathbf{\Lambda_p^k}$.
- Define $s_e^k = \Theta_e(f_e^k, \mathbf{p^k})$.
- Set $k = k + 1$.
- Update $\mu^\mathbf{k} = [\mu^{\mathbf{k-1}} + \alpha_k \mathbf{s^{k-1}}]_+$

4. Record k_{end} as the k value when the while loop terminates.

5. $\mu^* := \mu^{k_{\text{end}}}$

6. $[\mathbf{f}^*, \mathbf{\Phi_f^*}] = [\arg\min_{\mathbf{f}\in\mathbf{C_f}} L_{\text{net}}(\mathbf{f}, \mu^*), \min_{\mathbf{f}\in\mathbf{C_f}} L_{\text{net}}(\mathbf{f}, \mu_*)]$

7. $[\mathbf{p}^*, \mathbf{\Phi_p^*}] = [\arg\min_{\mathbf{p}\in\mathbf{C_p}} L_{\text{comm}}(\mathbf{p}, \mu^*), \min_{\mathbf{p}\in\mathbf{C_p}} L_{\text{comm}}(\mathbf{p}, \mu_*)]$

Note that s_e^k can be interpreted as the excess capacity on link e, that is the difference between the capacity provided by the current power configuration in the communication layer, and the amount of data transported along the link in the current configuration of the network flow layer. If we interpret the dual variable μ_e^k as the price per unit capacity of link e, the dual problem has an interesting interpretation. Given the prices μ^k, the network layer solves the network flow problem, (4.14), of minimizing the total cost of link capacities used. The communication resource layer solves the problem, (4.15), of minimizing the total cost function discounted by the revenue received from capacities that it supports. Interaction between the two layers is co-ordinated by the master dual problem, (4.16), through the vector of prices. The subgradient method can therefore be thought of as rule for updating the prices in order to arrive at the optimal co-operation between the two layers. Before we proceed to numerically test the algorithm, we need to discuss some computational issues thrown up during the design of the algorithm.

4.4.1 Implementation Issues

Although this appears to be a relatively straightforward algorithm, there are still several technical issues and questions to be confronted. Firstly, how do we update our stepsize, α_k? For simplicity we will set $\alpha_k = \rho/k$ for some $\rho \in \mathbb{R}$ in order to guarantee that we satisfy Lemma 4.10. Secondly, and most importantly, how will the non-convexity affect the convergence of our algorithm? If we cannot guarantee that we have solved the subproblems optimally, can we still guarantee that we have found a subgradient? Can we use a heuristic to get from a non-optimal dual solution to a near-optimal primal solution? We find the best way to confront these issues is through numerical experimentation.

We test our algorithm in MATLAB, on two different networks. Firstly, the original very simple network discussed in chapter two with two edges and two nodes. We use this for testing because we fully understand the behaviour of the objective function for different parameter choices based on the work we did in Section 2.5. Secondly, the backhaul 1-3-5-1 network as introduced in Section 3.8, but with a single destination node rather than three.

In order for the algorithm to find a subgradient, we need to be able to minimize L_{net} and L_{comm} in equations (4.11) and (4.12) for fixed dual variable μ, subject to the linear constraints on $\mathbf{p}$, $\mathbf{f}$ and $\mathbf{b}$. Now, minimizing $L_{\text{net}}(\mathbf{b}, \mathbf{f}, \mu)$ over a linear constraint set is a straightforward linear optimization problem, but minimizing $L_{\text{comm}}(\mathbf{p}, \mu)$ over a linear constraint set is a non-convex optimization problem, and so we cannot always hope to find an optimal solution. If it is the case that the $\mathbf{p}$ returned by this subproblem is not a minimizer, we can not be sure that $s_e^k = \Psi_e(\mathbf{c^k}, \mathbf{p^k})$ is in fact a subgradient of the dual problem.

For zero interference we know that the problem becomes convex. When we introduce high levels of interference, the subproblem of minimizing $L_{\text{comm}}(\mathbf{p}, \mu)$ becomes highly non-convex and the black box solver we use to solve the subproblem is no longer guaranteed to find a global minimizer. In this case, we are not guaranteed to find a subgradient at each

step and can expect to take some steps away from the optimum in the dual space. One way to prevent against this is to check at each step of the algorithm whether an improvement is registered in the dual function value. If there is no improvement or even a decline, one can reject the current solution and attempt again to solve the non-convex subproblem from a different starting point. In practice, this greatly slows down the algorithm, and as will be shown in the results that follow, it is possible to allow the dual algorithm to take steps which decrease the dual function value and still reach the maximal dual function value in reasonable time.

We investigate empirically how often the solver fails to find a global minimizer to the non-convex subproblem, (4.12), by first fixing all the problem parameters and the dual variables, and then attempting to solve the problem using the MATLAB black box nonlinear solver a number of times using different starting points for the primal variables. We then change the parameters and repeat. For each problem instance, we calculated the true global optimal point using a heuristic global solver, and recorded how often the black box solver attained the optimal function value.

For this particular test, we use the 1-3-5-1 backhaul network, and set $B = \tau = 1$, $N_0 = 0.1$. For the gain matrix we set $a_{i,i} = 1$ for all i, and $a_{i,j}$ is a uniformly distributed random number between 0 and R, where we use R to control the level of interference in the network. Dual variable values were chosen at random between 0 and 10, as were initial variables.

With R set to 0, there is no interference and therefore the subproblem should have a unique global minimizer. This is supported by the data in which the global minimizer is found in every problem instance with every choice of starting point.

With R set to 0.1, there is a small amount of interference, and one cannot be sure whether or not the problem has local minimizers. The data in this case again shows that the black box solver always finds the global minimizer under various different choices of

dual function value and initial primal function value.

With R set to 0.7, there is more interference in the network, and we can expect the problem to have several local minimizers. Indeed, the black box solver rarely finds a global minimizer in this case, finding one in approximately 35% of our experiments.

For tables of the data from this experiment to find when the subproblem returns an optimal solution see tables C.1, C.2 and C.3 in Appendix C. This investigation suggests that in networks suffering low or zero interference, the subgradient algorithm should proceed to a maximizer of the dual problem in a monotone fashion. In networks suffering high interference, we cannot expect such monotone progress of the dual objective function value.

4.4.2 Application to the Simplest Problem

We first test the algorithm on the simple problem introduced in Section 2.5, in the symmetrical case with $a_{11} = a_{22} = 1$ and $a_{12} = a_{21} = 0.1$. We set $B = N_0 = \tau = 1$ with a message of size one. By the work we did in the second chapter, we know that the optimal solution is to route half the message down each of the two paths, with equal power allocated to each path and a total power allocation of 0.86 Watts. We set the stopping criterion for the dual decomposition algorithm to be that whenever two consecutive dual values are within a certain tolerance, the algorithm terminates. In Figure 4.2 we show the convergence of the dual objective function to the minimizing value of the primal objective function for various step size choices. The graph clearly shows that a judicious choice of stepsize is important to achieve fast convergence. Even in the simple case we are exploring where the only stepsize parameter is ρ, and the stepsize at step k is defined as ρ/k, if the stepsize parameter is too small then convergence may be extremely slow, and since the stepsizes are always decreasing we may find that the function values of successive steps in the algorithm are close even though we have not reached our solution and so the algorithm

terminates prematurely. One would also think that increasing the stepsize would increase the rate of convergence, but again the graph shows that this is not the case. Increasing the stepsize parameter ρ from 2 to 5 decreases the rate in this example. One important thing to note is that although the algorithm has converged towards an optimal dual value μ^*, the primal value that we get by solving $(f, p) = \arg\min L(f, p, \mu^*)$ is infeasible. Using a simple heuristic however we are able to get back a near optimal solution. For example, in the case where $\rho = 2$, although the dual function value was close to the optimal primal value, the algorithm returned values of $\mathbf{p} = (0.42, 0.44)^T$, $\mathbf{c} = (1, 0)^T$. Since the subproblem in the $\mathbf{c}$ variables is a linear program, the solutions will be at corners of the feasible region. Corners of the feasible region in question are the points $(1, 0)$ and $(0, 1)$. In the problem in question, the optimal dual variables are equal, and so the entire feasible region is optimal for this subproblem, but the algorithm naturally chooses an end point. In order to get back to a near optimal solution from this infeasible one, we can simply assume that all the coupling constraints are active, and recalculate the $\mathbf{f}$ variables from the $\mathbf{p}$ variables. Using this approach we arrive at the solution $\mathbf{p} = (0.42, 0.44)^T$, $\mathbf{f} = (0.49, 0.51)^T$, which is sufficiently close to the optimal solution of $\mathbf{p} = (0.43, 0.43)^T$, $\mathbf{f} = (0.5, 0.5)^T$.

Still considering the simplest graph, with two nodes and two edges, we now look at the case where we have two local solutions, letting $a_{11} = 1$, $a_{12} = 0.4$, $a_{21} = 0.6$, $a_{22} = 0.9$. As discussed in Section 2.5.6, we know that this problem has a local minimizer at $\mathbf{f} = (0, 1)^T$, and a global minimizer at $\mathbf{f} = (1, 0)^T$ with optimal primal function value of 1. Using the dual decomposition algorithm we see in Figure 4.3 that the dual function value converges to the optimal primal function value, but the convergence is no longer monotone. The values arrived at for the primal variables using this algorithm were $\mathbf{f} = (1, 0)^T$ and $\mathbf{p} = (0.99, 0)^T$. After correcting to ensure feasibility of the coupling constraints we get that $\mathbf{f} = (0.99, 0)^T$, but find that this no longer satisfies the total flow constraint.

Even with these very simple problems we see that using the subgradient algorithm on

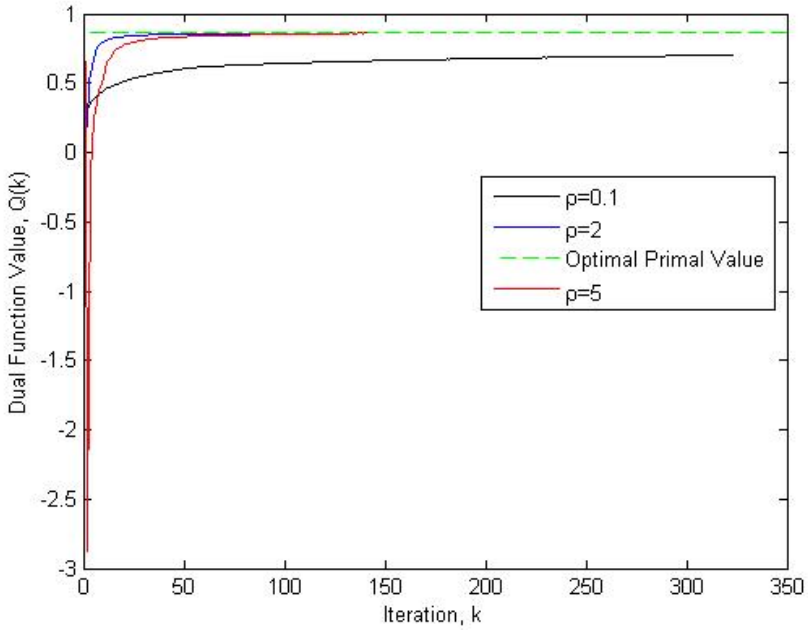

Figure 4.2: *Convergence of the dual function for varying stepsizes in the simple case with a unique global minimizer*

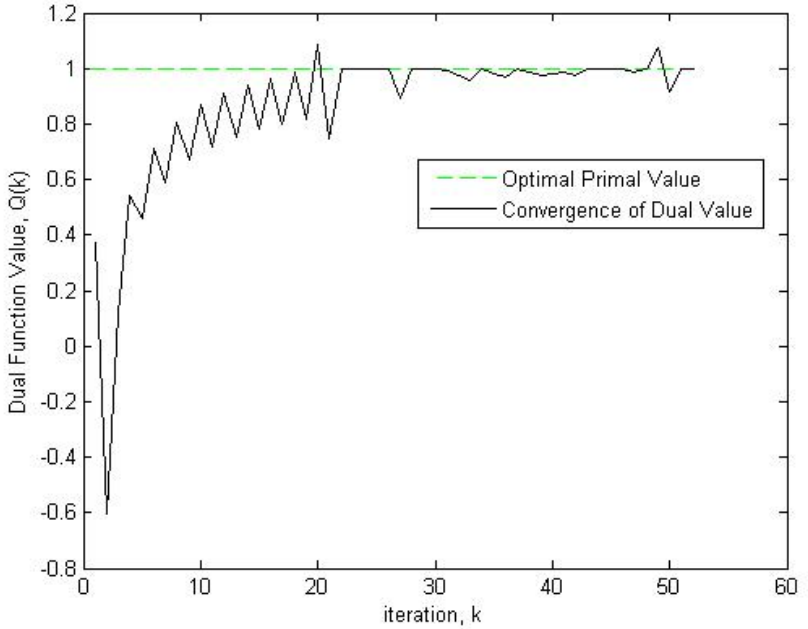

Figure 4.3: *Convergence of the dual function when we have a local and global minimizer*

the dual problem to our problem, we do not necessarily reach a feasible solution. We do however seem to reach near optimal power variables and can therefore hope to take these near optimal power variables to find near optimal flow variables. In the next section we shall apply the algorithm to a more useful network structure.

4.4.3 Application to the 1-3-5-1 Network

We now apply the dual decomposition algorithm to the 1-3-5-1 network introduced in Section 3.8 as a network important in wireless backhaul networks. Firstly, we apply the algorithm in the case where we have no cross channel interference, with $B = 1$, $N_0 = 0.1$ and $S = 10$. In Figure 4.4 we see that there is zero duality gap, but again for reasons discussed above, the algorithm returns an infeasible primal solution. The first two columns of Table C.4 in Appendix C show the returned network flow variables and power variables for the 23 edges in the network. The values are clearly infeasible. Using the heuristic developed in applying the algorithm in the simple case, we get the adjusted network flow variables in the third column. This provides a near optimal solution to the problem, as can be seen by comparing with the fourth and fifth columns of table C.4.

