# Optimal Design, Dimensioning and Tariffing of Telecommunications Networks

Deborah Robyn Brown, B.Sc. (Ma & Comp. Sc.) (Hons) (Adelaide)

*Thesis submitted for the degree of*

*Doctor of Philosophy*

*at*

*The University of Adelaide.*

Department of Applied Mathematics

Faculty of Mathematical and Computer Sciences

July 27, 1997

# Contents

# List of Tables

# List of Figures

# Abstract

This thesis is in two parts, both of which are concerned with the design and optimisation of telecommunication networks.

The first part discusses the physical design of telecommunication networks in which the links of the network form a tree structure. Networks of this form, known as spanning trees, are of importance in many cases. One of these occurs when link installation is expensive (for instance, due to digging trenches) compared to routing costs. In this case the cheapest design will be the minimum spanning tree. Further, minimum spanning trees may also be used as the backbone structure of more complex networks. Several optimisation techniques are used in the investigation of this problem.

The second part of the thesis discusses the optimal dimensioning and tariffing of telecommunication networks. In this part an underlying physical structure is assumed, but the operating company has the freedom to allocate capacity to the various links and to set tariffs for users. The usual objective used in dimensioning such networks is that of minimising network cost subject to grade-of-service constraints. However, many telephone companies are now operating in a private enterprise environment and hence wish instead to maximise their profit. This is the approach we take here. Both analytical and numerical approaches are presented.

# Signed Statement

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

SIGNED: DATE: ...29/7/97..........

# Acknowledgements

I would like to thank my supervisors Dr Peter Taylor and Dr Nigel Bean for their boundless enthusiasm, intelligence, encouragement and friendship. I would also like to thank Dr Diana Lucic for being fascinated enough by the idea of her being a secretary to have a very helpful chat; and (nearly Dr) David Standingford for being willing to have endless chats. Thanks also to Dr Franz Salzborn and Dr Liz Cousins for their supervision during the first part of my candidature, and to my family and friends for their support throughout.
Many thanks to the Teletraffic Research Centre who partly funded this research.

# Chapter 1

# Introduction

This thesis is in two parts, both of which are concerned with the design and optimisation of telecommunication networks.

In the design of telecommunications networks one usually has a set of nodes with associated traffic demands, and it is desired to connect the nodes by links such that certain specifications are met and the network cost is minimised. The specifications usually include a requirement that the capacity of the links included in the design must be sufficient to carry the traffic demands between every pair of nodes. A cost per unit of capacity is given for each link, and the cost of the network is then a function of this unit cost and the traffic carried by each link. We will assume that the cost of a link is proportional to its capacity.

In Part I of the thesis we will assume that one unit of traffic requires one unit of capacity, that is, traffic is a deterministic flow which can be completely accommodated if there is enough capacity. In Part II traffic will be assumed to be stochastic and hence subject to link or path blocking. When this happens capacity may be a non-linear function of traffic.

One specific type of network design problem, in which the links of the network form a tree structure, known as a spanning tree, will be discussed in Part I of the thesis. A spanning tree is a network in which there is a single path from any node to any other. The case in which the traffic demands and unit costs are arbitrary (not all zero and not all equal) and in which the minimum cost spanning tree is desired, is known as the **Optimum Communication Spanning Tree** (OCST) problem.

Networks of this form are of importance in many cases. One of these occurs when link installation is expensive (for instance, due to digging trenches) compared to routing costs. In this case the cheapest design will be the minimum spanning tree. Further, minimum spanning trees may also be used as the backbone structure of more complex networks or as an initial design for a network. Finding a good backbone or initial network design is critical.

We will use several optimisation techniques in the investigation of this problem. They include two random search techniques, namely simulated annealing and genetic algorithms. Several heuristics will also be discussed. Suggestions for the network design and solution method to use will be made.

Part II of the thesis discusses the optimal dimensioning and tariffing of telecommunication networks. An underlying physical structure is assumed, which may be designed using the techniques in Part I of the thesis. The operating company has the freedom to allocate capacity to the various links and to set tariffs for users. The usual objective used in dimensioning such networks is that of minimising network cost subject to grade-of-service constraints. However, many telephone companies are now operating in a private enterprise environment and hence wish instead to maximise their profit. This is the approach we will take.

In our discussion of optimal tariffing we incorporate the concept of a traffic elasticity function. This function acknowledges the fact that the traffic offered to the network is a decreasing function of the tariff charged to users. We also incorporate the notion of logical links. A logical link uses capacity on a set of physical links but has a separate link blocking probability and capacity. We have knowledge of a telecommunications company that is currently investigating incorporating logical links into their network and hence this work is timely.

A numerical approach is presented to this problem which uses a standard NAG library routine. An analytical approach is also given which supports the numerical results.

We will discuss the form the thesis will take.

In Chapter 2 we introduce the minimum cost spanning tree problem. Given a set of nodes and a list of offered traffic between OD pairs a communication tree is a spanning tree such that each link has sufficient capacity to carry all traffic assigned to it. The cost of the tree is the sum of the link costs. Each link cost is taken to be the product of the length of the link and its capacity. The problem of finding the minimum cost spanning tree is NP-complete. We discuss the representation of solutions to this problem and the application of a heuristic.

Chapter 3 discusses the first random search technique, simulated annealing. Simulated annealing algorithms move from one solution to another, with moves that cause a decrease in cost always being accepted and moves which cause a cost increase being accepted with a certain probability, related to a 'temperature schedule'. We then discuss the application of this algorithm to the minimum cost communication spanning tree problem and discuss similar applications.

The second random search technique, namely a genetic algorithm, will be discussed in Chapter 4. Genetic algorithms are based on the notion of biological evolution and work with a set of solutions which are combined in some way to form a new set. We will again discuss the application of the method to the problem and discuss similar applications.

The results of applying the solution methods are presented in Chapter 5. It is noted that in a more restrictive formulation of the problem for which the cost per unit of traffic for each link is uniform, the optimal solution to the problem is in general a star network. Results are shown for the unrestricted problem for both Euclidean and non-Euclidean traffic.

Chapter 6 starts the second part of the thesis. In this chapter we present an approach to optimal dimensioning and tariffing of communication networks. We choose link capacities, tariffs and the routing strategy in order to maximise the profit for the company operating the network. The tariffs and grade of service are subject to regulatory constraints. It is assumed that we have an existing network structure consisting of a set of nodes and physical links. By cross-connecting traffic through nodes at a high bandwidth rather than multiplexing and de-multiplexing it, a logical link (consisting of capacity on several physical links) is created. However, it may be better for an OD pair to take advantage of existing physical links rather than to initiate its own logical link. In this chapter we will also look at an optimisation procedure for the problem and give a numerical example.

Several results are presented in Chapter 7. The inverse of the Erlang fixed point approximation is used throughout the analytical and numerical work, and proofs both with and without the use of an approximation to the derivative are shown. The main results include a simple formula for the optimal tariff and the fact that only one of the possible routes (logical or physical) for each OD pair will be used in the optimal solution.

The problem formulation is extended in Chapter 8 to allow a larger number of path choices. Traffic may now use a combination of logical and physical links. We again find a simple formula for the optimal tariff. We also find a formula for the splitting probabilities at which there is a stationary point. However, it is not possible to determine whether these splitting probabilities are feasible or whether the objective function is maximised or minimised at this point. Thus we use a numerical example to conjecture that the splitting probabilities will be zero/one, with only one path for each stream being used. In fact, in the chosen network design which incorporates logical links, the routing will be shortest path.

Chapter 9 discusses the main results of the thesis and the implications for the network designs. General strategies for designing and optimising telecommunications networks that have arisen from the work in this thesis are given.

# Part I

# Optimal Communication Spanning Tree Design

In this part of the thesis, we introduce the minimum cost spanning tree problem, in which there is a single path from any node to any other. Given a set of nodes and a list of offered traffic between OD pairs, a communication tree is a spanning tree such that each link has sufficient capacity to carry all traffic assigned to it. The cost of the tree is the sum of the link costs. Each link cost is taken to be the product of the length of the link and its capacity. The problem of finding the minimum cost spanning tree is NP-complete. We discuss the representation of solutions to this problem and the application of a heuristic. Two random search techniques, simulated annealing and genetic algorithms, are discussed along with their application to this problem. The results of applying the algorithms are presented and their performance discussed.

# Chapter 2

# Problem formulation

## 2.1 Introduction

In this part of the thesis we will consider one specific type of network design problem, namely the **spanning tree problem**. A spanning tree is a network in which there is a single path from any node to any other. The case in which the traffic demands and unit costs are arbitrary (not all zero and not all equal) and in which the minimum cost spanning tree is desired, is known as the **Optimum Communication Spanning Tree** (OCST) problem.

Hu [36] considered two sub-problems of the OCST problem in detail. The first sub-problem consisted of the case in which the costs are all equal to one and the demands are arbitrary, known as the optimum requirement spanning tree. The second case occurs when the costs are arbitrary while the demands are all equal to one, known as the optimum distance spanning tree. Hu gave a polynomial-time algorithm for the first case, which we will discuss later, and stated that the optimal tree in the second case is a star network.

Many variations of the OCST problem have been considered. One of these is the Capacitated Optimum Communication Spanning Tree (COCST) problem in which an added specification is that each link capacity has an upper bound. Zhang and

Indulska [65], amongst others, have considered this problem and Papadimitriou [55] has shown it to be NP-complete.