We go on to test the algorithm for high interference, with a Gain matrix with the off diagonal terms randomly distributed between zero and 0.5. We see the dual function value converges to a number reasonably close to the optimal primal function value, see Figure 4.5, but again our method returns infeasible primal variables, see the first two columns of Table C.5 in Appendix C for the flow variables and power variables returned. Assuming that the coupling constraints are active at the solution, we use the power variables returned by the algorithm to calculate corresponding flow variables. We see that although there is a duality gap, we come close to getting back optimal primal variables, comparing the first and third columns to the fourth and fifth columns of Table C.5. These results are promising for our method, but of course in future work we need to test the

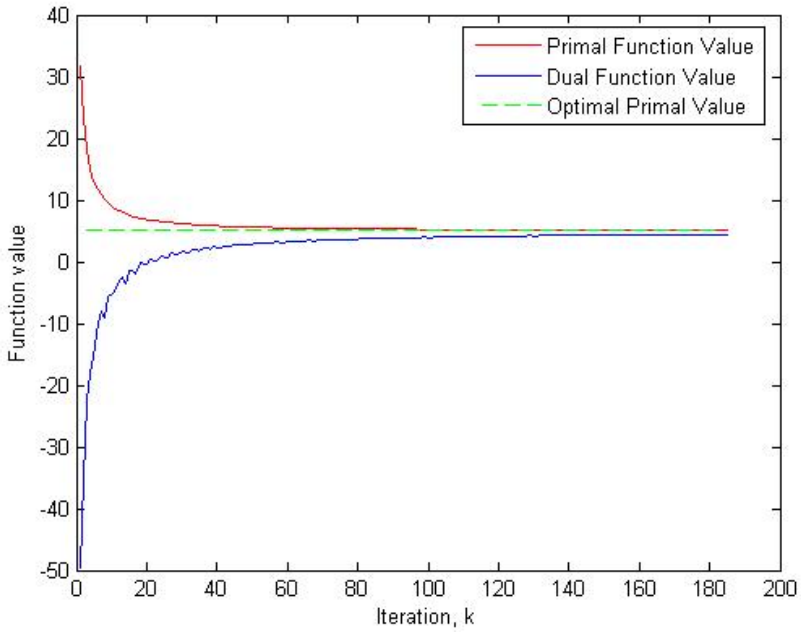

Figure 4.4: *Convergence of the dual and primal function values in the 1-3-5-1 network with zero interference*

algorithm using a gain matrix that represents a physically realistic situation. One problem that needs considering is that although the solutions we obtain are close to optimal, the routing variables do not satisfy the network flow constraints. This could potentially be addressed by introducing a final correction step that adjusts the network flow variables as little as possible but then satisfies the flow constraints, and then finally readjusting the power variables accordingly.

Having implemented the subgradient algorithm numerically, we see that its convergence is slow, often taking hundreds of iterations to get close to maximal dual variables. In the final part of this chapter we shall look at an acceleration technique to try and reduce the number of iterations required.

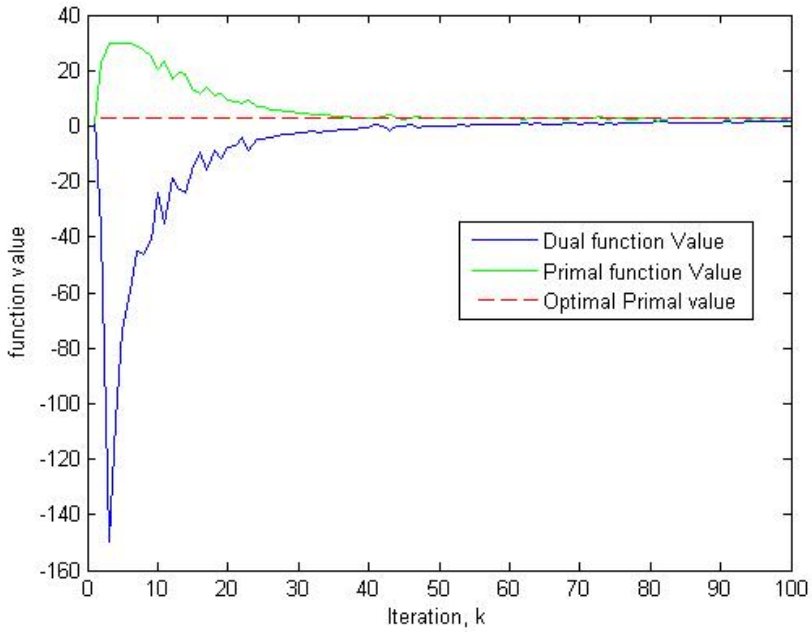

Figure 4.5: *Convergence of the dual and primal function values in the 1-3-5-1 network with high interference*

4.5 An Acceleration Technique

Since each step of the subgradient algorithm involves solving two optimization problems, each getting harder as we increase the size of the network, it would be desirable to accelerate the convergence of the dual variables to their maximizer. This could potentially be done by using more sophisticated techniques for choosing the step size in updating the dual variables, or using information about previous values of the dual variables to improve our understanding of where the dual variables are converging to. In this chapter we will discuss an acceleration technique based on Aitken's $\triangle^2$ method, using Steffensen's fixed point iterations. This technique significantly increases the convergence rate of the algorithm discussed in the previous chapter under certain conditions. We will look at the origins of this method, as well as ways in which we can implement it. The ideas of Aitken and Steffensen are well established, see [8].

4.5.1 Aitken's Method and Steffensen's Method

The idea behind Aitken's $\triangle^2$ method is that, given a linearly convergent sequence, we seek to create a sequence which converges faster to the same limit. To this end, consider a sequence $\{x_k\}$ which converges linearly to x at rate A, that is to say,

$$\lim_{k\to\infty} \frac{|x_{k+1} - x|}{|x_k - x|} = A. \tag{4.18}$$

Assume for now that equation (4.18) holds without the moduli. Then, we have that, for large k at least,

$$\frac{x - x_{n+1}}{x - x_n} \approx \frac{x - x_{n+2}}{x - x_{n+1}} \approx A.$$

Solving this for x, we will construct a sequence which uses the convergence properties of the sequence $\{x_n\}$ in order to construct a sequence which converges to x faster. Rearranging, we have that

$$(x - x_{n+1})^2 \approx (x - x_n)(x - x_{n+2}),$$

Expanding, and cancelling x^2 on both sides, we have that

$$x \approx \frac{x_n x_{n+2} - x_{n+1}^2}{x_{n+2} - 2x_{n+1} + x_n} := y_n.$$

This can be rearranged to give

$$y_n = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}.$$

This will be our definition of the Aitken sequence. It can be shown that for certain linearly convergent sequences, $\{y_n\}$ converges to x faster than $\{x_n\}$ in the sense that

$\lim_{n\to\infty} |\frac{x-y_n}{x-x_n}| = 0$, [21].

Rather than calculating the linearly converging sequence x_n and then in parallel calculating the faster converging sequence y_n, one can use the same idea at each step and hopefully construct an even faster converging sequence. This is known as Steffensen's acceleration.

Definition 4.11 *When Aitken's $\triangle^2$ method is combined with fixed point iteration, the result is known as* ***Steffensen's acceleration****.*

Suppose we are seeking the fixed point x^* where $x^* = f(x^*)$, using the fixed point iteration $x_{k+1} = f(x_k)$. We compute the Steffensen accelerated sequence as follows:

1. Given x_k, let $\widetilde{x}_{k+1} = f(x_k)$.
2. Let $\widetilde{x_{k+2}} = f(\widetilde{x}_{k+1})$
3. Calculate x_{k+1} by applying the Aitken's acceleration formula to $x_k, \widetilde{x}_{k+1}, \widetilde{x}_{k+2}$.
4. Set $k = k + 1$ and repeat.

4.5.2 Application of Acceleration Methods to Our Algorithm

Unfortunately, we cannot guarantee linear convergence for our dual variables. Our dual variables converge sublinearly, and since we are projecting onto the non-negative orthant, it is entirely possible that a dual variable takes the value zero more than twice in a row. This would give us a divide by zero error in the application of the Aitken formula. All of this means that we have to be very careful in the implementation of these ideas. Nevertheless, under certain conditions we are still able to observe marked improvement in the convergence of our algorithm through applying these techniques. There are two possible ways of applying these ideas, one is to directly apply Aitken's $\triangle^2$ Method, the other is to apply Steffensen's iteration.

Aitken's Method

The first approach is to run the standard algorithm to calculate our converging sequence of dual variables, and in parallel apply Aitken's $\triangle^2$ Method to calculate a sequence of dual variables that converges faster. We build in a measure which ensures we never have a divide by zero error, and if we have a repetition of zeros in a given dual variable, we allow the accelerated dual variable to be registered as a zero also. We then apply the stopping criteria to both the original sequence and the accelerated sequence. Although this approach does not appear to save on computation since we need to do twice as many function evaluations, we can gain something if the accelerated sequence reaches the stopping criteria in less than half as many steps as the original sequence.

In Figures 4.6 and 4.7 we present experimental data comparing convergence of the dual function for the standard sequence and the accelerated sequence. In both cases we use the 1-3-5-1 graph as used in the previous section and calculate the dual function value at each step in the accelerated sequence. In Figure 4.6 we assume no interference, and in Figure 4.7 we assume low interference, randomly distributed between zero and 0.1. In both cases it can indeed be seen that the accelerated sequence converges faster and in the low interference case, the improvement is significant and the convergence becomes monotonic as compared to the sawtooth like convergence of the original sequence.

Clearly, under certain conditions, with appropriately chosen stopping conditions, there is something to be gained by using Aitken's $\triangle^2$ method in its pure form.

Steffensen's Acceleration

The second approach is to use Steffensen's Acceleration. With this approach we update our dual variables using two subgradient steps, and then use Steffensen's method to jump closer to the solution. We test it on the simplest possible network with two nodes and two edges, as introduced in Section 2.5, in order to evaluate its performance. We apply

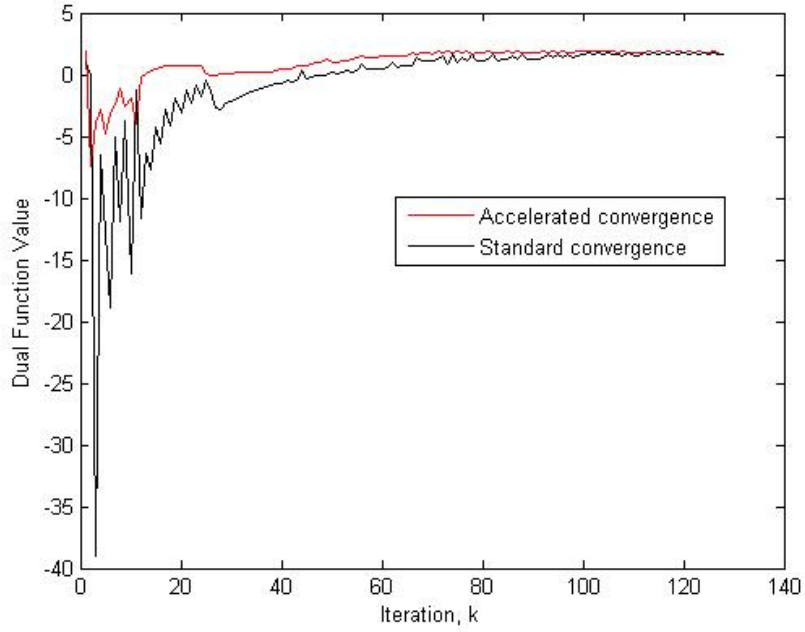

Figure 4.6: *Convergence of the dual function value using Aitken acceleration with no interference*

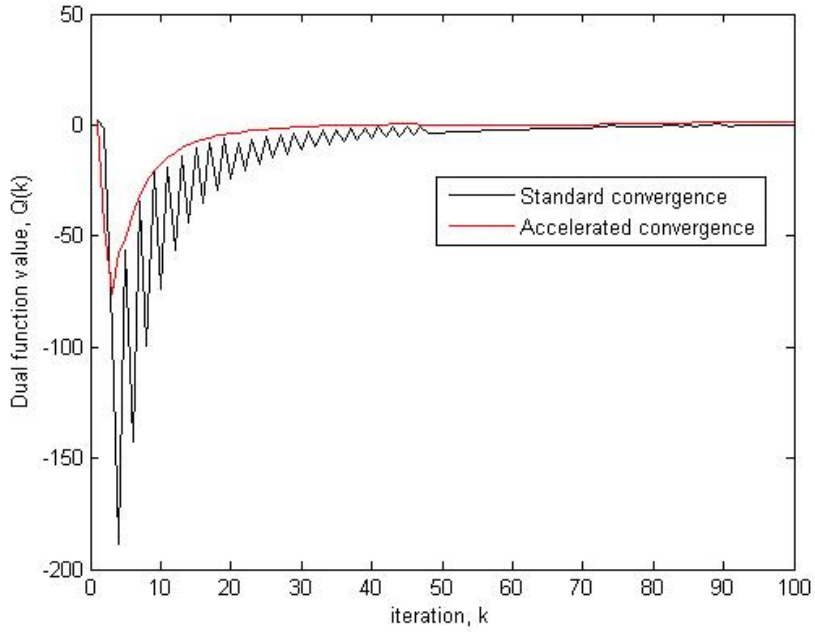

Figure 4.7: *Convergence of the dual function value using Aitken acceleration with low interference*

the acceleration technique in the case where there is a unique global minimizer. As can be seen in Figure 4.8, if the convergence of original algorithm is slow, we can significantly improve the convergence using Steffensen's Acceleration. Since each step in the accelerated algorithm requires two standard subgradient steps as well as an extra simple calculation, to gain anything by using it we would need to converge in less than half as many steps as the standard algorithm. In Figure 4.8, we see that this is the case and there are indeed significant gains to be made by using Steffensen's Acceleration.