Another specification that can be added is that certain nodes can be connected by at most a certain number of links, known as a node degree constraint ([25],[45]). A variation of this, given by Agarwal *et al.* [4], is that certain nodes are required to be outer nodes, thus having degree one. They also considered a requirement that certain pairs of nodes must be directly connected.

Myung *et al.* [51] considered a variation in which nodes are partitioned into mutually exclusive and exhaustive node sets and exactly one node from each set must be included in the tree.

A good summary on trees is the book by Magnanti and Wolsey [49] which discussed some of the problems above. In particular, the authors discussed the minimum spanning tree problem (link weights, no traffic demands), the rooted subtree problem (a certain node is chosen as a root node), the Steiner tree problem (nodes other than terminal nodes may be incorporated in the network), the K-median problem (limits on the number of node disjoint subtrees), and the C-capacitated problem (subtrees can contain at most C nodes). They did not, however, discuss the OCST problem which Johnson *et al.* [40] have shown to be NP-hard.

Of the work done on the OCST problem there have been three classes of solution methods used : exact branch and bound techniques ([5], [20]), a genetic algorithm ([54]) and a variety of heuristic approaches ([5], [20], [54], [57]). A simulated annealing algorithm has been used for a similar problem ([21]). Ahuja *et al.* [5] stated that the branch and bound algorithm is suitable only for small problem sizes as solution times grow exponentially with problem size. We shall discuss these approaches in further detail in this chapter and in Chapters 3 and 4.

As the number of spanning trees in the solution space for an $n$ node problem is $n^{n-2}$, enumerating all solutions to find the optimum is feasible only for small problems. Lower bounding techniques have been used to give an idea of the quality of the solutions obtained by these methods, but in general they do not give very good bounds and are computationally expensive ([13], [21], [5]). Hence, in the majority of the literature, the algorithms have been tested against each other.

Algorithms have been tested with both Euclidean and non-Euclidean link costs. Ahuja *et al.* [5] stated that the Euclidean problems are more difficult to solve as there is a large number of near-optimal solutions in Euclidean problems which makes them inherently difficult. This is a good reason for using simulated annealing and genetic algorithms for the OCST problem as they are designed to avoid getting trapped in local minima.

In the next section of this chapter, we formulate the OCST problem and discuss representations for solutions to the problem. We also discuss various heuristic techniques which may be applied to the problem. In Chapter 3, a general overview of simulated annealing is given, along with a discussion of its application to the OCST problem. In Chapter 4, a general overview of genetic algorithms is given, along with a discussion of its application to the OCST problem. The results from the implementation of these algorithms for the OCST problem are shown in Chapter 5, and they are compared with a heuristic detailed in Brown *et al.* [13]. The genetic algorithm and the simulated annealing algorithm were also reported in [13]. Independently of [13], Palmer and Kershenbaum [54] developed a genetic algorithm for this problem using, amongst other things, a different representation. The differences between the two algorithm implementations will be discussed further in Chapter 4.

## 2.2 OCST problem formulation

Consider a network in which there is a group of cities or nodes $n \in \mathcal{N}$ joined by a set of undirected links $j \in \mathcal{J}$. A spanning tree $T$ is a connected network which consists of the nodes $n \in \mathcal{N}$ and $n-1$ links $j \in \mathcal{J}$. We shall define this set of links that make up tree $T$ to be $\mathcal{J}_T$. Define $s \in S$ to be the set of streams or origin-destination pairs which can be represented by the unordered node pair $(o_s, d_s)$ where $o_s, d_s \in \mathcal{N}$.

The arrival rate of calls to stream $s$ is defined to be $\nu_s$ where all calls must be carried. As the network has a tree structure, all calls for stream $s \in S$ have a unique path, denoted $p_s$, through the tree $T$.

When a link $j \in \mathcal{J}_T$ is removed from the spanning tree two sub-trees are formed. The set of nodes belonging to one of these sub-trees is defined as $\mathcal{N}_j$ and the set of nodes belonging to the other is defined as $\overline{\mathcal{N}}_j$. The set of links, $k \in \mathcal{J}$, with one end in $\mathcal{N}_j$ and the other in $\overline{\mathcal{N}}_j$ is known as a fundamental cutset of tree $T$ and is denoted by $(\mathcal{N}_j, \overline{\mathcal{N}}_j)$. For every spanning tree $T$ there are $(n-1)$ fundamental cutsets labelled $(\mathcal{N}_j, \overline{\mathcal{N}}_j)$ for $j \in \mathcal{J}_T$.

Define $(o_s, d_s) \in S_j$ to be the set of streams such that when link $j$ is removed from tree $T$, $o_s \in \mathcal{N}_j$ and $d_s \in \overline{\mathcal{N}}_j$. Thus the amount of traffic that uses link $j$ in tree $T$ is equal to the traffic across this cutset and is denoted by $\rho_j$ where

$$\rho_j = \sum_{s \in S_j} \nu_s. \tag{2.2.1}$$

Defining the cost per unit of traffic on link $j \in \mathcal{J}$ to be $m_j$ gives the total cost of the spanning tree $T$ to be

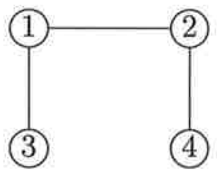
$$C_T = \sum_{j \in \mathcal{J}_T} \rho_j m_j. \tag{2.2.2}$$

The optimal communication spanning tree (OCST) problem is the problem of finding the minimum cost spanning tree $T$. There are often numerous closely matched solutions, which can make solving the problem more difficult.

### 2.2.1 Tree representations

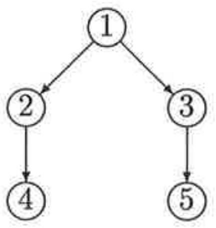
The most obvious choice for encoding solutions to the tree problem is a **bit string** where each position in the encoding represents a particular link. A one in a position implies the link is included in the solution, a zero that it is not. For example see Figure 2.2.1.

The **predecessor encoding** is a simple and short way of encoding a tree. To find the predecessor representation of a tree it is first necessary to choose a root node. For each node $i$ there is a unique path from that node to the root node. The



| Link | (1,2) | (1,3) | (1,4) | (2,3) | (2,4) | (3,4) |
|---|---|---|---|---|---|---|
| Link no. | 1 | 2 | 3 | 4 | 5 | 6 |
| Binary | 1 | 1 | 0 | 0 | 1 | 0 |

Figure 2.2.1: Binary encoding for a tree



| i | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| pred[i] | 0 | 1 | 1 | 2 | 3 |

Figure 2.2.2: Predecessor encoding for a tree

node adjacent to node $i$ on this path is called the predecessor of node $i$. The list of predecessors for all nodes $1, \dots, N$ forms the predecessor representation. The root node is given a predecessor of 0. Figure 2.2.2 gives an example of this.

However, this is not a unique representation for a tree as the encoding will differ depending on the choice of root node. There are, in fact, $N$ different encodings for each tree, where each node $1, \dots, N$ can be the root node. An example of this can be seen in Figure 2.2.3. By always choosing node 1 to be the root node we have a unique encoding for trees.

The **Prüfer encoding** for a tree $T$ with $N$ nodes is a list of $N-2$ numbers stored in $P(T)$. To obtain the Prüfer encoding from a tree choose the lowest numbered leaf node $i$ in the tree. Let the node to which node $i$ is connected in the tree be node $j$. Add $j$ to the right end of $P(T)$, and remove node $i$ and link $(i, j)$ from the tree. This is repeated until only two nodes remain in the tree. An example of this can be seen in Figure 2.2.4.

To obtain the tree from the Prüfer encoding, firstly let $\overline{P}(T)$ be the list of nodes

| | i | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| (a) | pred[i] | 0 | 1 | 1 | 2 |
| (b) | pred[i] | 2 | 0 | 1 | 2 |
| (c) | pred[i] | 3 | 1 | 0 | 2 |
| (d) | pred[i] | 2 | 4 | 1 | 0 |

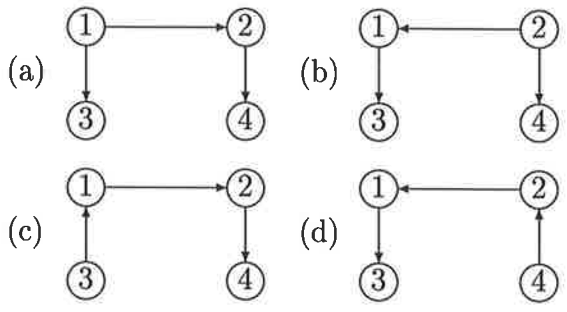


Figure 2.2.3: Root choice with predecessor representation

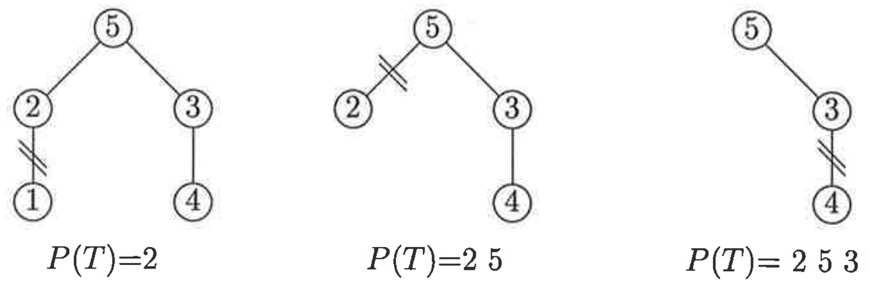


Figure 2.2.4: Finding the Prüfer encoding from a tree

not in $P(T)$. Let $i$ be the lowest numbered node in $\overline{P}(T)$ and $j$ be the leftmost digit of $P(T)$. Add link $(i, j)$ to the tree. Remove $i$ from $\overline{P}(T)$ and $j$ from $P(T)$. If $j$ no longer occurs in $P(T)$ add it to $\overline{P}(T)$. This is repeated until $P(T)$ is empty and $\overline{P}(T)$ contains two nodes, $i$ and $j$. Add $(i, j)$ to the tree which is now complete. An example of this can be seen in Figure 2.2.5.