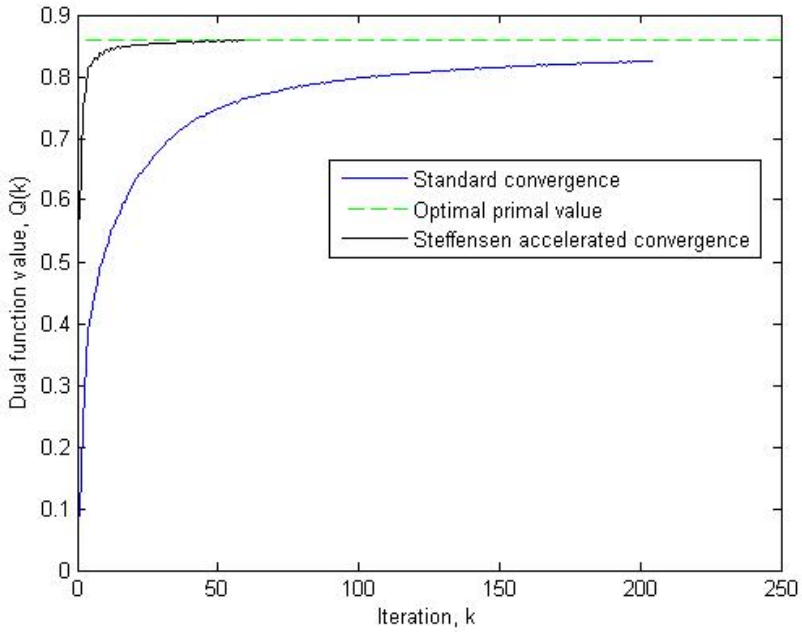

Figure 4.8: *Convergence of the dual function value using Steffensen's Acceleration*

Since the convergence of the subgradient method is sublinear and not monotone, we need to be careful in our implementation of Steffensen's Acceleration since the assumptions made in the development of the Aitken step are no longer valid. Allowing the accelerated algorithm to run with no safe guards, we sometimes run into problems, see Figure 4.9. Although initially the accelerated algorithm significantly outperforms the standard algorithm, it soon fails and seems to be stuck at a suboptimal dual value. This

problem can be solved by only applying the accelerated algorithm as long as improvements are being made to the dual function value. If after a certain number of steps there has been no improvement to the current best dual function value, we return to the best current known dual variable values, "switch off" the acceleration method and revert to the standard algorithm. In effect, we give the algorithm a head start towards the optimal value using Steffensen's Acceleration.

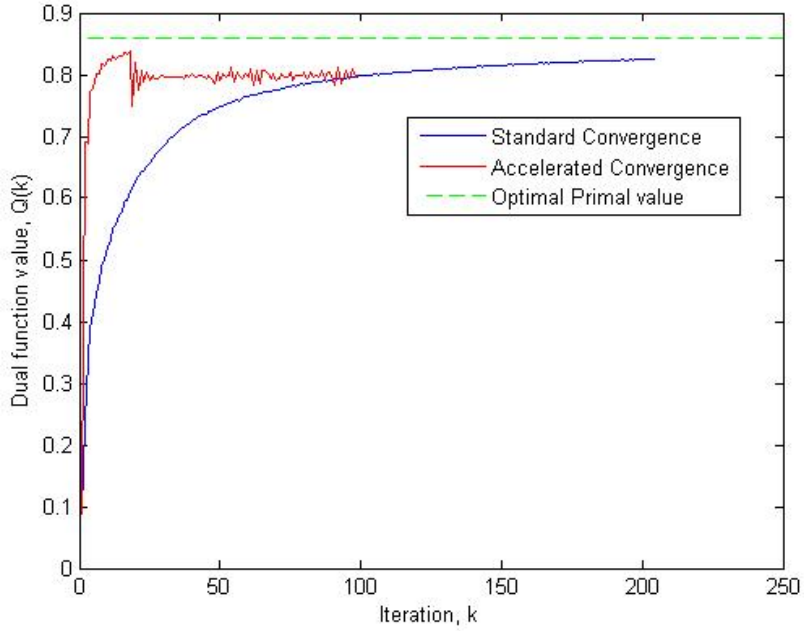

Figure 4.9: *Failure of Steffensen's Acceleration*

Another useful observation is that a judicious choice of step-size limits the gains we achieve by using Steffensen's acceleration. In Figure 4.10 we compare convergence of the dual function value using Steffensen's acceleration and the standard subgradient approach. We choose the subgradient stepsize parameter $\rho = 1$, which we found empirically to result in the fastest convergence (see Figure 4.2). In this case there is very little to be gained by using Steffensen's acceleration.

Both Aitken's $\triangle^2$ method and Steffensen's acceleration provide potential benefits with

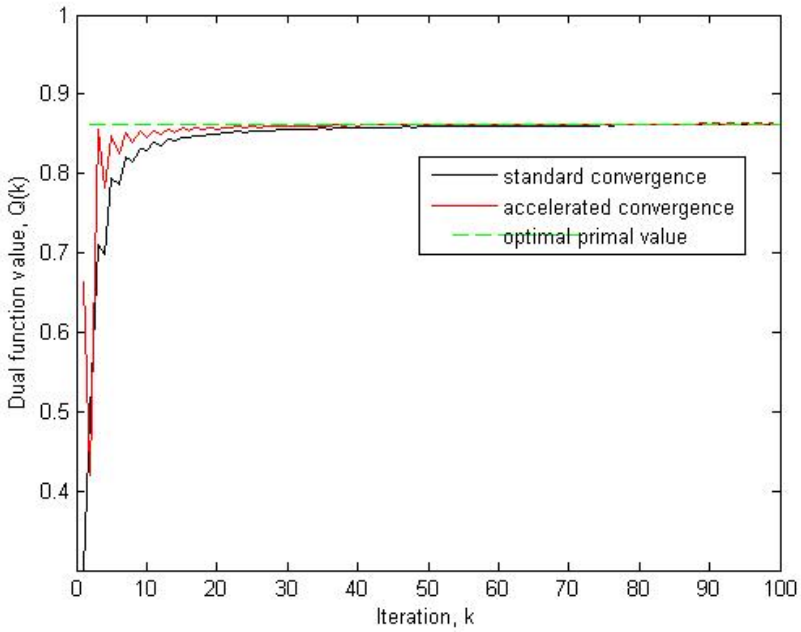

Figure 4.10: *Steffensen's acceleration provides small gains with appropriately chosen step-size parameter*

regard to the convergence of our algorithm. Further work is required to fully understand the gains we stand to make.

In this chapter we have introduced a subgradient method for solving the dual of our optimization problem. We found that the algorithm often finds near optimal power values but struggles to find a complete optimal, or even feasible solution. In some cases it was possible to get around this through the use of a heuristic based on the fact that we know all coupling constraints are active at an optimal solution.

Finally, we looked at ways of improving the poor convergence rate of the algorithm with considerable success. Further research is required to test these techniques on realistic network environments, as well as reintroducing the various variables and constraints we withheld to simplify the problem. In order to make this approach practical we would also need a more complete heuristic detailing the best way to get from the solution of the subgradient algorithm to the best possible feasible solution of the problem.

In the next chapter we will investigate a promising algorithmic technique based on primal co-ordinate descent which does not suffer from the slow convergence of the dual subgradient method, but still decomposes the problem into easier to handle subproblems.

Chapter 5

A Primal Co-ordinate Descent Approach

5.1 Motivation

In the previous chapter we discussed an approach to solving the optimization problems outlined in chapters two and three. We observed that its main problems are its slow rate of convergence, and the fact that we need to run the algorithm to optimality to have a chance of finding a feasible point in the primal domain. Its strength lies in the decoupling of the problem into two types of subproblem. We set out now to investigate a method which hopefully alleviates these negatives while still drawing on the positive aspects. To this end we consider again a general formulation of the problem which can encompass both the unicast and multicast problems. Throughout this chapter, this formulation will be known as (P1):

$$\begin{aligned} \min \quad & \Phi(\mathbf{p}, \mathbf{b}, \mathbf{f}) \\ \text{s.t} \quad & \mathbf{p} \in \mathbf{C_p} \\ & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \\ & \textstyle\sum_{m \in M} f_{e,t}^m \leqslant \Psi_{e,t}(\mathbf{p}) \; e \in E, t \in T. \end{aligned} \tag{5.1}$$

Here we assume Φ is monotone in $\mathbf{p}$, $\mathbf{C_p}$ and $\mathbf{C_{b,c,d,f}}$ are the specific polyhedral constraint sets detailed in Chapter 3, and Ψ is the coupling constraint of previous chapters.

The key motivating observation we need here is as follows: The variables of this problem can be split into flow variables $(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$ and physical communication resource variables $\mathbf{p}$, and if we fix either set of variables the problem we are left with closely resembles, or is equivalent to an existing well studied optimization problem. Specifically, for fixed power variables, we can reformulate the problem as a nonlinear minimum cost multi-commodity flow problem, and for fixed flow variables we have a standard power control problem. As can be seen in Section 5.3, both of these subproblems can be formulated as convex optimization problems, and many solution methods exist for solving these subproblems in a distributed manner.

To make use of the apparent simplicity of the subproblems, and to avoid the drawbacks we found in the dual approach, it would be desirable to employ a primal block co-ordinate descent method where we alternately fix the physical communication variables, $\mathbf{p}$, and solve the problem over the network flow variables $(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$, and then fix the $(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$ variables and solve over the $\mathbf{p}$ variables. The idea would be to toggle between these two problems until an optimal point is found.

As the following simple example shows however, caution is required when employing co-ordinate descent methods in constrained optimization since even in convex optimization problems with a unique global minimizer, co-ordinate descent methods may not converge to it.

Example 5.1 *We present a simple linear optimization problem which cannot be solved*

using a co-ordinate descent method.

$$\begin{aligned} \min \quad & x + y \\ \text{s.t.} \quad & x, y \geqslant 0 \\ & 2x + y - 1 \geqslant 0 \end{aligned}$$

Clearly the optimal function value is 0.5, attained at $(0.5, 0)$*, but if we apply co-ordinate descent, minimizing over* x *first, starting from any feasible* (x^0, y^0) *with* $y^0 > 0$ *we see from figure 5.1 that the algorithm does not converge to the optimal solution, and instead terminates after one step at a suboptimal point* (x^1, y^0)*, since through minimizing in the* x *co-ordinate, we have activated the constraint* $2x + y - 1 \geqslant 0$*, and it becomes impossible to make any progress in the* y *co-ordinate direction.*

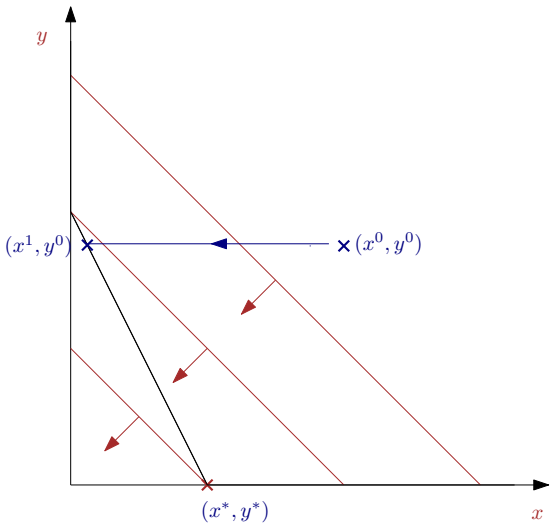

Figure 5.1: *A simple linear program which cannot be solved by co-ordinate descent methods*

Another problem which arises when using co-ordinate descent methods is that the objective function may not contain any terms in a certain co-ordinate. Obviously, a minimization step in this co-ordinate direction cannot hope to achieve anything as is shown in the next example.

Example 5.2 *We present a simple convex optimization problem which cannot be solved by a co-ordinate descent method since the objective function contains no terms in* y*.*

$$\begin{aligned} \min \quad & x \\ \text{s.t.} \quad & x^2 + y^2 \leqslant 1 \end{aligned}$$

Clearly the optimal function value is -1, found at $(-1, 0)$*, but if we apply co-ordinate descent, minimizing over* x *first, starting from any feasible* (x^0, y^0) *with* $y^0 \neq 0$ *we find that since* y *does not appear in the objective function, we are stuck after one step at* $x^1, y^0)$*, see figure 5.2.*

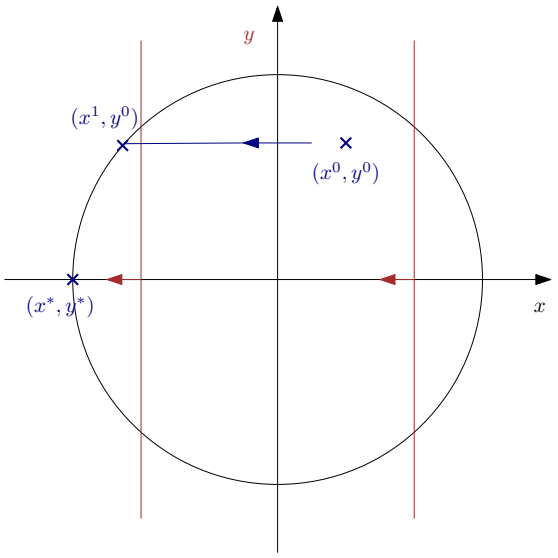

Figure 5.2: *A simple quadratic program which cannot be solved by co-ordinate descent methods*

With these potential stumbling blocks, we need to look closely at the structure of our problem to see if a block co-ordinate descent approach can indeed be of use. Encouragement comes from the work of Fliege and Dekorsy in [17] which provides empirical evidence that the algorithm always comes to a solution quickly, and more importantly, that in zero interference cases a minimizer is found. By closely studying the problem

structure, as well as the co-ordinate descent mechanism, we will back up these empirical observations with mathematical theory. The first stage in this process will be tackled in the next section in which we shall reformulate our problem in such a way as to sidestep the problems highlighted in the examples above.

5.2 Equivalence of Problem Statements

In order to begin considering employing a co-ordinate descent method, we need to deal with the fact that some variables may not appear in the objective function. To do this we will make use of Theorem 3.12 in Chapter 3, namely that at a locally optimal solution, all coupling constraints are active.

We would like now to show that several different problem formulations are equivalent, but first need to explain exactly what we mean for two problems to be equivalent.

Definition 5.3 *Two optimization problems are* ***equivalent*** *if they have the same set of local minimizers, and the same optimal objective function values.*

Consider the following formulation which we will label (P2):

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{p}) \\ \text{s.t} \quad & \mathbf{p} \in \mathbf{C_p} \\ & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \\ & \textstyle\sum_{m \in M} f_{e,t}^m = \Psi_{e,t}(\mathbf{p}) \; e \in E, t \in T \end{aligned}$$

Immediately from the above definition and Theorem (3.12) in chapter 3 we can say (P2) is equivalent to (P1).

In order to apply block co-ordinate descent methods, one would like to have network variables appear in the objective, since in the most common objective function, that of

minimizing total transmit energy, we only have power variables in the objective. In order to do this we rearrange coupling constraint,

$$\sum_{m \in M} f_{e,t}^{m} = \Psi_{e,t}(\mathbf{p}) \; e \in E, t \in T, \tag{5.2}$$

in terms of $p_{e,t}$:

$$p_{e,t} = \frac{1}{a_{e,e}}(2^{\frac{1}{B\tau}\sum_{m \in M} f_{e,t}^{m}} - 1)(\sum_{k \neq e} a_{k,e} p_{k,t} + \sigma^2) := J_{e,t}(\mathbf{f}, \mathbf{p}) \tag{5.3}$$

In the same way as we have grouped our variables as vectors, we will shorthand the vector $(J_{1,1}(\mathbf{f}, \mathbf{p}), \ldots, J_{|E|,t^{\max}}(\mathbf{f}, \mathbf{p})) := \mathbf{J}(\mathbf{f}, \mathbf{p})$.We therefore see that we can write a third equivalent formulation which we shall label (P3):

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f}, \mathbf{p})) \\ \text{s.t} \quad & \mathbf{p} \in \mathbf{C_p} \\ & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \\ & p_{e,t} = J_{e,t}(\mathbf{f}, \mathbf{p}) \end{aligned}$$

Finally, we would like a lemma to show that a fourth formulation, (P4), is equivalent:

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f}, \mathbf{p})) \\ \text{s.t} \quad & \mathbf{p} \in \mathbf{C_p} \\ & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \\ & p_{e,t} \geqslant J_{e,t}(\mathbf{f}, \mathbf{p}) \end{aligned}$$

Unfortunately, in general such a reformulation is not possible, as illustrated by the following simple example:

Example 5.4 *We present a simple two-dimensional constrained optimization problem. For illustrative purposes, we can consider this problem to be in formulation (P1*):*

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & x, y \geqslant 0 \\ & x \leqslant y \end{aligned}$$

Clearly the unique global minimizer here is $(0, 0)$*. There are no local minimizers and so we can reformulate as (P2*) without losing the global minimizer:*

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & x, y \geqslant 0 \\ & x = y \end{aligned}$$

This problem can be rewritten in the form (P3) with* x *in the objective function as follows:*

$$\begin{aligned} \min \quad & x \\ \text{s.t.} \quad & x, y \geqslant 0 \\ & y = x \end{aligned}$$

We run into difficulties when we relax the equality constraint to form (P4):*

$$\begin{aligned} \min \quad & x \\ \text{s.t.} \quad & x, y \geqslant 0 \\ & y \geqslant x \end{aligned}$$

This problem has solutions $(0,t)$ *for all* $t \in \mathbb{R}_+$*, and so is not equivalent to formulations (P1*), (P2*) and (P3*) since it has a different set of minimizers. Note that all minimizers of (P4*) have the same objective function value as the unique global minimizer of (P1*), (P2*) and (P3*).*

Although in general (P4) is not equivalent to the other formulations, we can at least say something about the minimizers of (P4):

Lemma 5.5 *For each local minimizer* $(\mathbf{b}^0, \mathbf{c}^0, \mathbf{d}^0, \mathbf{f}^0, \mathbf{p}^0)$ *of (P4) there exists a local minimizer* $(\mathbf{b}^0, \mathbf{c}^0, \mathbf{d}^0, \mathbf{f}^0, \mathbf{p}^1)$ *of (P4) with:*

- $(\mathbf{b}^0, \mathbf{c}^0, \mathbf{d}^0, \mathbf{f}^0, \mathbf{p}^1)$ *is a local minimizer of (P3).*
- $(\mathbf{b}^0, \mathbf{c}^0, \mathbf{d}^0, \mathbf{f}^0, \mathbf{p}^0)$ *and* $(\mathbf{b}^0, \mathbf{c}^0, \mathbf{d}^0, \mathbf{f}^0, \mathbf{p}^1)$ *have the same objective function value.*

Proof. Consider a local minimizer $\mathbf{x} = (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}, \mathbf{p})$ of (P4). **Case 1**: All coupling constraints are active and we are done. **Case 2**: $\exists\ I \neq \emptyset, I \subseteq E \times T$ such that $p_{e,t} > J_{e,t}(\mathbf{f}, \mathbf{p})\ \forall\ (e,t) \in I$.
Consider $(e,t) \in I$.
Case 2a: $a_{e,l} \neq 0$ for some $l \neq e$, s.t. $\sum_{m \in M} f_{l,t}^m > 0$, and so $p_{e,t}$ appears in $J_{l,t}(\mathbf{f}, \mathbf{p})$ since

$$J_{l,t}(\mathbf{f}, \mathbf{p}) = \frac{1}{a_{l,l}}(2^{\frac{1}{B\tau}\sum_{m \in M} f_{l,t}^m} - 1)(\sum_{k \neq l} a_{k,l} p_{k,t} + \sigma^2)$$

Since $\Phi(\mathbf{b}, \mathbf{f}, \mathbf{p})$ is strictly monotone in $\mathbf{p}$,

$$\widehat{\mathbf{p}} \leqslant \overline{\mathbf{p}} \Rightarrow \Phi(\mathbf{b}, \mathbf{f}, \widehat{\mathbf{p}}) \leqslant \Phi(\mathbf{b}, \mathbf{f}, \overline{\mathbf{p}})$$

Here the inequality is a vector inequality, ie $\widehat{\mathbf{p}} \leqslant \overline{\mathbf{p}} \Leftrightarrow \widehat{p}_{e,t} \leqslant \overline{p}_{e,t} \forall\ e \in E, t \in T$. Further, by strict monotonicity, if the vector inequality is strict in at least one co-ordinate, then

the inequality in the function values is strict also. Similarly,

$$\mathbf{J}(\mathbf{f}, \widehat{\mathbf{p}}) \leqslant \mathbf{J}(\mathbf{f}, \overline{\mathbf{p}}) \Rightarrow \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f}, \widehat{\mathbf{p}})) \leqslant \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f}, \overline{\mathbf{p}})).$$

Now consider $\mathbf{p}_{\text{new}}$ identical to $\mathbf{p}$ but with $p_{e,t}$ replaced by $(p_{e,t} - \delta)$ for some $\delta > 0$. Then,

$$J_{l,t}(\mathbf{f}, \mathbf{p}_{\text{new}}) = \frac{1}{a_{l,l}}(2^{\frac{1}{B\tau}\sum_{m \in M} f^m_{e,t}} - 1)(\sum_{k \neq l k \neq e} a_{k,l} p_{k,t} + a_{e,l}(p_{e,t} - \delta) + \sigma^2) < J_{l,t}(\mathbf{f}, \mathbf{p}).$$

Clearly, decreasing $p_{e,t}$ to $p_{e,t} - \delta$ cannot increase $J_{k,t}$ for any $(k,t) \in E \times T$, and so $\mathbf{J}(\mathbf{f}, \mathbf{p}_{\text{new}}) \leqslant \mathbf{J}(\mathbf{f}, \mathbf{p})$, with strict inequality in at least one co-ordinate. By strict monotonicity, $\Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f}, \mathbf{p}_{\text{new}})) < \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f}, \mathbf{p}))$, and so the descent vector used in the proof of Theorem (2.5) is again a descent direction in this case, and for the same reason as in that proof it is feasible. $\mathbf{x}$ is therefore not a local minimizer, so Case 2a cannot occur.

Case 2b: $p_{e,t}$ does not appear with a non-zero coefficient in any co-ordinate of $\mathbf{J}(\mathbf{f}, \mathbf{p})$, and therefore does not appear in the objective function. We can therefore reduce $p_{e,t} \forall (e,t) \in I$ until $p_{e,t} = J_{e,t}(\mathbf{f}, \mathbf{p}) \forall (e,t) \in I$, without compromising feasibility of any other constraint, to get a new point with the same objective function value, with all coupling constraints active, without altering $\mathbf{b}, \mathbf{c}, \mathbf{d}$ or $\mathbf{f}$. □

With Lemma 5.5 we see that any local minimizer of (P1), (P2) and (P3) is also a local minimizer of (P4), and for any locally optimal solution of (P4), we can find a locally optimal solution of (P1), (P2) and (P3) with the same objective function value without altering the $(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$ variables.

Having ensured that we have formulations of our problem with the various different variables in the objective function to enable us to employ a co-ordinate descent method, we now look closely at the two subproblems we get from fixing either the communication resource variables $\mathbf{p}$, or the network flow variables $(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$ in our optimization problem.

5.3 The Two Subproblems

In this section we present two general classes of optimization problem into which a lot of research has gone. By then comparing these general formulations to our specific optimization problem with one or other set of variables fixed, we see that the two subproblems we will need to solve in our primal co-ordinate descent method fall into these categories and we can therefore expect their solution to be straightforward.

5.3.1 Multi-Commodity Flow Problem

Given a graph $G(V, E)$ where each edge $e \in E$ has an associated capacity $c(e)$. Suppose we have k commodities, $K_1, \ldots, K_k$, defined $K_i = (s_i, d_i, t_i)$, where s_i and d_i are the source and destination of commodity K_i, and t_i is the traffic requirement of commodity K_i. Define the flow of commodity K_i along edge e as $f_i(e)$, and with each edge, associate a cost $a(f(e))$, the cost of sending flow of size f along edge e. A solution to the **multi-commodity flow problem** is a set of flows $f_i(e)$ satisfying the following constraints:

- Non-negativity

$$f_i(e) \geqslant 0 \ \forall i = \{1 \ldots k\}, e \in E$$

- Capacity Constraints

$$\sum_{i=1}^{k} f_i(e) \leqslant c(e)$$

- Flow Conservation

$$\sum_{e \in E_+(v)} f_i(e) = \sum_{e \in E_-(v)} f_i(e) \ \forall v \in V \backslash \{s_i, d_i\}$$

- Demand Satisfaction

$$\sum_{e \in E_+(s_i)} f_i(e) = \sum_{e \in E_-(d_i)} f_i(e)$$

If, in addition, with each edge is associated a cost, which is a function of the amount of flow being carried along that edge, $a(f(e))$, and we find a flow minimizing the total cost,

$$\min \sum_{e \in E} a(\sum_{k=1}^{K} f_i(e))$$

then we have a **minimum cost multi-commodity flow problem**. In a lot of research these costs are assumed to be linear, and so if they are not, this problem is known as a **nonlinear multi-commodity flow problem**.

Looking back at formulation (p4) of our problem, we see that it is the form of a nonlinear minimum cost multi-commodity flow problem, for which much research has been undertaken. See[37], and the papers cited therein for distributed algorithms of convex multicommodity flow problems, particularly when applied to the application of wireless communication networks.

5.3.2 Standard Power Control Problem

In cellular wireless communications, the power control problem is the problem of assigning a vector of transmit powers, $\mathbf{p}$, such that $p_j \geqslant I_j(\mathbf{p})$ for all i, where $I_j(\mathbf{p})$ is a function measuring the interference experienced by user j. The power control problem may also include extra constraints such as non-negativity, and upper bounds on powers. Much research has gone into this problem for CDMA networks, see for example [36] and [48]. In [52], Yates brings together a lot of the previous work, providing conditions on the interference function $\mathbf{I}(\mathbf{p})$ under which the fixed point iteration,

$$\mathbf{p}(t+1) = \mathbf{I}(\mathbf{p}(t))$$

converges to a unique feasible solution minimizing the total power. In this work he uses the concept of a **standard interference function**, and shows that for such an interference

function, the power control algorithm converges to the unique stationary point of the problem, which minimizes the total power.

Definition 5.6 *An interference function* $\mathbf{I}(\mathbf{p})$ *is* ***standard*** *if for all* $\mathbf{p} \geqslant 0$*, the following properties are satisfied.*

- *Positivity:* $\mathbf{I}(\mathbf{p}) > 0$.
- *Monotonicity: If* $\mathbf{p} \geqslant \mathbf{p}'$*, then* $\mathbf{I}(\mathbf{p}) \geqslant \mathbf{I}(\mathbf{p}')$.
- *Scalability: For all* $\alpha > 1, \alpha\mathbf{I}(\mathbf{p}) > \mathbf{I}(\alpha\mathbf{p})$.

For fixed $\mathbf{f}$, the function $\mathbf{J}(\mathbf{f}, \mathbf{p})$ defined in (5.3) trivially satisfies monotonicity, and satisfies positivity on every edge where $c_e \neq 0$. To check scalability, we see that

$$\begin{aligned}\alpha J_{e,t}(\mathbf{f}, \mathbf{p}) &= \frac{1}{a_{e,e}}(2^{\frac{1}{B\tau}\sum_{m\in M} f_{e,t}^m} - 1)(\alpha \sum_{k\neq e} a_{k,e}p_{k,t} + \alpha\sigma^2) \\ &> \frac{1}{a_{e,e}}(2^{\frac{1}{B\tau}\sum_{m\in M} f_{e,t}^m} - 1)(\alpha \sum_{k\neq e} a_{k,e}p_k + \sigma^2) = J_e(c_e, \alpha\mathbf{p})\end{aligned}$$

To get around the non-positivity on edges where $c_e = 0$, the iteration can be slightly altered, by immediately setting the power on all such edges to zero, and then applying the fixed point iteration on all edges with non-zero traffic.

In this manner, we see that, for fixed feasible $(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$ variables, the problem can be solved using fixed point iteration on a standard interference function.

Having shown how the two subproblems arising from fixing either the network flow variables or communication resource variables can be solved using well established existing techniques, we are ready to explain the way in which we will use the different problem formulations detailed in Section 5.2 to design a feasible block co-ordinate descent method for solving our optimization problem.