Another representation, used by Palmer and Kershenbaum [54], uses a **node and link weight encoding**. These weightings are used to create a new cost matrix. An optimisation algorithm is applied to this cost matrix (ignoring the traffics) to find the minimum spanning tree. To find the evaluation of the solution represented by this encoding, the minimum spanning tree is then evaluated using the original cost matrix and the traffic matrix.

This representation encourages certain nodes to be interior and others to be leaf nodes. The encoding for each solution contains a bias $b_i$ for every node and $b_{ij}$ for

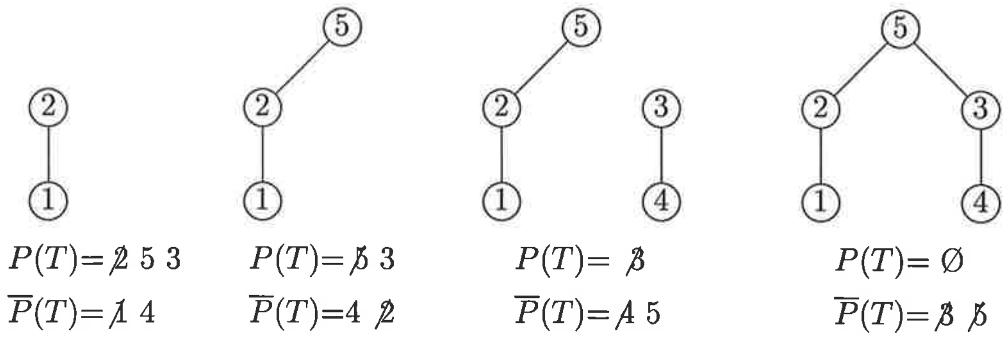

$P(T)=\cancel{2}\ 5\ 3$
$\overline{P}(T)=\cancel{1}\ 4$
$P(T)=\cancel{5}\ 3$
$\overline{P}(T)=4\ \cancel{2}$
$P(T)=\ \cancel{3}$
$\overline{P}(T)=\cancel{4}\ 5$
$P(T)=\emptyset$
$\overline{P}(T)=\cancel{3}\ \cancel{5}$

Figure 2.2.5: Obtaining the tree from a Prüfer encoding

every link in the network. The cost matrix then becomes

$$C'_{ij} = C_{ij} + P_1 b_{ij} C_{\max} + P_2(b_i + b_j) C_{\max} \tag{2.2.3}$$

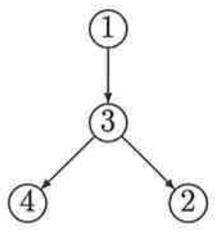
where $P_1$ and $P_2$ are multiplier parameters for the link and node biases respectively and $C_{\max}$ is the maximum link cost. The tree that the encoding represents is found by applying Prim's algorithm [27] to find a minimum spanning tree using the biased cost matrix. This tree is then evaluated using the original cost matrix to find the cost of the communication spanning tree.

## 2.2.2 Tree and star generation

To seed a solution method for the OCST problem, it is often necessary to be able to generate random feasible trees. It is also often valuable to be able to generate the best star as its evaluation is often within 10% of the optimal tree's evaluation.

The best star is found by comparing the $n$ star networks. Each node is taken in turn as the hub node and the cost of this star, where all other nodes are connected to the hub node, is calculated.

Randomly generated trees are formed by keeping a list of nodes which are in the tree and a list of the rest which are not. At the start node 1 is the only node in the 'in tree' list. A node $i$ is then randomly selected from the 'in tree' list and a node $j$ from the 'out of tree' list. Node $j$ is moved to the 'in tree' list and link $(i, j)$ is



| In | Out of | Links which | Pred array | | | |
|---|---|---|---|---|---|---|
| tree | tree | can be added | 1 | 2 | 3 | 4 |
| 1 | 2 3 4 | (1,2)**(1,3)**(1,4) | 0 | | | |
| 1 3 | 2 4 | (1,2)(1,4)(3,2)**(3,4)** | 0 | | 1 | |
| 1 3 4 | 2 | (1,2)**(3,2)**(4,2) | 0 | | 1 | 3 |
| 1 3 4 2 | - | | 0 | 3 | 1 | 3 |

Figure 2.2.6: Creating a random tree

added to the tree. This is repeated until the 'out of tree' list is empty and the tree is complete. This guarantees a feasible solution. An example of this is shown in Figure 2.2.6. This method also gives a directed tree and thus no repairing is needed to get a tree or the predecessor array. Starting with node 1 in the 'in tree' list and adding from there guarantees a feasible directed tree, rooted at node one, but does not rule out any possible trees.

## 2.3 Heuristic algorithms for the OCST problem

The heuristics that have been used for this problem have been discussed in Palmer and Kershenbaum [54], Brown *et al.* [13], Ahuja *et al.* [5] and Dionne and Florian [20].

Palmer and Kershenbaum's heuristic works on the principle "that one or two stars are good". The heuristic performs three searches. The first search finds the best star tree, the second, the best two-hub tree. The third search starts with a minimum spanning tree (as opposed to a minimum communication spanning tree) and tries to reduce the number of interior nodes by redirecting links from leaf nodes until there are no further improvements possible. The heuristic then chooses the best of these three solutions as its final solution.

The heuristic of Ahuja *et al.* chooses a node and builds the tree by successively adding the link which causes the smallest increase in the cost of the tree as it is so far. Every link of the tree is then examined, and if interchanging it with a link not in the tree reduces the overall cost, the link is replaced. This continues until there is no possible interchange which results in a cost reduction. Ahuja *et al.* report that this heuristic is reasonably good computationally and has good accuracy.

This is similar to a heuristic by Camerini *et al.* [14]. However, the heuristic of Camerini *et al.* requires a specialised form of demand matrix and doesn't perform as well as that of Ahuja *et al.* Dionne and Florian's heuristic uses a variation of the branch and bound algorithm which is computationally very expensive.

Salzborn's heuristic ([57], [13]) is not heavily star based as any star can be completely reduced and rebuilt to improve the cost of the tree. It is similar to the heuristic of Ahuja *et al.* in that both are two phase algorithms. The heuristic of Ahuja *et al.* has a tree-building and a tree-improvement phase; Salzborn's has a tree-reduction and a tree-improvement phase. It would be expected that these heuristics would have similar performance and produce similar results, and that both would outperform the other heuristics.

### 2.3.1 Our heuristic

The heuristic we use by itself and in conjunction with the genetic algorithm and the simulated annealing algorithm is one designed by Salzborn ([57],[13]) which uses as its basis a polynomial algorithm and the concept of two-hub trees. The starting tree for the heuristic is the best two-hub tree. The output is a single solution.

A two-hub tree is a spanning tree in which the hubs $h_1$ and $h_2$ are connected by a link and all the nodes are connected to either $h_1$ or $h_2$.

For given hubs, $h_1$ and $h_2$, the nodes must be divided into two sets where $h_1 \in X_1$ and $h_2 \in X_2$ with the nodes $X_i - \{h_i\}$ connected to node $h_i$, $i = 1, 2$. This division of the nodes is done by finding the cutset $(X_1, X_2)$ with minimum value. By Ford and Fulkerson's Theorem [24] this is known to be equivalent to finding the maximum

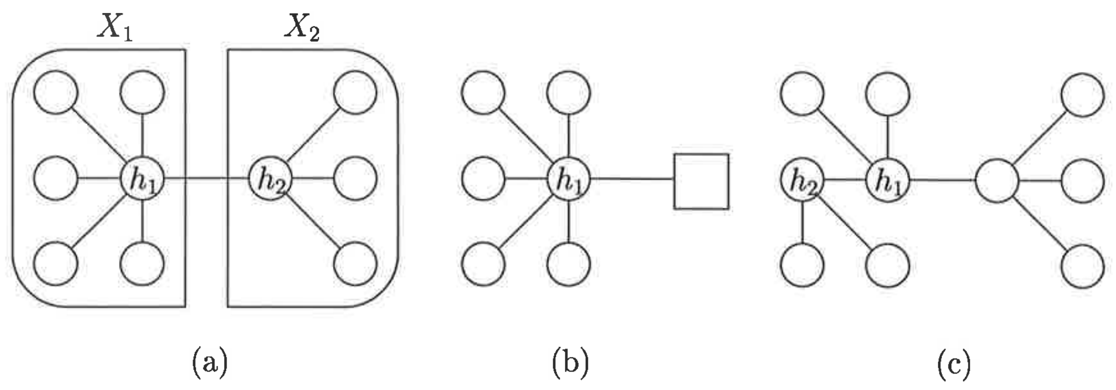


Figure 2.3.7: Heuristic

flow from $h_1$ to $h_2$, for which there are polynomial algorithms of complexity $O(n^3)$.