5.4 The Co-ordinate Descent Algorithm

The idea of Fliege and Dekorsy in [17] is to iteratively solve (P4) for fixed $\mathbf{p}$ variables, then feed the optimal $(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f})$ into (P1) as constants. This process is then repeated until there is no more improvement in objective function value. Since this method consists of taking our overall routing and power control algorithm and decomposed it into a routing subproblem and a power control subproblem, this algorithmic method is referred to in [17] as the **Routing and Power Control Decomposition Algorithm**, or **RPCD algorithm**. Let us look closely at the two subproblems in the RPCD:

For fixed $\widehat{\mathbf{p}} \in \mathbf{C_p}$ we define SP1($\widehat{\mathbf{p}}$), a multi-commodity network flow problem, as:

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f}, \widehat{\mathbf{p}})) \\ \text{s.t} \quad & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \\ & \widehat{p}_{e,t} \geqslant J_{e,t}(\mathbf{f}, \widehat{\mathbf{p}}) \end{aligned}$$

For fixed $(\widehat{\mathbf{b}}, \widehat{\mathbf{c}}, \widehat{\mathbf{d}}, \widehat{\mathbf{f}}) \in \mathbf{C_{b,c,d,f}}$, we define SP2($\widehat{\mathbf{b}}, \widehat{\mathbf{c}}, \widehat{\mathbf{d}}, \widehat{\mathbf{f}}$), a power allocation problem, as:

$$\begin{aligned} & \min \Phi(\widehat{\mathbf{b}}, \widehat{\mathbf{f}}, \mathbf{p}) \\ & \text{s.t } \mathbf{p} \in \mathbf{C_p} \\ \sum_{m \in M} \widehat{f}^m_{e,t} \leqslant \Psi_{e,t}(\mathbf{p}) \; & e \in E, t \in T \end{aligned}$$

We note that both subproblems are convex and a solution to SP2($\widehat{\mathbf{b}}, \widehat{\mathbf{c}}, \widehat{\mathbf{d}}, \widehat{\mathbf{f}}$) will have all coupling constraints active. Denote the optimal function value of SP1($\widehat{\mathbf{p}}$) by $\Phi^*(\widehat{\mathbf{p}})$, and the corresponding values of (b, c, d, f) as $(\mathbf{b}^*, \mathbf{c}^*, \mathbf{d}^*, \mathbf{f}^*)(\widehat{\mathbf{p}})$. Denote the optimal function value of SP2($\widehat{\mathbf{b}}, \widehat{\mathbf{c}}, \widehat{\mathbf{d}}, \widehat{\mathbf{f}}$) by $\Phi^*(\widehat{\mathbf{b}}, \widehat{\mathbf{c}}, \widehat{\mathbf{d}}, \widehat{\mathbf{f}})$, and the corresponding value of $\mathbf{p}$ as $\mathbf{p}^*(\widehat{\mathbf{b}}, \widehat{\mathbf{c}}, \widehat{\mathbf{d}}, \widehat{\mathbf{f}})$.

With this notation we can now outline the procedure of the "Routing and Power Control Decomposition" (RPCD) Algorithm:

- Choose initial $\mathbf{p^0}$, set $i = 0$.
- while $\Phi^*(\mathbf{p}^i) \neq \Phi^*(\mathbf{b}^i, \mathbf{c}, \mathbf{d}^i, \mathbf{f}^i)$
  - $i = i + 1$
  - Solve SP1($\mathbf{p}^{i-1}$)
  - Set $(\mathbf{b}^i, \mathbf{c}^i, \mathbf{d}^i, \mathbf{f}^i) = (\mathbf{b}^*, \mathbf{c}^*, \mathbf{d}^*, \mathbf{f}^*)(p^{i-1})$
  - Solve SP2($\mathbf{b}^i, \mathbf{c}^i, \mathbf{d}^i, \mathbf{f}^i$)
  - Set $\mathbf{p}^i = \mathbf{p}^*(\mathbf{b}^i, \mathbf{c}^i, \mathbf{d}^i, \mathbf{f}^i)$.
- Return $(\mathbf{b}^i, \mathbf{c}^i, \mathbf{d}^i, \mathbf{f}^i, \mathbf{p}^i)$

Fliege and Dekorsy observed in [17] that experimentally this algorithm stops within two cycles of the while loop, We now show why this is the case, based on our understanding of the solution sets of SP1 and SP2.

Lemma 5.7 *After one solution of each of the subproblems SP1 and SP2, the RPCD algorithm terminates at an iteration point at which all coupling constraints are active.*

Proof. We prove this constructively, working through the algorithm from an arbitrary starting point: Let $\mathbf{p}^0$ be our initial choice of $\mathbf{p}$, and $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ the solution to SP1($\mathbf{p}^0$).

Let $\mathbf{p}^1$ be the solution of SP2($\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1$).

We know, from the proof of Thm (2.5) that $\sum_{m \in M} (f_{e,t}^m)^1 = \Psi_{e,t}(\mathbf{p}^1) \ \forall \ e \in E, t \in T$.

Consider now SP1($\mathbf{p}^1$), especially the coupling constraints $p_{e,t}^1 \geqslant J_{e,t}(\mathbf{f}, \mathbf{p}^1)$, which we rearrange back to $\sum_{m \in M} f_{e,t}^m = \Psi_{e,t}(\mathbf{p}^1) \ \forall \ e \in E, t \in T$. Now, we already know we have a feasible point $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ which uses all available capacity on all edges, and in solving SP1($\mathbf{p}^1$) we are seeking another feasible point which uses **less** capacity on some edges than

$\mathbf{f}^1$, but never any **more** capacity on any edges than $\mathbf{f}^1$. Clearly no such point exists and so the feasible set for SP1($\mathbf{p}^1$) is a singleton, namely $\{(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)\}$, so $(\mathbf{b}^2, \mathbf{c}^2, \mathbf{d}^2, \mathbf{f}^2)$, the solution of SP1($\mathbf{p}^1$) must be $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ and therefore $\mathbf{p}^2 = \mathbf{p}^1$ and so on.

By definition, the optimal function value of SP2($\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1$) is $\Phi(\mathbf{b}^1, \mathbf{f}^1, \mathbf{p}^1)$, and the optimal function value of SP1($\mathbf{p}^1$) is $\Phi(\mathbf{b}^2, \mathbf{f}^2, \mathbf{J}(\mathbf{f}^2, \mathbf{p}^1)) = \Phi(\mathbf{b}^1, \mathbf{f}^1, \mathbf{J}(\mathbf{f}^1, \mathbf{p}^1))$, and since $\sum_{m \in M} (f_{e,t}^m)^1 = \Psi_{e,t}(\mathbf{p}^1)\ \forall\ e \in E, t \in T$, we see that $\mathbf{J}(\mathbf{f}^1, \mathbf{p}^1) = \mathbf{p}^1$, and so the optimal function values to the two subproblems are equal for all but SP1($\mathbf{p}^0$).

Note that Lemma (5.7) does not say a great deal about the quality of solutions provided by the RPCD algorithm, other than that they lie on the surface where all coupling constraints are active. At least this is a sensible place to be since we know all solutions lie on this surface also.

We now present an illustrative example of when the RPCD algorithm fails to find a local minimizer.

Example 5.8 *Consider the simple problem of minimizing transmit power while sending a message of size M along two separate edges between two nodes. For simplicity we assume zero interference. The problem (P1) can then be represented as follows:*

$$\begin{aligned} \min \quad & p_1 + p_2 \\ \text{s.t} \quad & f_1, f_2, p_1, p_2 \geqslant 0 \\ & p_1, p_2 \leqslant P^{\max} \\ & f_1 + f_2 = M \\ & f_1 \leqslant B\tau \log_2(1 + a_1 p_1) \\ & f_2 \leqslant B\tau \log_2(1 + a_2 p_2) \end{aligned}$$

For simplicity we assume $M = B = \tau = a_1 = a_2 = 1$. Suppose we run the RPCD

algorithm on this problem from the starting point $\mathbf{p}^0 = (0.1, 1)$. *SP1(*$\mathbf{p}^0$*) is:*

$$\begin{aligned} \min \quad & 2^{f_1} + 2^{f_2} - 2 \\ \text{s.t} \quad & f_1, f_2 \geqslant 0 \\ & f_1 + f_2 = M \\ & f_1 \leqslant \log_2(1.1) \\ & f_2 \leqslant \log_2(2) \end{aligned}$$

with unique minimizer $\mathbf{f}^1 = (0.1375, 0.8625)$, *and objective function value* $\Phi^*(\mathbf{p}^0) = 0.918$. *SP2(*$\mathbf{f}^1$*) is:*

$$\begin{aligned} \min \quad & p_1 + p_2 \\ \text{s.t} \quad & p_1, p_2 \geqslant 0 \\ & p_1, p_2 \leqslant P^{\max} \\ & 0.1375 \leqslant \log_2(1 + p_1) \\ & 0.8625 \leqslant \log_2(1 + p_2) \end{aligned}$$

with unique minimizer $\mathbf{p}^1 = (0.1, 0.818)$, *and objective function value* $\Phi^*(\mathbf{f}^1) = 0.918$. *In this example the algorithm stops after one cycle of the while loop, and to illustrate Lemma*

5.7, notice that SP1($\mathbf{p}^1$) is:

$$\begin{aligned} \min \quad & 2^{f_1} + 2^{f_2} - 2 \\ \text{s.t} \quad & f_1, f_2 \geqslant 0 \\ & f_1 + f_2 = 1 \\ & f_1 \leqslant 0.1375 \\ & f_2 \leqslant 0.8625 \end{aligned}$$

Clearly the only feasible point of this problem is $\mathbf{f}^1 = (0.1375, 0.8625)$. *We see that even in this simple zero interference example, choosing a "bad"* $\mathbf{p}^0$ *leads to a suboptimal solution,* $(\mathbf{f}, \mathbf{p}) = (0.1375, 0.8625, 0.1, 0.818)$ *with objective value 0.918. The global minimizer in this instance is* $(\mathbf{f}, \mathbf{p}) = (0.5, 0.5, \sqrt{2} - 1, \sqrt{2} - 1)$ *with objective function value 0.82.*

Observation 5.4 *With sufficiently large initial* $\mathbf{p}^0$*, the RPCD finds the minimizer to the above simple problem. By sufficiently large, we mean* $\mathbf{p}_1^0 \geqslant \sqrt{2} - 1, \mathbf{p}_2^0 \geqslant \sqrt{2} - 1$.

This observation suggests that in the zero interference case, with sufficiently large initial $\mathbf{p}$ guesses, the RPCD may converge to the global minimizer. We will now formalize this into two results.

Lemma 5.9 *Suppose we wish to solve optimization problem (P1), 5.1, using the RPCD algorithm. Assume the network suffers from no interference, that is*

$$\Psi_{e,t}(\mathbf{p}) = B\tau \log_2(1 + \frac{a_{e,e}}{\sigma^2} p_{e,t}).$$

Assume further that at $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$*, the minimizer of SP1($\mathbf{p}^0$) no coupling constraints are active, that is*

$$\sum_{m \in M} (f_{e,t}^m)^1 < \Psi_{e,t}(\mathbf{p}^0) \forall e \in E, t \in T.$$

Then, if $\mathbf{p}^1$ *is the global minimizer of SP2(*$\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1$*),* $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p}^1)$ *is a global minimizer of (P1).*

Proof. Note that, in the zero interference case, power variables no longer appear in the objective functions of problem formulations P3 and P4 since

$$J_{e,t}(\mathbf{f}, \mathbf{p}) = \frac{\sigma^2}{a_{e,e}}(2^{\frac{1}{B\tau}\sum_{m\in M} f_{e,t}^m} - 1) = J_{e,t}(\mathbf{f}).$$

SP1($\mathbf{p}^0$) is therefore:

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f})) \\ \text{s.t.} \quad & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \\ & p_{e,t}^0 \geqslant J_{e,t}(\mathbf{f}) \end{aligned}$$

This is a convex optimization problem, and so $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ is a global minimizer for SP1($\mathbf{p}^0$). Now, by assumption $p_{e,t}^0 > J_{e,t}(\mathbf{f}^1)\ \forall e \in E, t \in T$, and so $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ is also a global minimizer for the same problem without any coupling constraints:

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f})) \\ \text{s.t.} \quad & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \end{aligned}$$

By extension, for any $\mathbf{p} \in \mathbf{C_p}$, $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p})$ is a global minimizer of:

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f})) \\ \text{s.t.} \quad & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C_{b,c,d,f}} \\ & \mathbf{p} \in \mathbf{C_p} \end{aligned}$$

Adding additional constraints cannot decrease the optimal objective function value for a problem, so if $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p})$ remains feasible for a problem with additional constraints, it will be a global minimizer for that problem as well.

Now, since we assumed $p_{e,t}^0 > J_{e,t}(\mathbf{f}^1) \forall\ e \in E, t \in T$ in the statement of the lemma,

$(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p}^0)$ is clearly feasible for

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f})) \\ \text{s.t.} \quad & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C}_{\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{f}} \\ & \mathbf{p} \in \mathbf{C}_{\mathbf{p}}, \\ & \textstyle\sum_{m \in M} f_{e,t}^m \leqslant B\tau log_2(1 + a_{e,e}p_{e,t}) \ \forall e \in E, t \in T. \end{aligned}$$

$(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p}^0)$ is therefore a global minimizer for this problem, which is exactly formulation (P4) of the original problem. Solving SP2$(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ we obtain $\mathbf{p}^1$ for which all coupling constraints are active. $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p}^1)$ is thus a global minimizer for (P4) for which all constraints are active, and therefore a global minimizer of P1. □

Lemma 5.9 provides us with a simple test to perform on the solution of SP1$(\mathbf{p}^0)$ to tell us if we will reach a global minimizer. The following lemma provides a condition on the choice of $\mathbf{p}^0$ to guarantee convergence to the global minimizer in the zero interference case.

Lemma 5.10 *Assume* $\mathbf{C}_{\mathbf{p}}$ *contains only edge power bounds and not node power bounds and that the networks suffers no interference. Then the RPCD algorithm terminates at the global minimizer of (P1) if we choose* $p_{e,t}^0 = P_e^{\max} \ \forall e \in E, t \in T$.

Proof. Consider $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$, the minimizer of SP1$(\mathbf{p}^0)$:

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f})) \\ \text{s.t.} \quad & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C}_{\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{f}} \\ & P_e^{\max} \geqslant J_{e,t}(\mathbf{f}) \ \forall e \in E, t \in T. \end{aligned} \tag{5.5}$$

Since this problem is convex, $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ is a global minimizer for this problem, and

therefore $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p}^0)$ is a global minimizer for the problem:

$$\begin{aligned} \min \quad & \Phi(\mathbf{b}, \mathbf{f}, \mathbf{J}(\mathbf{f})) \\ \text{s.t.} \quad & (\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{f}) \in \mathbf{C}_{\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{f}} \\ & 0 \leqslant p_{e,t} \leqslant P_e^{\max} \ \forall e \in E, t \in T, \\ & p_{e,t} \geqslant J_{e,t}(\mathbf{f}) \ \forall e \in E, t \in T. \end{aligned} \tag{5.6}$$

since the feasible region for (5.6) is a subset of the feasible region for (5.5). Solving SP2$(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1)$ we attain $\mathbf{p}^1$ for which all coupling constraints are active. $(\mathbf{b}^1, \mathbf{c}^1, \mathbf{d}^1, \mathbf{f}^1, \mathbf{p}^1)$ is thus a global minimizer of (P4) for which all coupling constraints are active and is therefore a global minimizer of (P1). □

Given that we know how to guarantee optimality in the zero interference case, it would be nice to have a stronger result in the case where we suffer interference. As the next section shows, such a result would depend heavily on the choice of starting point, and the way to choose such a starting point is unclear.

5.5 Difficulties Encountered When Faced With Interference

The empirical work of Fliege and Dekorsy in [17] suggests that the RPCD algorithm converges to a minimizer for several classes of interference limited problems. Research is still ongoing into finding a result guaranteeing the convergence of the algorithm in these more general cases. Given our example in the previous section that shows that even without interference, convergence is not guaranteed from certain starting points, one would expect any theorem proving convergence would also involve strict conditions on the starting point of the algorithm. Unfortunately in high interference cases, we can not automatically impose that we set all initial powers to their maximum possible values, as

the next example shows:

Example 5.11 *Consider the same problem as in Example 5.8, but this time we assume extremely high interference, say* $a_{21} = a_{12} = a_{11} = a_{22} = 1$*, and* $\sigma^2 = 0.1$*,* $B = \tau = 1$*. Then, assuming* $p_1^0 = p_2^0 = p$*, we see that our first sub-problem has the constraints:*

$$f_i \leqslant \log_2(1 + \frac{p}{p + 0.1}) \leqslant \log_2(2) = 1$$

Therefore if the message to be transmitted is larger than $2B\tau$ *bits, this initial subproblem is infeasible, whereas if we had chosen* $p_1^0 = p$*, and* $p_2^0 = 0$*, then for large enough* p*, the problem can be feasible for much larger messages.*

The message to take away from this example is that for high interference situations, the choice of initial point is very much dependent on the gain matrix, and to some extent requires prior knowledge of the solution.

Nonetheless, it is not all bad news for the RPCD algorithm. In low interference cases where we are able to set pre-determined start points, we can expect the algorithm to considerably out perform the dual approach in terms of speed, since at every iteration of the dual decomposition algorithm, the solutions to two subproblems of similar difficulty to the subproblems in the RPCD are required. In the RPCD, these subproblems need only be solved once each, whereas in the dual decomposition subgradient approach, each subproblem was solved tens or hundreds of times during the course of the algorithm. In the next section we will present numerical results, backing up the theoretical results shown in this chapter as well as showing the value of the RPCD algorithm, even if only treated as a heuristic with no guarantee of convergence to a local minimizer.

5.6 Numerical Testing of the RPCD Algorithm

We test the RPCD algorithm on the same networks we used to test the benefits of network coding compared to naive routing in section 3.8, that is to say, we use the butterfly network and the 1-3-5-3 backhaul network, each with a single multicast message. In each problem instance, we compare the solution found by the RPCD algorithm with the solution found by a blackbox solver, both in the naive routing case and in the network coding case. We test the algorithm using various different choices of all parameter values and with varying levels of interference. The blackbox solver we use is the NAG sparse nonlinear solver, [35]. In nonconvex problem instances we ran the blackbox solver ten times from random starting points to attempt to get close to the global minimizer. The values we use in our comparative results from our black box solver are always the minimum value obtained from these repeated simulations.

5.6.1 Implementation of the RPCD in our Tests

Due to some of the unanswered questions about the algorithm, certain decisions have had to be made regarding the implementation of the algorithm in our testing. Since both subproblems have been shown to have unique solutions, it was not deemed to be important to solve them using methods one would practically use in wireless settings. We want to get a feel for the performance of the algorithms overall performance and are not too interested in the details of solving at subproblem level. For this reason, in our tests both subproblems are solved with the best and quickest serial solver available to us, that is to say the NAG dense linear solver, e04mf, [34], and the NAG sparse nonlinear solver e04ug, [35]. Since both subproblems are convex, the starting point for the subproblems are unimportant, and so the main choice in implementation was in initializing the power variables.

Since no solution has yet been found to the starting point problem raised in Section

5.5, the approach taken in testing has been to initialize all starting powers to maximum, since this is known to lead to optimal solutions in the zero interference case. In some high interference cases this means the starting point is infeasible and so the algorithm fails, but it was felt that for the purpose of testing it would make no sense to tailor individual starting points for specific problems.

For parameter values used in testing in both the butterfly network and the 1-3-5-3 backhaul network, see Appendix D.

5.6.2 Results

In line with Lemma 5.10, it was found that for all problem instances tested the RPCD algorithm terminated after one solution of each subproblem to a point where all coupling constraints were active. This comes as no surprise to us but backs up our theory with empirical data.

Supporting Lemmas 5.9 and 5.10, we tested both networks in the zero interference case over a range of values of bandwidth, timeslot length, number of available timeslots, message size and background noise, using both Network Coding and Naive Routing, and found that in each case the RPCD algorithm returned a global minimizer.

The main results of interest came when looking at problem instances with interference. In both the butterfly network and the backhaul network, we see that the RPCD algorithm often finds feasible solutions that are close to optimal.

In Figure 5.3 we see that in the butterfly network with full interference (see Appendix D for Gain matrix), the RPCD algorithm returns near optimal solutions when the size of message to be transmitted is sufficiently small, but returns increasingly suboptimal solutions as the message size increases. Eventually, when the message size is too large, our initial $\mathbf{p}^0$ values result in an infeasible subproblem SP1($\mathbf{p}^0$) and so the RPCD fails to return a feasible solution at all. Note that with network coding, the RPCD can handle

larger messages than with naive routing before it is unable to return a feasible solution.

In Figure 5.4 we vary the maximum number of timeslots in the butterfly network with full interference. Again, decreasing the number of available timeslots results in more data having to be transmitted along an edge in any given timeslot and so eventually this leads again to an infeasible SP1 with our initial choice of $\mathbf{p}^0$ if the number of timeslots is limited too harshly. As the maximum number of timeslots is increased, the RPCD algorithm returns solutions close to global minimizers. It is again seen that using network coding as opposed to naive routing allows the RPCD to return solutions when the number of timeslots is more strictly limited.

In Figures 5.5 and 5.6 we investigate the 1-3-5-3 network where we have used realistic parameter values and gain matrices (see Appendix D), considering full interference in Figure 5.5 and intercell interference only in 5.6. In both instances we see that for small messages, the RPCD algorithm returns solutions near solutions found by running the our blackbox solver, the NAG sparse nonlinear solver, multiple times from multiple starting points. As the message size increases, the solutions become suboptimal, and for messages too large, our choice of $\mathbf{p}^0$ results in an infeasible subproblem SP1($\mathbf{p}^0$). Using network coding again results in the RPCD being able to handle larger messages than naive routing.

It is also worth noticing that in most cases, applying the RPCD algorithm to the model in which we use network coding results in solutions, which although suboptimal for this problem, are still better than the optimal solution to the problem in which we use naive routing, and so in a sense, a simple algorithm applied to a complicated model performs better than any complicated algorithm applied to an overly simplistic model.

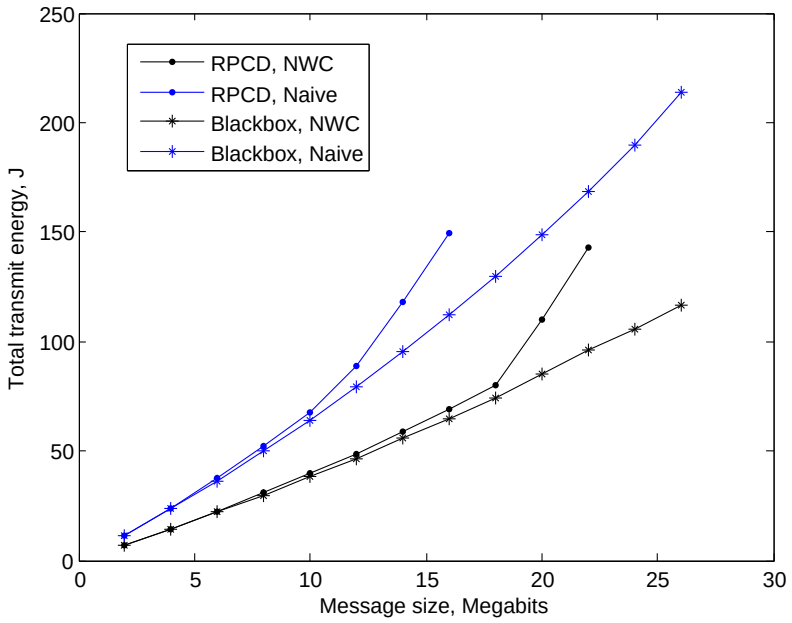

Figure 5.3: *Total multicast transmit energy for varying message sizes, with full interference in the butterfly network over 20 timeslots, comparing RPCD to a blackbox solver.*

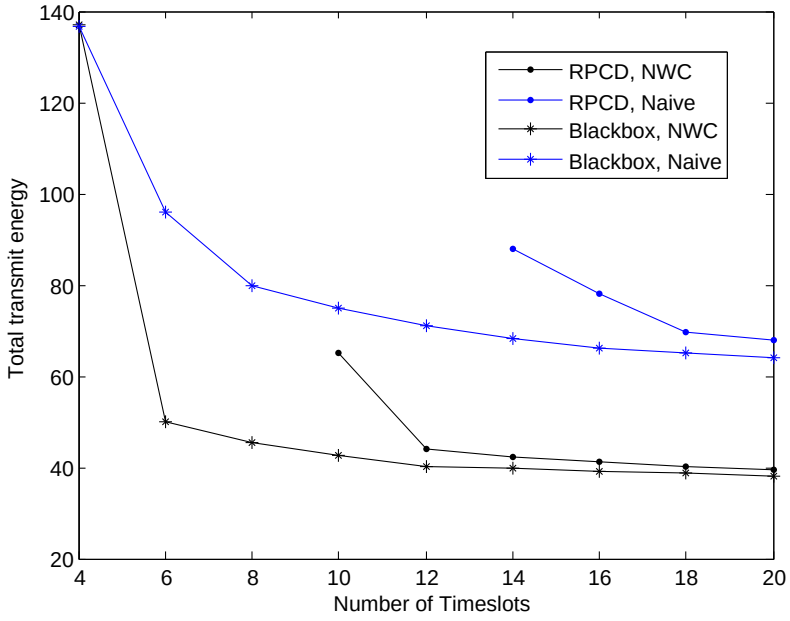

Figure 5.4: *Total multicast transmit energy for varying numbers of timeslots, with full interference in the butterfly comparing RPCD to a blackbox solver.*

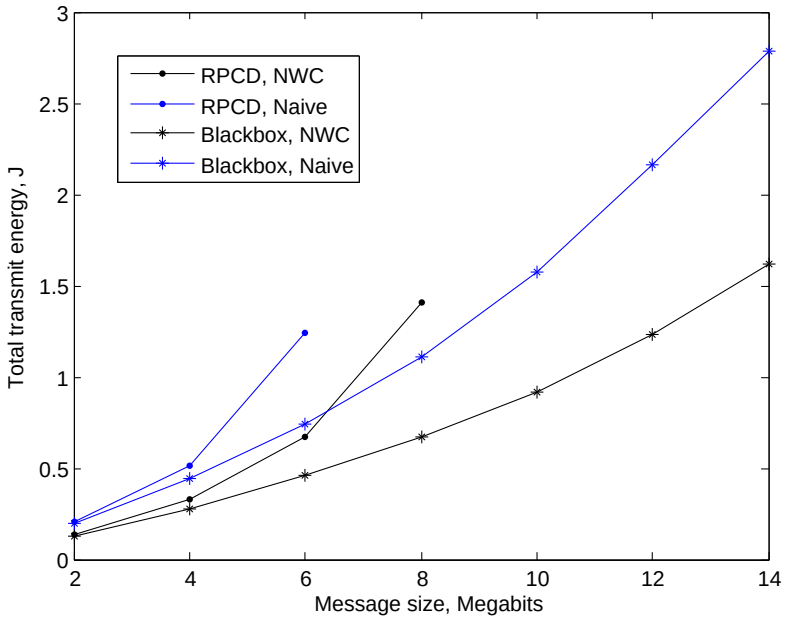

Figure 5.5: *Total multicast transmit energy for varying message sizes, with full interference in the backhaul network over 20 timeslots, comparing RPCD to a blackbox solver.*

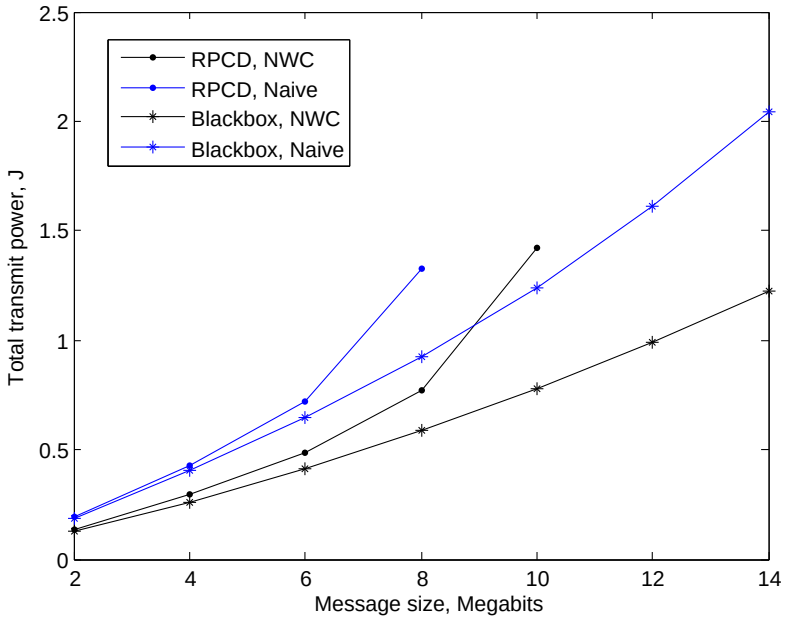

Figure 5.6: *Total multicast transmit energy for varying message sizes, with inter-cell interference only in the backhaul network over 20 timeslots, comparing RPCD to a blackbox solver.*

Chapter 6

Conclusions, Open Questions and Research Directions

In this thesis we have successfully brought together ideas from various areas of mathematics to solve a problem from communication engineering. Specifically, we have developed a mathematical framework and nonlinear program for the problem of transmitting unicast and multicast messages through a timeslotted CDMA multihop wireless network using point to point transmission and allowing multipath routing, simultaneously optimizing over communication resource variables and network flow variables.