To find the minimum cost two-hub network this process is simply repeated for each possible pair of hubs, of which there are $\frac{n(n-1)}{2}$.

The heuristic then proceeds by selecting a node $h$ with more than one neighbour. Every neighbour, $g_i$, of $h$, and the nodes connected to $g_i$ when node $h$ is removed from the tree, is then considered to be a pseudo-node. A pseudo-node is a set of one or more nodes which is considered to be a single node with the properties of the set of nodes it contains. The traffic between two pseudo-nodes for instance is the sum of all the traffics between the two sets of nodes the pseudo-nodes represent.

If we selected node $h_1$ from (a) in Figure 2.3.7 for this process, the new network would be (b) with the appropriate properties.

The algorithm repeats the process of choosing every possible pair of nodes in this new network as hub nodes in an attempt to find a cheaper two-hub network. If one exists, the links of the network are replaced with those of the cheaper one. For instance, (b) in Figure 2.3.7 may now become (c). This process continues until none of the stars can be split with a further reduction in cost.

This heuristic can also be applied such that once this process has stopped, the reverse procedure is applied. That is, a two-hub network is combined to form a star if this results in a lower cost network. Again, this continues until no further reduction in cost can be made.

### 2.3.2 Greedy algorithm

A possible addition to the end of a heuristic is a greedy algorithm. The starting tree for the greedy algorithm is the best tree found by the heuristic. The greedy algorithm attempts to improve upon this solution by only accepting moves which cause an improvement in cost. The predecessor representation is used and moves from one solution to another are performed by choosing one link to remove from the tree and replacing it by another link which retains the tree structure.

## 2.4 Other solution methods

For problems in which the search space is discrete and the number of elements is factorially large it is not possible to explore the search space exhaustively or use many of the conventional optimisation techniques.

Simulated annealing and genetic algorithms are random search techniques which differ from other methods in that they allow moves which cause an increase in cost. This allows them to extricate themselves from (non-global) local minima. They are also not heavily dependent on the initial solution and are relatively simple to use.

Genetic algorithms are based on the ideas of natural evolution and work with a set of solutions which are combined in some way. Simulated annealing uses the concepts of statistical mechanics and moves from solution to solution accepting all cost decreasing moves and some cost increasing moves.

We will discuss each of these techniques in more detail in Chapters 3 and 4.

# Chapter 3

# Simulated annealing

## 3.1 Simulated annealing background

Simulated annealing is a random search technique based on the principles of statistical mechanics and thermodynamics. These were first linked to combinatorial optimisation problems by Kirkpatrick *et al.* [46] in 1983. The explanation of statistical mechanics which follows is mainly taken from [17] and [56].

Statistical mechanics is the study of the behaviour of very large systems of interacting components such as atoms in liquids. At high temperatures the molecules of a liquid move freely, but if the liquid is cooled slowly enough, known as annealing, the atoms can line themselves up to form a pure crystal. This is the minimum energy state.

Let $s$ be a state represented by a list of the spatial positions of the components, and $S$ be the set of all possible states. If a system is in thermal equilibrium at a temperature T, the probability $\overline{p}_T(s)$ of being in configuration $s \in S$ depends on the energy $E(s)$ and Boltzmann's constant $k$, which relates temperature to energy. This probability follows the Boltzmann distribution giving

$$\overline{p}_T(s) = \frac{e^{-E(s)/kT}}{\sum_{w \in S} e^{-E(w)/kT}}. \tag{3.1.1}$$

In 1953, Metropolis *et al.* [50] applied these principles for use in numerical calculations. The Metropolis algorithm gives the probability $p$ of moving from configuration $s'$ at time $t$ to a randomly selected configuration $s$ at time $t+1$ as

$$\begin{aligned} p &= \frac{\overline{p}_T(s)}{\overline{p}_T(s')}, && (3.1.2) \\ &= e^{\frac{-(E(s)-E(s'))}{kT}}. && (3.1.3) \end{aligned}$$

Hence, if

- $E(s) \leq E(s')$, then as $p \geq 1$, configuration $s$ is automatically accepted.
- $E(s) > E(s')$, then as $p < 1$, configuration $s$ is accepted with probability $p$.

Thus higher energy configurations can be accepted to avoid entrapment in local minimums, but this becomes less likely as the temperature decreases and low energy states dominate due to the Boltzmann distribution. As $t \to \infty$, the probability of being in configuration $s$ is $\overline{p}_T(s)$, regardless of the starting configuration. Thus the distribution of configurations generated converges to the Boltzmann distribution, see [26].

Enough time must be spent at each temperature, especially freezing point, to reach thermal equilibrium, otherwise the probability of obtaining a very low energy configuration is reduced. This is known as the annealing schedule.

For a given sequence of temperatures $\{T_t\}$, Geman and Geman [26] showed that the annealing schedule with $T_t \to 0$ as $t \to \infty$ and $T_t > c/\log(t)$ for a large constant $c$ is sufficient for convergence. The probability that the system is in configuration $s$ as $t \to \infty$ is then $\overline{p}_0(s)$ but to get acceptable results less conservative schedules can be used.

In optimisation problems, the energy function is the objective function, the configuration is the solution representation, the temperature is the control parameter which regulates the probability of accepting a cost increasing move, and a low energy state is a near optimal solution. There are four components of the simulated annealing algorithm which are listed below.

1. The representation of solutions.
2. A move set to replace one solution by another.
3. An objective function.
4. An initial value for the control parameter $T_0$ and an annealing schedule to lower $T_t$ after a certain number of attempted or accepted moves.

These are now discussed in more detail.

### 3.1.1 Representation

The encoding used to represent the solutions to the problem can influence the simulated annealing algorithm greatly. It must be chosen with the other components of the simulated annealing algorithm, such as the move set and the evaluation, in mind. Any possible solution must be able to be encoded and decoded quickly and easily. It is preferable that each solution is represented uniquely to avoid unnecessarily enlarging the solution space. The representation must facilitate quick and easy moves as the simulated annealing algorithm will try a very large number of moves even though it will only accept a small proportion of these. For the same reason the encoded solution's evaluation, or the difference between it's evaluation and the current solution's evaluation, must be able to be calculated quickly. Small changes to the representation should also cause small changes to the solution it represents to allow the algorithm to take both small steps, when nearing the optimal solution, and large steps, when further away.

### 3.1.2 Move set

Let $S$ be the set of all feasible states. Replacing the current solution $s \in S$ with another solution $s' \in S$ is known as a move. For each $s \in S$ there is a set of possible moves that can be made. These define the neighbourhood of $s$. The move set must be such that any state can be reached from any other state via a sequence of moves. Like the representation, the move set is very important and as previously discussed, the moves and their evaluations must be able to be performed quickly and easily.

### 3.1.3 Objective function

Each of the solutions should be associated with a unique value via an objective function.

### 3.1.4 Schedule

The main concepts of the annealing schedule have been mentioned in the discussion of statistical mechanics. It was noted there that the simulated annealing algorithm works by starting with an initial temperature $T_0$ which is decremented after a certain number of moves until a stopping criteria is met. Whilst moves that cause a cost decrease are always accepted, moves which cause a cost increase are accepted with probability $p$ which is dependent on the magnitude of the cost increase, Boltzmann's constant, $k$, and the current temperature, $T$. As $T$ decreases the probability $p$ of accepting a cost increasing move also decreases.

There are several ways of determining how many moves should be investigated at a certain temperature, known as a step. One is to choose a certain number of solutions to be accepted before decreasing the temperature. In this case the number of solutions examined at each step will increase as the temperature decreases and the criteria for accepting moves becomes tighter. Another method is to examine a number of moves comparable to the size of the neighbourhood. It is also possible to use experimentation. One way of doing this is to slowly increase the number of moves taken at each temperature. When the average scores of the solutions are independent of the number of moves made, it is possible to assume that enough moves have been made to scan the cost distribution at equilibrium.

There are also several ways of choosing a method of reducing the temperature at each step. A quick and simple method is to replace $T_k$ by $\alpha T_k$. Several authors have used this method, amongst them, Ackely [3, p177] and Press *et al.* [56] with $\alpha = 0.9$, and Ersoy [21] and Aarts and Korst [1] with $0.75 \leq \alpha \leq 0.99$. Other methods suggested by Press *et al.* include selecting a total number of moves $K$ to investigate and after $m$ moves at that temperature and $k$ moves in total, set $T_{k+1} = T_0(1 - k/K)^\alpha$ where $\alpha = 1, 2$ or 4. Larger values of $\alpha$ cause the simulated annealing algorithm to spend more iterations at a lower temperature. Press *et al.* also suggest that after every $m$ moves, $T_{k+1} = \beta(f_1 - f_b)$, where $\beta$ is a constant of order 1 determined by experimentation, $f_1$ is the best function value at the current temperature and $f_b$ is the best function value so far. Another method would be to consider the number of moves that are being accepted at each temperature and the cost distribution at that step.

The initial temperature $T_0$ should be chosen such that the majority of moves are accepted. This can be calculated by generating random solutions and setting $T_0$ to be considerably larger than the largest difference in cost encountered.