The main contributions in the modelling part of the thesis came at the ends of chapters two and three. Having developed a mathematical framework and nonlinear program for the unicast problem in the first part of chapter two, at the end of the chapter we undertook a thorough analysis of the behaviour of a simple problem, providing new insight into the way in which the nonconvexity of the coupling constraints affects the solution set for the problem, finding conditions on the communication channel parameters which result in the problem having a unique solution.

Having introduced the possible ways to handle multicast transmissions at the start of chapter three, the main contribution comes in detailing exactly how, using random linear

network coding and introducing conceptual flow and buffer variables, we can incorporate multicast transmissions into our model whilst avoiding the expensive combinatorial optimization problems associated with the tree packing approach. Numerical experiments run using a NAG nonlinear solver in MATLAB clearly show the benefits of the network coding approach over a naive approach. Further numerical results exhibit the potential cost transporting the overheads associated with the network coding approach, namely having to transport the random encoding coefficients along with the encoded packets. Experiments show that in typical situations, with overheads of up to 5%, the benefit compared to naive routing is still sizeable.

The main contributions of the algorithmic part of the thesis came at the end chapter four and throughout chapter five. Having introduced a fairly standard dual decomposition and subgradient approach to solving our optimization problem, at the end of chapter five we introduce a novel use of Aitken's Δ^2 method and Steffensen's iteration to accelerate the convergence of the subgradient method. This approach showed considerable promise and greatly improved the convergence rate in some cases.

Throughout chapter six we examined the behaviour of an innovative, recently proposed primal block co-ordinate descent method. Before this work little was known about the performance of this algorithm other than empirical observations. We carefully proved the equivalence of the different problem formulations used in the algorithm and explained why the algorithm always stops quickly, as well as characterizing the general set of points the algorithm stops in. In the zero interference case we proved convergence to a global minimum if a certain constraint is inactive at a certain stage of the algorithm. We also proved convergence to a global minimum for a certain starting point. We highlighted possible difficulties in using the algorithm in high interference cases, before finally testing the algorithm extensively and finding that in many cases, although suboptimal, the points to which the algorithm converge are those with close to optimal objective function values.

An interesting theme running through this work is that of balance between simplicity and complexity. In chapter three whilst looking for ways to include multicast in our framework we found that the simplest approach provided highly suboptimal solutions, but that the use of a complicated random linear coding protocol eventually led to a problem formulation which is almost as simple, but leads to much better solutions, and compares favourably to the computationally expensive approach of packing Steiner trees.

In the algorithmic part of the thesis we found that the implementation of a simple primal algorithm can lead very quickly to optimal or near optimal solutions, and applying this algorithm to the problem formulation where we use network coding we see that even when the solutions are suboptimal, they still remain far better than optimal solutions to the naive problem formulation.

As well as the successful outcomes of this research we have found many interesting unanswered questions and directions for future research. The first set of possible research directions involves altering our model to better capture the technology of wireless communication. In this work we have only considered point to point transmissions, whereas a larger number of wireless applications use broadcast technology. Broadcast transmissions could be included in our model by changing our underlying graph into a hypergraph. With a hypergraph, instead of edges we have hyperedges which share common properties but where an edge connects a single source node to a single destination node, a hyperedge connects a single source node with multiple destination nodes. This direction of research is particularly interesting since the benefits of network coding can often be more pronounced in networks using broadcast transmissions, as was shown in section (3.3.2).

A second direction within this area of technological advances is to consider the possibility of using network coding to combine packets from completely separate multicast and unicast transmissions. This process is known as intersession coding, whereas so far we have only considered intrasession coding. As was shown in section (3.3.1), there are potential

throughput gains to be had by employing this approach. Currently, although research has gone into showing that potential gains exist in few very specific network structures, little is known about possible ways of employing intersession coding in a distributed manner for arbitrary networks.

A third possible way to change our model would be to relax the coupling constraints. The constraint we impose means that each communication can theoretically be achieved perfectly with no error. If instead we allowed messages to be transmitted at rates which meant error was introduced, we would gain more flexibility in our model, and as long as we ensured that there was enough information at the end, the message would still be decodable. This ties in nicely with the idea of network coding, since with network coding it would simply be a case of ensuring each destination has enough error free encoded packets.

In terms of research directions with regards to the algorithms we have looked at, there are again several interesting research questions. Firstly, with the dual decomposition algorithm, one could consider the effect of including other constraints than the coupling constraints into the Lagrangian. Another possibility would be to look at different step size controls, or to ensure that the algorithm rejects moves which worsen the dual objective function value.

As far as the primal co-ordinate method is concerned, we have proven that in the zero interference case for certain starting points the algorithm converges to the global optimum. It would be nice to generalise this result to include low interference cases. From an implementation point of view, work needs to go into the exact way in which this algorithm can be performed in a distributed manner.

Finally, a promising new algorithmic approach would be to further exploit the fact that all coupling constraints at a solution are active to rewrite the problem with power constraints removed, as a problem purely in the network flow variables. Preliminary

investigations indicate this to be a worthwhile idea to investigate further.

.

Appendix A
Convex Programming

Definition A.1 *A subset C of $\mathbb{R}^n$ is called* ***convex*** *if*

$$\alpha x + (1-\alpha)y \in C, \forall x, y \in C, \forall \alpha \in [0,1]. \tag{A.1}$$

Definition A.2 *Let C be a convex subset of $\mathbb{R}^n$. A function $f : C \mapsto \mathbb{R}$ is called* ***convex*** *if*

$$f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha)f(y), \forall x, y \in C, \forall \alpha \in [0,1]. \tag{A.2}$$

The function f is called ***concave*** *if $-f$ is convex. The function f is called* ***strictly convex*** *if the above inequality is strict for all $x, y \in C$ with $x \neq y$, and all $\alpha \in (0,1)$.*

If the function in question is once, or twice continuously differentiable, there are equivalent definitions of convexity based on the gradient or hessian of the function in question.

Proposition A.3 *Let C be a convex subset of $\mathbb{R}^n$ and let $f : \mathbb{R}^N \mapsto \mathbb{R}$ be differentiable over $\mathbb{R}^n$.*

- *f is convex over C if and only if*

$$f(z) \geqslant f(x) + (z-x)^T \nabla f(x), \quad \forall x, z \in C.$$

- *f is strictly convex over C if and only if the above inequality is strict whenever $x \neq z$.*

Proof. See [3, Appendix B1]. □

Definition A.4 *Let $A \in \mathbb{R}^{n \times n}$ be symmetric. We say A is* ***positive semidefinite*** *if $x^T A x \geqslant 0 \forall x \in \mathbb{R}^n$. A is* ***positive definite*** *if $x^T A x > 0 \forall x \in \mathbb{R}^n, x \neq 0$.*

With this definition, we can now class a function as convex using information about its Hessian.

Proposition A.5 *Let C be a convex subset of $\mathbb{R}^n$ and let $f : \mathbb{R}^N \mapsto \mathbb{R}$ be differentiable over $\mathbb{R}^n$.*

- *If $\nabla^2 f(x)$ is positive semidefinite for all $x \in C$, then f is convex over C.*
- *Of $\nabla^2 f(x)$ is positive definite for every $x \in C$, then f is strictly convex over C.*
- *If C is open and f is convex over C, then $\nabla^2 f(x)$ is positive semidefinite for all $x \in C$.*

Proof. See [3, Appendix B1] □

It is often useful to appeal to an extended version of Sylvester's criterion to show that a matrix is or is not positive semi-defininite. In order to understand the theorem, we first need a definition.

Definition A.6 *If we have an $n \times n$ matrix A, and let I be a subset of $\{1, \ldots n\}$. If B is formed by removing from A all the rows and columns indexed in I, then the determinant of B is a **principal minor** of A. The leading principle minor Δ_i is formed by removing all but the first i rows and columns of A.*

Proposition A.7 *A matrix A is positive semi-definite if and only if all of its principal minors are non-negative.*

Proof. See [7]. □

Definition A.8 *A mathematical program is called **convex** if the objective and all the constraints are convex.*

For a full treatment of the topic of convexity and convex programming, see [6].

Appendix B
Glossary of Communication Engineering Terms

Bandwidth	The difference between the upper and lower cut-off frequencies of a communication channel.
Broadcast Communication	A transmitted message can be received concurrently by multiple receivers.
Buffer	Memory within a communication device for storing data packets.
CDMA	Code Division Multiple Access, an access scheme based on assigning different sessions different codes.
Communication Device	Any wireless device used for sending or receiving data, typically a mobile phone, laptop, remote sensor, or wireless antenna.
Communication Resource	Any resource in finite supply which is used in wireless message transmission. Typically bandwidth, timeslot length, transmit power.
FDMA	Frequency Division Multiple Access, an access scheme based on assigning different sessions different parts of the available bandwidth.
Frequency Reuse	The process of reusing the same frequency for transmissions physically suitably far apart.
Intercell Interference	Interference originating from other nodes than the source node of the received message.
Intercell Interference	Interference originating from the same node as the received message.
Multicast	The sending of a message from one source to multiple destinations.
Multihop	The sending of a message via intermediate relay devices as opposed to a single hop to a base station.
Multipath Routing	Allowing messages to be split and take a number of different multihop paths through a network.
Noise Spectral Density	N_0, Noise power per unit of bandwidth. We assume this to be white noise and so the total noise across a bandwidth B is BN_0.
Packet	A finite quantity of data, formatted to be ready to be transmitted through the network.

Point to Point Communication	Transmitting a message directly from one device to one other device.
Relay	Any device which receives data packets and then forwards them on.
SINR	Signal to interference and noise ratio, commonly used in calculating channel capacities, this is the ratio between received signal strength and received interference and noise strength.
TDMA	Time Division Multiple Access, an access scheme based on assigning different portions of the available time period.
Unicast	The sending of a message from one source to one destination.

Appendix C
Additional Data for Dual Decomposition Results

In the three tables that follow, each row is the objective function value returned by the black box solver we use to solve the nonconvex subproblem of minimizing $L_{\text{comm}}(\mathbf{p}, \mu)$ with respect to $\mathbf{p}$ for fixed μ within the subgradient algorithm. Along each row, the solver solves an identical problem but starting from different starting points. In different rows we have the same problem but with different values of the dual variable μ. One would therefore hope to find the same results across a row if the solver was consistently finding the optimal solution. As can be seen in the zero interference case, the optimal solution is always found, and as interference increases, the likelihood of finding the optimal solution decreases.

Each row corresponds to a series of experiments using the same dual values but different start points									
-490.4747	-490.4747	-490.4747	-490.4747	-490.4747	-490.4747	-490.4747	-490.4747	-490.4747	-490.4747
-659.0946	-659.0946	-659.0946	-659.0946	-659.0946	-659.0946	-659.0946	-659.0946	-659.0946	-659.0946
-470.3824	-470.3824	-470.3824	-470.3824	-470.3824	-470.3824	-470.3824	-470.3824	-470.3824	-470.3824
-602.8001	-602.8001	-602.8001	-602.8001	-602.8001	-602.8001	-602.8001	-602.8001	-602.8001	-602.8001
-566.098	-566.098	-566.098	-566.098	-566.098	-566.098	-566.098	-566.098	-566.098	-566.098
-488.0502	-488.0502	-488.0502	-488.0502	-488.0502	-488.0502	-488.0502	-488.0502	-488.0502	-488.0502
-629.7947	-629.7947	-629.7947	-629.7947	-629.7947	-629.7947	-629.7947	-629.7947	-629.7947	-629.7947
-572.3668	-572.3668	-572.3668	-572.3668	-572.3668	-572.3668	-572.3668	-572.3668	-572.3668	-572.3668
-360.0783	-360.0783	-360.0783	-360.0783	-360.0782	-360.0783	-360.0783	-360.0783	-360.0783	-360.0783
-593.1785	-593.1785	-593.1785	-593.1785	-593.1785	-593.1785	-593.1785	-593.1785	-593.1785	-593.1785

Table C.1: *Uniqueness of Solution Data for the non-convex subproblem with no Interference*

Each row corresponds to a series of experiments using the same dual values but different start points									
-193.2599	-193.2599	-193.2599	-193.2599	-193.2599	-193.2599	-193.2599	-193.2599	-193.2599	-193.2599
-191.623	-191.623	-191.6229	-191.623	-191.6229	-191.623	-191.623	-191.623	-191.623	-191.623
-202.3041	-202.3041	-202.3041	-202.3041	-202.3041	-202.3041	-202.3041	-202.3041	-202.3041	-202.3041
-204.6733	-204.6733	-204.6733	-204.6733	-204.6733	-204.6733	-204.6733	-204.6733	-204.6733	-204.6733
-202.6117	-202.6117	-202.6117	-202.6117	-202.6117	-202.6117	-202.6117	-202.6117	-202.6117	-202.6117
-153.3747	-153.3747	-153.3747	-153.3747	-153.3747	-153.3747	-153.3747	-153.3747	-153.3747	-153.3747
-165.4392	-165.4392	-165.4392	-165.4392	-165.4392	-165.4392	-165.4392	-165.4392	-165.4392	-165.4392
-168.0536	-168.0536	-168.0536	-168.0536	-168.0536	-168.0536	-168.0536	-168.0536	-168.0536	-168.0536
-164.1279	-164.1279	-164.1279	-164.1279	-164.1279	-164.1279	-164.1271	-164.1279	-164.1279	-164.1279
-178.155	-178.155	-178.155	-178.155	-178.155	-178.155	-178.155	-178.155	-178.155	-178.155

Table C.2: *Uniqueness of Solution Data for the non-convex subproblem with Low Interference*