As $T \to 0$ the simulated annealing algorithm moves between solutions with costs close to or equal to the global minimum. A possible stopping criteria is when the cost remains constant, that is, no moves are accepted, after a fixed number of changes in temperature. It is possible to then examine all the solutions in the current solution's neighbourhood or to restart the algorithm with the current solution. The results of restarting the algorithm havebeen examined in [56] where it is found to be beneficial on some occasions and not on others.

The following schedules are amongst those that have been used for optimisation problems. Ackley [3, p177] used $T_0 = 100, T_{\text{min}} = 0.1, \alpha = 0.9$ and set the number of moves per temperature to be $n$, which is the number of variables in the problem. Selman and Hirst [60, p151] used $T_0 = 10000, T_1 = 4.0, T_2 = 2.0$ and then decreased $T$ by 0.2 until $T = 0.6$. The number of moves per temperature was 2000. Touretzky and Hinton [63, p164] used $T = 300$ for 2 steps , $T = 32$ for 10 steps , and $T = 0.1$ for 4 steps . Although this is not a true annealing schedule they reported obtaining good results. Press *et al.* [56] changed the temperature $T$ after $100n$ moves or $10n$ successful moves.

A rule of thumb suggested by Ackley [3, p173] that can be used if the simulated annealing algorithm is to be used for many problems is to choose parameters such that they have constant value, or are dependent on $n$ (dimensionality of the function space) or the value depends on the other parameters. These are then hand-tuned. However, Ackely also notes that "It is perfectly possible that there are parameter values that would produce better average performance than the values I arrived at. Parameter tuning is more of an art than a craft."

Hajek [32] showed that for simulated annealing to find the global optimum an infinite number of moves are required.

### 3.1.5 Application to Markov chains

As seen in the previous section, the schedule is usually determined by trial and error. By modelling the simulated annealing algorithm as a Markov chain, and using some weak assumptions, a schedule can be devised which is problem independent. In simulated annealing there is a current state. The probability of being in this state depends only on its score, the score of the previous state and the value of the control parameter $T$. Thus, the sequence of states generated is a Markov chain in which the next state does not depend on the states that preceded the current state. The transition probability that $s$ will be the next state given that $s'$ is the current state is $\tau(s, s', T)$. Two states that are not connected by a move have a transition probability of zero. By requiring chain reversibility, and symmetric and reflexive moves, the simulated annealing algorithm can be modelled as a homogeneous Markov chain, see [53].

Using expected values, average scores, score variance and the accessibility of the states so far in the algorithm's run, the initial value, the decrements and the final value of $T$ can be estimated and updated during the run of the algorithm.

Although this guarantees good results, it appears that there is little necessity for using this more complicated schedule as, in general a suitable schedule can be found with relative ease.

### 3.1.6 Heuristic

It is also possible to use a heuristic at the start or end of the algorithm. Using a heuristic at the start of the simulated annealing algorithm is similar to starting from a local minimum. Therefore it is possible to use a lower temperature than usual, and it is as though the algorithm is being started part way through.

A heuristic can also be used at the end of the algorithm with the final tree as input. The heuristic then attempts to improve upon this solution.

## 3.2 Simulated annealing for the OCST problem

### 3.2.1 Representation

Among the possible representations in Section 2.2.1 that could be used for the OCST Problem, we chose the **predecessor representation** with node one being the root node.

### 3.2.2 Move set

The **move** from one solution to another consists of replacing a randomly selected link from the tree with another belonging to its fundamental cutset. This is equivalent to adding a link not in the tree and removing a link in the cycle that is formed. The links in the cycle may need to be redirected if the link added has node one as an endpoint. The other links will not need to be redirected. This move ensures feasibility and allows quick calculation of $\Delta$, which is the change in cost between the current solution and the one being examined.

As any link may be replaced by any other link that creates a feasible tree using this representation, using node one as the root node does not restrict the move set in any way.

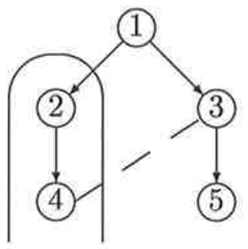
We also tried restricting the moves to a smaller set where the only links that can be removed are those that are incident with a leaf node. This still allows every possible solution but leads to premature convergence due to the small step size and restricted move set.

### 3.2.3 Initial tree

The initial tree can be selected in several ways. It can be randomly generated, the lowest cost star can be used, or the heuristic can be used as in Sections 2.2.2 and 2.3. The cost of this initial tree is then calculated by finding the $n-1$ fundamental cutsets and the traffic crossing them using equations (2.2.1) and then using (2.2.2). The traffic on each cutset is recorded.

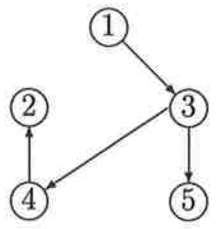
### 3.2.4 Objective function

A move is then made from this tree $T_1$ to a new tree $T_2$. The change in cost can be calculated without calculating the traffic on each of tree $T_2$'s links. The traffic on the link that is added to tree $T_2$ will be the same as that of the link that was removed from tree $T_1$ as they have the same fundamental cutset. The traffic on links that are not in the cycle will be unchanged as their fundamental cutsets will remain unchanged and thus these are known from tree $T_1$. Hence it is only the links in the cycle, excluding the link that was added, for which we need to find the new fundamental cutsets, and hence the traffic they carry. This makes finding the change in cost $\Delta$ between the two trees very quick which is necessary for simulated annealing algorithms as they try a very large number of moves.



| Link | $X$ | $\overline{X}$ |
|---|---|---|
| (1,2) | 2,4 | 1,3,5 |
| (1,3) | 1,2,4 | 3,5 |
| (2,4) | 4 | 1,2,3,5 |
| (3,5) | 1,2,3,4 | 5 |

| i | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| pred[i] | 0 | 1 | 1 | 2 | 3 |



| Link | $X$ | $\overline{X}$ |
|---|---|---|
| (3,4) | 2,4 | 1,3,5 |
| (1,3) | 1 | 2,3,4,5 |
| (2,4) | 2 | 1,3,4,5 |
| (3,5) | 1,2,3,4 | 5 |

| i | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| pred[i] | 0 | 4 | 1 | 3 | 3 |

Figure 3.2.1: Simulated annealing move

### 3.2.5 Example

Figure 3.2.1 shows how the moves are performed. The link that has been chosen to be removed is $(1, 2)$. The fundamental cutset is then represented by $(\{2, 4\}, \{1, 3, 5\})$ and consists of the links $(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)$. Any of these links can be chosen to replace the one being removed, except itself. Link $(3, 4)$ has been chosen. The links in the cycle in the predecessor array are then redirected.

Link (3,4), which was chosen to be added must now carry all the traffic link (1,2) was carrying since these two links have the same fundamental cutset. The other links in the cycle, namely (1,3) and (2,4) must have the traffic they carry re-evaluated as their fundamental cutsets have changed. Link (3,5) is not in the cycle and hence carries the same amount of traffic as before the move and does not need re-directing. The paths from the root node, node one, to the nodes in the cycle have changed causing the links in the cycle to need to be redirected. The cutsets and the predecessor arrays for both trees can be seen in Figure 3.2.1.

### 3.2.6 Schedule

Through extensive testing of the simulated annealing algorithm, we decided to use a quasi simulated annealing algorithm to decide on the initial temperature. This was done by choosing the temperature which accepted $\frac{1}{3}$ of the moves at the first temperature which resulted in a solution with higher cost than the previous solution, and rejected $\frac{2}{3}$ of these solutions. This temperature was then passed to the simulated annealing algorithm. Better results were achieved using this method than the generally accepted $\frac{1}{2} : \frac{1}{2}$ method. We also ran the simulated annealing algorithm with a temperature of $10n$ and compared the results, where $n$ is the number of variables. The algorithm was tested using $100n$ steps per temperature, with each temperature $\tau$ being replaced by $0.9\tau$ after these $100n$ steps. The algorithm was allowed to run until a minimum number of 40 temperature steps had been tried or until no change occurred in the best solution after 10 temperature changes.

### 3.2.7 Simulated annealing for a similar problem

Erosy and Panwar [21] used a simulated annealing algorithm to design minimum-delay spanning tree topologies for the interconnection of LANs. They assumed that there was a certain number of LANs which need to be connected and a traffic requirement matrix. They attempted to find the spanning tree with the minimum average network delay for the given requirements. They also extended this to design the overall LAN/MAN topology.

The move set that Ersoy and Panwar used is the same as ours, but as they assumed fixed capacities, they only considered the move if it was feasible. Their control parameter was decremented using $\tau_{k+1} = \alpha\tau_k, \quad k = 0, 1, 2 \ldots$ where $0.75 \leq \alpha \leq 0.99$, after acceptance of a set number of moves. Hence, the schedule they used is also similar to ours.

They tested their simulated annealing algorithm on problems with $6, \ldots, 30$ LANs, and various traffic patterns, and compared their solutions to a local search algorithm and a lower bounding technique. The gap between the simulated annealing algorithm and the lower bounding was between 18.2 and 61.6%. They conjectured that this gap was mainly due to the lack of tightness of the lower bound.

The simulated annealing algorithm outperformed the greedy local search in all cases. They also compared their results to 10, 000 randomly generated solutions and found that the simulated annealing solution was always better than the best random solution. Ersoy and Panwar stated that they believed, based on these comparisons, that the simulated annealing algorithm results were very close to optimal. As their implementation and schedule are similar to ours, this gives us further faith in this method.

# Chapter 4

# Genetic Algorithms

## 4.1 Genetic algorithm background

A genetic algorithm is a random search technique which derives from the idea of biological evolution. A population evolves and changes by parents having children which replace the previous generation. The new generation is hopefully fitter than the previous one due to a bias which causes fit parents to produce more offspring than those parents which are less fit.

Genetic algorithms are different from other optimisation procedures because they work with a set of solutions rather than an individual current solution.

In a genetic algorithm the parents and children are represented by **chromosomes**, each of which encodes a particular solution to the problem in question. The cost of each chromosome is calculated by an evaluation function. A **fitness function** is then used to *scale* the evaluations so that the best chromosomes do not dominate the population.

An **initial population** of size $P$ is generated first. Each of the $P$ chromosomes is chosen to be either a random feasible solution to the problem or a heuristically created solution encoded as necessary. A number of children, $C$, no larger than $P$, to be created at each generation is also chosen. $P - C$ is known as the **generation**

**gap**. These children are created by combining or changing parents via reproductive methods called **operators**, where each of the operators has a certain likelihood of being used.

When $C$ children have been created any duplicate children are removed and new children created as before to replace them. These $C$ distinct children now replace the least fit $C$ parents in the generation. The $C$ new members of the population are then evaluated and the new generation is complete. This process of creating $C$ children and updating the population continues until a user-specified total of $T$ chromosomes has been created. The best solution found is recorded and updated during the run of the algorithm. At the start of the run the chromosomes are extremely varied. Towards the end the population begins to converge.

The number of children to be created, the population size, the total number of children to create, the generation gap and the operator rates are the main **parameters** of the genetic algorithm.

In the next five sections we will look at the main components of the genetic algorithm in greater detail.

## 4.1.1 Chromosome encoding

As noted earlier, solutions to the problem in question are encoded in a string called a chromosome. The encoding used to represent the solution must be chosen carefully as it has a large impact on the performance of the genetic algorithm. The several rules with which the encoding must comply are now described.

Each chromosome must represent a solution to the problem which can be decoded easily to allow its evaluation. Any possible solution must be able to be encoded by the representation and created using the operators. The representation should be unbiased, by representing each solution an equal number of times, preferably uniquely. The encoding should also allow good components of the solutions to be combined in an attempt to create better solutions. Small changes to the chromosome should only cause small changes to the solution which it represents. This requirement is known as locality.

If a chromosome represents an infeasible solution, it can either be repaired to represent a feasible solution or left as is. Repairing chromosomes is time consuming and allowing infeasible solutions may confuse the genetic algorithm. Thus most genetic algorithm theory suggests that allowing infeasible solutions is best avoided. Davis [16, p88] stated that "An algorithm that generates many illegal solutions will perform worse than one that generates no illegal solutions".

The most frequently used genetic algorithm encoding, and the one about which there is the most theory, is the bit string. (For example, if the decimal number 11 is encoded as a bit string it would be 1011.) However, many researchers have obtained better results by using other encoding techniques. Davis [16, p62] explained that the reason that bit strings retain their popularity in spite of this is that their simplicity makes them easy to create and manipulate and allows genetic algorithm theories to be more easily proven. Another advantage is that, as they are not problem specific, they allow a genetic algorithm to be used for multiple problems without modification.

The success of genetic algorithms is due to their ability to combine the good components of solutions to create better ones. Goldberg [30, p19] and Holland [34], [35] described these solution components as *schemata*, and the set of possible values an element of a chromosome may contain as the *alphabet*. A bit string encoding has the alphabet {0,1,*}, where * is known as the *don't care* symbol and 00110 and 10111 contain, for example, the schema *011*. Alternatively, we can say that the schema *011* represents the set of strings {00110, 00111, 10110, 10111}.

Goldberg [30] gave a rule for choosing the alphabet called the "Principle of Minimal Alphabets". This suggests that the smallest alphabet which can represent the problem should be used. Using this principle encourages use of the binary alphabet since the smallest alphabet that can be used to represent any problem would have two symbols. Goldberg also states that the binary encoding maximises the number of schemata available to the genetic algorithm. This can make recognition of patterns that lead to good solutions easier.

### 4.1.2 Evaluation and fitness functions

Each chromosome is uniquely associated with a score obtained by an evaluation function. In optimisation problems this is the objective function. Using these values as the fitnesses of the chromosomes, however, may lead to fit chromosomes dominating the population to too great an extent, causing a loss of diversity and entrapment in a local minimum. For this reason the evaluations are scaled to obtain the fitnesses. There are several ways of doing this, including linear scaling, sigma truncation and power law scaling [30, p124], [16], [18].

Let us define $f$ to be the fitness or evaluation, $f'$ to be the scaled fitness and $\overline{f}$ to be the average fitness.

In linear scaling, the scaled fitnesses are $f' = af + b$. The coefficients $a$ and $b$ are chosen such that the evaluation and scaled average fitnesses are the same, and the maximum scaled fitness is an integer multiple of the average fitness. Caution must be taken to avoid negative fitnesses.

Sigma ($\sigma$) truncation uses population variance information where $\sigma$ is the standard deviation of f. Fitnesses are set using the equation $f' = f - (\overline{f} - c\sigma)$, where $c$ is a constant which is a reasonable multiple of the population standard deviation (usually between 1 and 3). Negative results are then arbitrarily set to zero.

In power law scaling, the scaled fitness is some specified power of the raw fitness $f$. Thus $f' = f^k$ where $k$ is problem dependent and may need to be changed during the course of the run.

Baker [8] also tried a method in which each chromosome in the population was given a fitness value which was a function of the chromosome's objective function value only. He showed that this method gives the same resistance to early convergence and domination as normal selection schemes used with scaling procedures.

Goldberg [30] noted that "This method essentially disassociates the fitness function from the underlying objective function; however, the direct link assumed between fitness and objective function is not grounded in theory and the ranking procedure does provide a consistent means of controlling offspring allocation. "

Thus, chromosomes are ordered by their evaluations from best to worse (in a minimisation problem this is from least costly to highest cost). They are then assigned a fitness from $F$ to $f$ decreasing by $(F - f)/(P - 1)$, for instance. This gives a larger spread in the solutions than using $1, 2, \ldots, P$ as the fitnesses. Some higher weighting may be given to the best chromosomes to encourage best schemata to reproduce and combine.

### 4.1.3 Selection of mates and operators

There are several ways of choosing chromosomes to mate. Goldberg [30, p122] listed six of these which we will discuss below. The expected number of offspring for each string is $e_i = f_i/\bar{f}$ assuming that the entire population is reproduced each generation.

1. Stochastic selection with replacement. This is also known as roulette selection where each parent is allocated a sector of the roulette wheel depending on their fitness. The wheel is then, in effect, spun to choose a parent.

2. Stochastic sampling without replacement. Each time a string is selected for mating, it's expected offspring count is decreased by 0.5. When the offspring count is less than zero, the individual is not available for selection. This forces the number of offspring for each string to be less than $f/\bar{f} + 1$.

3. Deterministic sampling. Each string receives $\text{int}(e_i)$ offspring. The population is then sorted according to the fractional part of $e_i$ and the remainder of the offspring are drawn from the top of the list.

4. Stochastic remainder sampling with replacement. Each string receives int($e_i$) offspring. The fractional parts of the expected values are used in a roulette wheel to give the remainder of the offspring.

5. Stochastic remainder sampling without replacement. Each string receives int($e_i$) offspring. The fractional parts of the expected values are treated as probabilities. Bernoulli trials are performed using the fractional parts as success probabilities. For instance, if a string has an expected value of 1.5 then it gets one copy for certain and another with probability 0.5. This continues until the population is full.

6. Stochastic tournament. Successive pairs of individuals are selected using a roulette wheel. The string with higher fitness is inserted into the population.

Goldberg states that stochastic remainder sampling without replacement appears to be the mostly widely used and accepted method.

## 4.1.4 Initial population

It is widely accepted by most genetic algorithm practitioners that the initial population should be randomly generated. There are two main reasons for this. Firstly, much of the work done on genetic algorithms to date involves their utility under the most challenging circumstances, where there is no problem specific knowledge available. However, Davis and Steenstrup [17, p3] suggested that "For industrial applications, it may be expedient to initialize with more directed methods."

The second reason for random generation is that incorporating high value chromosomes into the initial population, known as seeding, can lead to early convergence due to good schemata dominating the population. However, it has been acknowledged by Grefenstette [31, p45], Braun [12, p131] and Lienig and Brandt [47, p596] amongst others, that as long as there is sufficient variety in the initial population, seeding can be a valuable tool for the genetic algorithm. Used in conjunction with operators and operator rates that encourage diversity and a fitness function that does not allow fit chromosomes and schemata to dominate, seeding can speed up convergence and help to obtain better solutions.

### 4.1.5 Reproductive operators

Reproductive operators are used as a means of increasing the fitness of the population by changing a chromosome by some means or by combining together the good schemata. The most commonly used operators are crossover and mutation. Problem specific heuristics are sometimes used as operators to incorporate problem specific knowledge into the genetic algorithm. These operators are now discussed in more detail.

#### Crossover

Crossover is used to combine together the good schemata in chromosomes. Generally, two parents are chosen to mate creating two children which have some of the characteristics of each parent. It is hoped that the child will receive the best of the schemata from each parent thus creating a fitter child chromosome.