Each row corresponds to a series of experiments using the same dual values but different start points									
-57.3968	-65.5257	-57.3968	-65.2365	-80.8538	-61.8858	-57.3968	-88.9827	-65.5257	-65.2432
-86.026	-86.026	-86.026	-86.026	-86.026	-72.5611	-86.026	-86.026	-69.6442	-86.026
-85.1495	-78.0546	-99.2955	-76.7123	-97.6945	-92.2391	-97.6945	-97.6945	-84.2854	-113.2212
-84.4341	-79.8381	-89.0396	-68.1495	-84.4341	-77.3509	-89.0396	-89.0396	-84.4341	-84.4341
-51.6736	-64.734	-64.734	-65.2551	-50.1242	-64.734	-62.5713	-64.734	-64.734	-64.734
-80.3189	-82.9626	-82.9626	-80.3189	-77.2283	-83.2035	-83.8048	-61.1818	-77.2283	-82.1119
-46.7507	-61.2422	-61.2422	-61.2422	-59.6753	-61.2422	-51.3525	-61.2422	-49.8839	-61.2422
-104.212	-79.9874	-91.0976	-91.0976	-104.212	-79.9874	-107.5355	-104.212	-105.2035	-94.7612
-84.3537	-83.6146	-68.7946	-82.8062	-83.6146	-84.3537	-71.8385	-71.8385	-84.3537	-84.6199
-95.1853	-69.7354	-95.1853	-69.7354	-64.0871	-90.5792	-69.3301	-69.7354	-55.0245	-69.7354

Table C.3: *Uniqueness of Solution Data for the non-convex subproblem with High Interference*

Dual Algorithm Solution			Black Box Solution	
Edge Powers f_e	Output edge flows p_e	Adjusted edge flows $\hat{f}_e$	Edge Powers	Edge flows
1.047	6.5	3.519	0.908	3.333
1.050	3.5	3.523	0.908	3.333
1.047	0	3.519	0.908	3.334
0.061	0	0.684	0.059	0.666
0.062	0	0.696	0.059	0.666
0.050	6.5	0.586	0.059	0.666
0.050	0	0.583	0.059	0.668
0.070	0	0.769	0.059	0.667
0.059	0	0.667	0.059	0.666
0.046	3.5	0.542	0.059	0.666
0.052	0	0.603	0.059	0.670
0.073	0	0.788	0.059	0.665
0.066	0	0.731	0.059	0.666
0.064	0	0.716	0.059	0.667
0.065	0	0.726	0.059	0.667
0.067	0	0.741	0.059	0.665
0.051	0	0.597	0.059	0.667
0.047	0	0.557	0.059	0.666
0.307	0	2.026	0.300	2.000
0.296	3.5	1.987	0.300	2.000
0.292	6.5	1.971	0.300	2.001
0.297	0	1.989	0.300	2.000
0.307	0	2.024	0.300	1.999
Total power=5.525			Total power=5.105	

Table C.4: *Comparing dual decomposition solution to black box global solver solution in the 1-3-5-1 network with zero interference*

Tables (C.4) and (C.5) present the solutions for two different problem instances using the 1-3-5-1 backhaul network with varying interference levels.

Dual Algorithm Solution			Black Box Solution	
Edge Powers f_e	Output edge flows p_e	Adjusted edge flows $\hat{f}_e$	Edge Powers	Edge flows
0.000	10	0	0	0
0.000	0	0	0	0
96.978	0	9.923	102.3	10
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
0.000	10	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
83.017	0	9.699	102.3	10
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
0.000	0	0	0	0
97.653	0	9.933	102.3	10
0.000	0	0	0	0
0.000	10	0	0	0
0.000	0	0	0	0
Total power=277.6			Total power=306.9	

Table C.5: *Comparing dual decomposition solution to black box global solver solution in the 1-3-5-1 network with zero interference*

Appendix D
Parameter Values used in Results for Chapters Three and Five

Presented here are the parameter values for the graphs in the results sections (3.8) and (5.6.2). For the butterfly network considered in figures (3.16), (3.18), (5.3) and (5.4), the problem parameters used were as follows:
Network Incidence Matrix:

$$\begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{D.1}$$

$B = 100MHz, sigma^2 = 0.1W, \tau = 0.1s, P_v^{\max} = 1000W = P_e^{\max}$
Zero Interference Gain Matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.172 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.172 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{D.2}$$

Full Interference Gain Matrix:

$$\begin{pmatrix} 1 & 1 & 0.5 & 0.5 & 0.108 & 0.172 & 0.108 & 0.108 & 0.108 \\ 1 & 1 & 0.5 & 0.5 & 0.108 & 0.172 & 0.108 & 0.108 & 0.108 \\ 1 & 0.5 & 1 & 1 & 0.172 & 0.293 & 0.128 & 0.172 & 0.128 \\ 0.5 & 1 & 1 & 1 & 0.128 & 0.293 & 0.172 & 0.128 & 0.172 \\ 1 & 0.5 & 1 & 1 & 0.172 & 0.293 & 0.128 & 0.172 & 0.128 \\ 1 & 1 & 1 & 1 & 0.293 & 1.000 & 0.293 & 0.293 & 0.293 \\ 0.5 & 1 & 1 & 1 & 0.128 & 0.293 & 0.172 & 0.128 & 0.172 \\ 0.293 & 0.293 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.293 & 0.293 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{D.3}$$

For the backhaul network considered in figures (3.17), (3.19), (5.5) and (5.6), the problem parameters used were as follows: Network Incidence Matrix:

$$\begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$B = 10MHz, \sigma^2 = 1.26 \times 10^{-9}, \tau = 0.2s, P_v^{\max} = 1000W = P_e^{\max}$

The following three tables present the gain matrices for the backhaul network in the zero interference case (D.1), the case where we have intracell interference only (D.2), and the case where we have intercell interference only (D.3). To obtain the gain matrix for the case where we have full interference, we add the intercell interference matrix to the intracell interference matrix and subtract the zero interference matrix.

$$10^{-7} \times \begin{pmatrix}
0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.148 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.063 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.148 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.148 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.063 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.148 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.148 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.063 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.049 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.148 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.111 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.148 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.049 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.063 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.148
\end{pmatrix}$$

Table D.1: *Gain Matrix for Backhaul Network with zero Interference*

$10^{-7} \times$

0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.443	0.443	0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.443	0.443	0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.148	0.148	0.148	0.148	0.148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.111	0.111	0.111	0.111	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.063	0.063	0.063	0.063	0.063	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.148	0.148	0.148	0.148	0.148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.443	0.443	0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.443	0.443	0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.443	0.443	0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.148	0.148	0.148	0.148	0.148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.063	0.063	0.063	0.063	0.063	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.111	0.111	0.111	0.111	0.111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.148	0.148	0.148	0.148	0.148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.443	0.443	0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.443	0.443	0.443	0.443	0.443	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.148	0	0	0	0	0.148	0	0	0	0	0.148	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.063	0	0	0	0	0.063	0	0	0	0	0.063	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.049	0	0	0	0	0.049	0	0	0	0	0.049	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.443	0	0	0	0	0.443	0	0	0	0	0.443	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.148	0	0	0	0	0.148	0	0	0	0	0.148
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.111	0	0	0	0	0.111	0	0	0	0	0.111	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.443	0	0	0	0	0.443	0	0	0	0	0.443	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.443	0	0	0	0	0.443	0	0	0	0	0.443	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.443	0	0	0	0	0.443	0	0	0	0	0.443	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.111	0	0	0	0	0.111	0	0	0	0	0.111
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.148	0	0	0	0	0.148	0	0	0	0	0.148	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.443	0	0	0	0	0.443	0	0	0	0	0.443	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.049	0	0	0	0	0.049	0	0	0	0	0.049	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.063	0	0	0	0	0.063	0	0	0	0	0.063	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.148	0	0	0	0	0.148	0	0	0	0	0.148

Table D.2: *Gain Matrix for Backhaul Network with Intracell Interference*

$$
10^{-7} \times \begin{pmatrix}
0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.443
\end{pmatrix}
$$

Table D.3: *Gain Matrix for Backhaul Network with Intercell Interference*

List of References

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, July 2000.

[2] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Bellmont, Mass, 1998.

[3] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Bellmont, Mass, 2nd edition, 2003.

[4] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice Hall, Upper Saddle River, New Jersey, 1992.

[5] S. P. Boyd, S. J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 8(1):67–127, 2007.

[6] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[7] E.T. Browne. *Introduction to the Theory of Determinants and Matrices*. University of North Carolina Press, Chapel Hill, NC, 1958.

[8] R. L. Burden, J. D. Faires, and A. C. Reynolds. *Numerical Analysis*. Prindle, Weber and Schmidt, Boston, 2nd edition, 1981.

[9] M. Chiang. Nonconvex optimization for communication networks. In *Advances in Mechanics and Mathematics, Vol. III*. Springer, 2006.

[10] P. A. Chou and Y. Wu. Network coding for the internet and wireless networks. *IEEE Signal Processing Magazine*, 2007.

[11] R. L. Cruz and A. V. Santhanam. Optimal routing, link scheduling and power control in multihop wireless networks. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1, pages 702–711, 2003.

[12] S. Deb, M. Effros, T. Ho, D. Karger, R. Koetter, D. Lun, M. Medard, and N. Ratnakar. Network coding for wireless applications: A brief tutorial. In *IWWAN*, 2005.

[13] J. Edmonds. Edge-disjoint branchings. *Combinatorial Algorithms (Ruskin Ed.)*, pages 91–96, 1973.

[14] T. ElBatt and A. Ephremides. Joint scheduling and power control for wireless ad hoc networks. *IEEE Trans. on Wireless Communications*, 3(1):74–85, 2004.

[15] J. Fliege, A. Dekorsy, and J. Hodgskiss. Accelerating a dual algorithm for the simultaneous routing and power control problem. In *Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2007.

[16] J. Fliege, A. Dekorsy, and C. Leibig. Accelerated power control for cdma systems with beamforming or multiuser detection. In *Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology*, 2004.

[17] J. Fliege and Armin Dekorsy. Optimal distributed routing and power control decomposition for wireless networks. In *Proceedings of the IEEE Global Communications Conference 2007*, 2007.

[18] C. Fragouli, J. Le Boudec, and J. Widmer. Network coding: An instant primer. Technical Report 2005-010, EPFL, 2005.

[19] C. Fragouli and E Soljanin. Decentralized network coding. In *Information Theory Worshop*, 2004.

[20] R. Gallagher. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communication*, 25(1):73–85, 1977.

[21] J. Gilbert. Graphical derivation of aitken's $\triangle^2$-method. `http://www.maths.lancs.ac.uk/~gilbert/m243b/node6.html`, 2010.

[22] S. Glisic and B. Lorenzo. *Advanced Wireless Networks*. Wiley, Chichester, 2009.

[23] A. Hadish and G. F. Manne. Scalable parallel graph coloring algorithms. *Concurrency: Practice and Experience*, 12(12), 2000.

[24] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. *ISIT*, 2003.

[25] T. Ho, M. Medard, J. Shi, M. Effros, and D. Karger. On randomized network coding. In *Allerton*, 2003.

[26] B. Johansson, P. Soldati, and M. Johansson. Mathematical decomposition techniques for distributed cross-layer optimization of data networks. *IEEE Journal on Selected Areas in Communications*, 24(8):1535–1547, 2006.

[27] M. Johansson, L. Xiao, and S. Boyd. Simultaneous routing and power allocation in cdma wireless data networks. In *IEEE International Conference on Communications*, pages 51–55, 2003.

[28] R. Jurdak. *Wireless Ad Hoc and Sensor Networks: A Cross-Layer Design Perspective*. Springer, New York, 2007.

[29] R. Koetter and M.Mdard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5), Oct. 2003.

[30] L. Li and A.J. Goldsmith. Capacity and optimal resource allocation for fading broadcast channels. *IEEE Transactions on Information Theory*, 47(3), 2001.

[31] S. Li, R. Yeung, and N. Cai. Linear network coding. *IEEE transactions on Information Theory*, 2003.

[32] S. Low. Multipath optimization flow control. In *IEEE International Conference on Networks*, pages 39–43, 2000.

[33] D. Luenberger. *Linear and Nonlinear Programming*. Springer, New York, 2nd edition, 1984.

[34] National Algorithm Group. Documentation for the nag e04mf dense lp solver. `http://www.nag.co.uk/numeric/MB/manual_22_1/pdf/E04/e04mf.pdf`, 2010.

[35] National Algorithm Group. Documentation for the nag e04ug sparse nlp solver. `http://www.nag.co.uk/numeric/MB/manual_22_1/pdf/E04/e04ug.pdf`, 2010.

[36] R.W. Nettleton and H. Alavi. Power control for spread-spectrum cellular mobile radio systems. In *IEEE Vehicular Technology Conference*, 1983.

[37] A. Ouorou, P. Mahey, and J. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46(1), 2000.

[38] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.

[39] C. R. Paul. *Fundamentals of Electric Circuit Analysis*. John Wiley and Sons, New York, 2000.

[40] Y. E. Sagduyu and A. Ephremides. Joint scheduling and wireless network coding. In *NetCod 2005, Riva del Garda, Italy*, 2005.

[41] P. Sanders, S. Egner, and L. Tolhuizen. Polynomial time algorithms for network information flow. In *Proc. 15th ACM Symposium on Parallel Algorithms and Architectures*, 2003.

[42] C. E. Shannon. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[43] S. Skiena. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley, Reading, MA, 1990.

[44] T. Stern. A class of decentralized routing algorithms using relaxation. *IEE Transactions on Communication*, 25(10):1092–1102, 1977.

[45] R. Trudeau. *An Introduction to Graph Theory*. Dover, New York, 2nd edition, 1994.

[46] D. N. C. Tse. Optimal power allocation over parallel gaussian broadcast channels. In *IEEE International Symposium on Information Theory*, 1999.

[47] D. N. C. Tse and S. V. Hanly. Multiaccess fading channelspart i: Polymatroid structure, optimal resource allocation and throughput capacities. *IEEE Transactions on Information Theory*, 44(7), 1998.

[48] A.M. Viterbi and A.J. Viterbi. Erland capacity of a power-controlled cdma sytem. *IEEE journal on selected areas in communications*, 11(6), 1993.

[49] Y. Wu, P. A. Chou, and K. Jain. A comparison of network coding and tree packing. In *ISIT*, 2004.

[50] Y. Xi and E. M. Yeh. Optimal distributed power control and routing in wireless networks. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 2506–2510, Seattle, USA, 2006.

[51] L. Xiao, M. Johansson, and S. P. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Tranasctions on Communications*, 52(7):1136–1144, 2004.

[52] R. D. Yates. A framework for uplink power control in cellular radio systems. *IEEE journal on selected areas in communications*, 13(7), 1995.

[53] J. Yuan, Z. Li, W. Lu, and B. Li. A cross-layer optimization framework for multihop multicast in wireless mesh networks. *IEEE Journal on Selected Areas in Communication*, 24(11):2092–2103, 2006.