There are many different ways of performing crossover but the ones which are most often used, and about which there is the most theory, are 1-point, n-point and uniform crossover. These have been designed for, and mainly used in conjunction with, binary string representations. In 1-point crossover two parents are selected and a crossover point is randomly chosen. One child receives the schemata of the first parent before the crossover point and the schemata of the other parent after the crossover point. The second child receives the opposite of this. For example, performing crossover on the two bit strings below

$$\begin{array}{cccccc|cccccc} 1&1&0&1&1&0&1&1&1&0&0&0\\ 0&1&0&0&1&1&0&1&0&1&0&1 \end{array}$$

$$\uparrow$$

crossover point

gives

$$\begin{array}{cccccc|cccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0
\end{array}$$

Similarly, n-point crossover is performed by selecting n crossover points with schemata taken from each parent in turn at these points. Uniform crossover differs in that each bit ~~or gene~~ of the child chromosome is taken randomly from either parent 1 or parent 2. In the example below the crossover pattern indicates from which parent child 1 receives each bit (with child 2 receiving the opposite). A '-' indicates the bit is the same in both parents. Performing uniform crossover on two strings where

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | parent 1 : | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| | parent 2 : | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| with | crossover pattern : | 2 | - | - | 2 | - | 2 | 1 | - | 1 | 2 | - | 1 |
| gives | | | | | | | | | | | | | |
| | child 1 : | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| | child 2 : | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

One-point crossover works on the assumption that interacting genes are placed together on the chromosome, and so uniform crossover is much more disruptive as it breaks up these genes. However, for many problems it is not known at the outset which genes are interacting and should be placed together. Thus in uniform crossover, whether the schemata are short or not is irrelevant. With 1-point crossover this is an important requirement for the encoding.

Until recently much of the work done involving genetic algorithms used 1-point crossover. More recent work has investigated the ability of uniform crossover and has found it to be superior in several situations. One of these is when disruption is necessary due to the population converging. For instance, when the population size is small the genetic algorithm will converge more quickly, possibly to a local minimum. This is overcome by using uniform crossover which allows the genetic algorithm to visit a more diverse range of solutions in the search space.

Uniform crossover is also more likely than 1-point crossover to produce children that differ from their parents when the population is converging. De Jong and Spears [19, p43] stated that "long term performance can frequently be improved at the expense of short term performance by selecting more disruptive crossover operators."

Syswerda [62] and Eshelman *et al.* [22] also investigated the use of uniform crossover and both tested it against 1 and 2 point crossover on several problems. They both concluded that in general it is superior and when in doubt should be used.

### Mutation

The **mutation** operator changes one parent in some way to form a child. Mutation introduces variety into the population to avoid premature convergence, and possible loss of solutions, or parts of solutions ([30]).

Mutation changes a chromosome in an attempt to introduce diversity into the population. In binary string mutation, a bit is mutated in a chromosome by flipping its value from a 1 to a 0 or vice versa. Mutation allows schemata, other than those already present, to be introduced into the population and can cause relatively small or large changes to the evaluation of the chromosome and the solution it represents. For instance, using binary notation and simply letting it represent its equivalent base 10 number, gives $0111 = 7$. Mutating the first bit to a 1 gives $1111 = 15$, whereas mutating the last bit gives $0110 = 6$. Bit mutation is usually applied after crossover to the chromosomes that are created. Each bit is mutated with a certain, usually small, probability.

### Heuristic

In hybrid genetic algorithms, a heuristic is incorporated to provide guidance to an otherwise random search technique. This can take the form of an operator known as knowledge based mutation, where a parent is altered via the heuristic, [61], or the heuristic can be applied to the chromosome of the final generation which has the best evaluation [47].

Another alternative is applying the heuristic to every member of every generation ([16, p57],[62],[64]). Inayoshi and Manderick [37, p623] used this method and, comparing their results with a local search method, simulated annealing and a non-hybrid genetic algorithm, find the hybrid genetic algorithm, in conjunction with uniform crossover, consistently outperforms the other algorithms.

Jog *et al.* [39, p113] investigated whether this superior performance of hybrid genetic algorithms is simply due to the effectiveness of the heuristic or a combination of the properties of the genetic algorithm and the heuristic. They concluded that incorporating the features of the genetic algorithm, such as crossover, substantially improved the performance with convergence time decreasing. The combination of the heuristic with operators such as crossover allows good schemata to be combined early whilst still allowing worse offspring to be introduced thus avoiding convergence to a local, but not global, minimum.

Although a heuristic may outperform a genetic algorithm [33, p231], a hybrid genetic algorithm guarantees solutions at least as good as the heuristic.

### 4.1.6 Parameters

There are many parameters in a genetic algorithm which need to be set. They have a large controlling effect over how the genetic algorithm works and how well it performs. Although some work has been done on finding universally acceptable parameters for bit strings with 1-point, n-point and uniform crossover, choosing parameters is still a difficult and time consuming task about which little is known [58]. Each of the parameters is dependent on the others and the operators used, making this a still more difficult task.

The parameters which need to be set include those listed below, although there are many more which can be used or required which further influence the performance of the genetic algorithm.

The **population size** gives the number of chromosomes in each generation. In conjunction with the length of the chromosomes, it determines the size of the search space and the number of schemata present in each generation with which the genetic algorithm can work. One would think that a population size which is too small may lead to early convergence due to lack of diversity in the population, and a population size which is too large would take a long time to run and converge, although it would be more likely to reach the global minimum. Davis [16, p347] stated that his experience indicates, however, that this is not necessarily true, and that "the most effective population size is dependent on the problem being solved, the representation being used, and the operators manipulating the representation."

By performing reproductive operations on the current population to form offspring, which are then incorporated into the population, a new generation is formed. The genetic algorithm is run until a certain **number of generations** has been created. It is expected that the more generations for which the genetic algorithm is run, the fitter the population.

The **generation gap** determines how many of the current population will be replaced by their offspring. The $x$ least fit chromosomes in the current population are removed and replaced by offspring. Replacing all but one of the current population by offspring is known as elitism. Retaining a small number of the best chromosomes from generation to generation ensures that good schemata survive encouraging higher fitness in future populations. This may lead to early convergence if these fit chromosomes are allowed to dominate and for this reason the evaluations of the chromosomes are scaled.

The next two parameters relate to the operators and their frequency of use. The **crossover rate** is the probability that crossover is selected to mate two chromosomes. As crossover is used to combine the good schemata in the population, a high crossover rate will generally lead to quick, but possibly premature, convergence. In general, crossover is used to create two children. If it only creates one this will affect the rate at which it is used.

The **mutation rate** is the probability that each of the genes of a child created by crossover is mutated. There is an alternative definition which states that it is the probability that mutation is performed on one randomly chosen gene of a chromosome from the population. This definition makes it possible to keep track of how many times each operator is used by keeping the operators separate. Davis [16] suggested this as a less confusing way of defining operator rates. As mutation is used to introduce diversity and new schemata into the population, a high mutation rate will slow down convergence but maximise diversity, avoiding early convergence.

Many experiments and some theoretical work have been done on the optimal parameter settings for a genetic algorithm. In general, population sizes have ranged from 20-100, generations from 40-100, mutation rates from 0.0001-0.5 and crossover rates from 0.65-0.85. Larger populations and generation numbers have been tried, for instance [58], with such experiments recorded as taking anything up to 1.5 CPU years on a Sun.

Part of the difficulty in determining parameter values lies in their interaction. Changing one parameter changes the performance of all the others. For instance, increasing the generation gap is a guarantee that good schemata will remain in the population from generation to generation [16, p38]. This allows the mutation rate to be increased and the crossover operator used to be more disruptive. Alternatively, a larger population can be combined with operators which are less disruptive than a smaller population [58, p52]. Also using a heuristic will cause higher convergence in the population so the operator rates can be increased to increase diversity.

It can be useful to modify the operator rates over the run of the genetic algorithm. At the start when the population is random and hence diverse, a high crossover rate and a low mutation rate will guide the genetic algorithm's search. Increasing the mutation rate, and decreasing the crossover rate, towards the end of the run will avoid premature convergence by maintaining diversity. This can be done by linear interpolation with the rates at the start and the end of the genetic algorithm's run set.

Meta genetic algorithms have been used to set parameter values, with the chromosomes containing values for the parameters. The evaluation function value is determined by running the original genetic algorithm with these values. However, this is extremely time consuming as it will require the original genetic algorithm to be run many times.

It has been acknowledged that this difficulty with setting the parameters is a downfall of genetic algorithms. Davis [15, p61] noted that "It could take longer to derive parameter values tailored to one's problem than the time available for solving the problem itself." It looks unlikely that this will change. De Jong and Spears [19, p47] also noted that "there is very little likelihood of finding globally correct answers to questions such as the choice of population size and crossover operators." A final comment worth noting from [16] is "At present genetic algorithms are as much an art as a science".

## 4.2 Genetic algorithm for the OCST problem

### 4.2.1 Chromosome encoding

As discussed in the general representation section earlier, using the widely accepted bit string encoding enables us to incorporate the wealth of knowledge and experience already gained in their use. However, there are problems associated with the use of the bit string encoding for this problem.

Palmer and Kershenbaum [54] noted that, for a network with $N$ nodes, the process of transforming a tree to a bit string representation and vice versa takes $O(N^2)$ steps and that a randomly created bit string is unlikely to represent a tree. In fact, they stated that the probability that a random bit string which represents a graph with $N-1$ edges represents a tree is $O(2^{-[N(N/2-log2(N))]})$. Specialised crossover operators are also required using this representation to avoid infeasible non-tree solutions from being

| LINKS | (1,2) | (1,3) | (1,4) | (2,3) | (2,4) | (3,4) |
|---|---|---|---|---|---|---|
| LINK NOS | 1 | 2 | 3 | 4 | 5 | 6 |

$$\begin{array}{c|ccc|ccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline
(a) & 1 & 1 & 0 & 0 & 1 & 0 \\
(b) & 1 & 0 & 0 & 1 & 1 & 0
\end{array}$$

$\uparrow$

crossover here gives

$$\begin{array}{c|ccc|ccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline
(c) & 1 & 1 & 0 & 1 & 1 & 0 \\
(d) & 1 & 0 & 0 & 0 & 1 & 0
\end{array}$$

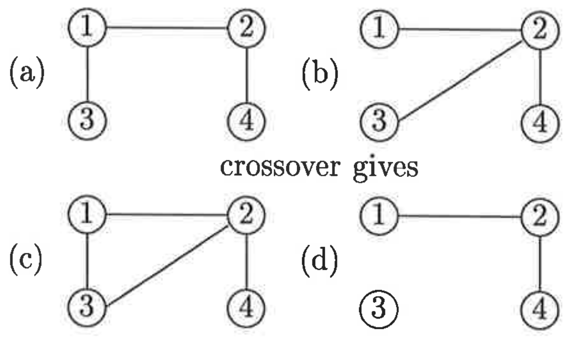


Figure 4.2.1: Binary crossover

created. An example of how this can occur is given in Figure 4.2.1 where crossover is performed on the two strings shown. Davis [16, p55] stated that "Although genetic algorithms using binary representation and single-point crossover and binary mutation are robust algorithms, they are almost never the best algorithms to use for any problem." This leads us to investigate the other encodings in Section 2.2.1.

The Prüfer representation is unique, unbiased and always represents a tree. However, this encoding possesses little locality, allowing offspring to differ greatly from their parents. Julstrom [41] used this representation for the Steiner tree problem with limited success. For this reason, this encoding will be discussed no further here.

Palmer and Kershenbaum [27] used the node and link weighted representation and tested their GA with the parameter $P_1$ set to zero. They noted in their conclusions that this "representation has an inherent bias towards star-based networks" and stated that it may be better to use one of the other possible encodings and repair the solutions.

We have chosen to use the predecessor encoding with the root node being node one. This gives a unique encoding for trees, has locality and covers the set of all solutions. The value in each gene and its position also give the actual link, making the transformation between the representation and a link list unnecessary.

The disadvantage in using this encoding is that many infeasible non-trees arise when generating a random initial population and using standard operators. This can be avoided by using simple specialised operators and initialisation techniques. These allow every feasible tree to be created that could be formed using the binary representation whilst avoiding all the infeasible tree solutions that the binary link representation allows. Thus this encoding is more compact than the binary encoding.

In a hybrid genetic algorithm it is generally accepted that the same representation should be used by the heuristic and the genetic algorithm. The heuristic we use, described in Section 2.3 uses a list of links representation which is equivalent to the predecessor array.

The main advantage of this representation, with specialised operators, over Palmer and Kershenbaum's representation is that a transformation between the encoding and the tree it represents is not necessary, whereas Palmer and Kershenbaum's encoding requires the use of Prim's algorithm for every encoding. As the predecessor array can also represent a directed graph it is easier to find the cutsets necessary for use in the evaluations. Palmer and Kershenbaum do not give details of their operators, selection of mates or test data in their paper so further comparison with the work in this thesis is not possible.

### 4.2.2 Evaluation and fitness functions

The **evaluation** of each chromosome is the cost of routing the traffic on the tree represented by the chromosome. This cost is calculated using equations (2.2.1) and (2.2.2).

As explained in Section 4.1, using these evaluations as chromosome fitnesses may allow a chromosome with a much higher evaluation than the rest of the population to dominate and cause premature convergence. To avoid this we tried using two scaling functions, rank scaling and linear scaling, see Section 4.1.2.

### 4.2.3 Selection of mates and operators

We implemented two of the selection methods from Section 4.1.3 in our genetic algorithms. They are stochastic selection with replacement and stochastic remainder sampling without replacement. The first is very simple to implement and it is easy to ensure that the two parents selected for crossover differ to avoid chromosome domination. The later is more widely accepted and ensures that a certain number of offspring are created using a certain parent. However, this method works on the principle that for each parent selected to mate, a child is created. Since our crossover operator combines two parents to create one child only, this causes some difficulty. This was circumvented by creating two children, which may be the same, by using the crossover operator twice with the same parents. Using this method, it is also harder to avoid having two parents which are the same selected for crossover. Thus this selection method appears to have more chance of causing early convergence.

### 4.2.4 Initial population

Our initial population is a collection of $P - 1$ randomly generated trees and the lowest cost star all of which are encoded using the predecessor labeling and created as in Section 2.2.2.

A diverse initial population with a guarantee of some good schemata is thus formed.

### 4.2.5 Reproductive operators

The three operators which are used by our genetic algorithm are explained below. As has been mentioned in the encoding section, specialised operators are required to guarantee feasible trees are created. Repairing chromosomes is possible with standard operators, but as noted in the general section, is best avoided.

| PRED | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| (a) | 0 | 1 | 4 | 2 |
| (b) | 0 | 4 | 1 | 3 |

crossover ↑ gives

| PRED | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| (c) | 0 | 1 | 4 | 3 |
| (d) | 0 | 4 | 1 | 2 |

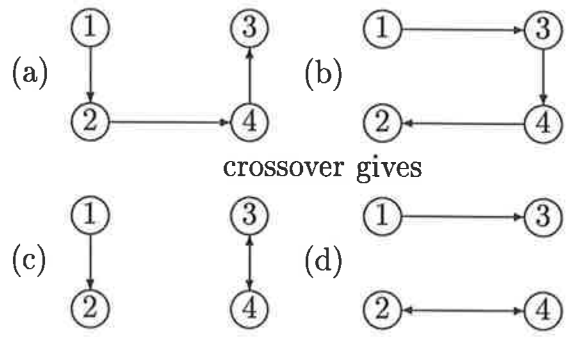


Figure 4.2.2: Invalid predecessor crossover

### Crossover

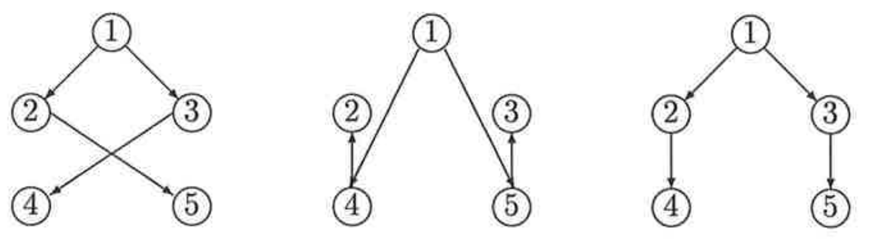
In Figure 4.2.2 we can see how it is possible to obtain invalid solutions using 1-point crossover on the predecessor representation. Other standard crossover operators also give many invalid solutions when using the predecessor representation. Hence we have devised a specialised crossover operator that always creates valid solutions, is quick and easy and is *as though* we were performing uniform crossover on the binary encoding but only such that feasible child trees are created.

This specialised **crossover operator** chooses two parents and selects links from each parent to create one child chromosome. The child is disregarded if it is a copy of either of its parents.

The procedure for combining the parents is similar to that of creating the initial population, except that the links that may be chosen are restricted to those in the two parents. For instance, if node $i$ is selected from the 'in tree' list, then node $j$ must be selected from the 'out of tree' list such that link $(i, j)$ exists in one or both of the parents. As before, the link $(i, j)$ is then added to the tree and node $j$ is moved to the 'in tree' list. An example of this can be seen in Figure 4.2.3. Only one child is created as we can only guarantee one feasible child solution by this method.

Example 4.2.4 shows two results of the specialised crossover performed on parent chromosomes (a) and (b). The genes/schemata that are taken from each parent are shown in bold face. In creating child (c) using the predecessor encoding, it can

| IN TREE | OUT OF TREE | LINKS WHICH CAN BE ADDED | PRED ARRAY 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| 1 | 2 3 4 5 | (1,2)**(1,3)**(1,4)(1,5) | 0 | | | | |
| 1 3 | 2 4 5 | (1,2)(1,4)(1,5)(3,4)**(3,5)** | 0 | | 1 | | |
| 1 3 5 | 2 4 | **(1,2)**(1,4)(3,4)(5,2) | 0 | | 1 | | 3 |
| 1 3 5 2 | 4 | (1,4)**(2,4)**(3,4) | 0 | 1 | 1 | | 3 |
| 1 3 5 2 4 | - | | 0 | 1 | 1 | 2 | 3 |



(a) (b) (c)

Figure 4.2.3: Performing specialised crossover on (a) and (b) to get (c)

be seen that the links need to be reordered. Thus, this child could not be created using a standard crossover operator with the predecessor encoding. Investigating the results of the same crossover performed using the binary link encoding, shows that it is *as though* one-point or uniform crossover is being performed. In creating child (d) using the specialised operator, the links of the predecessor array do not need to be reordered, and the crossover operator appears to be the same as one-point. The crossover using the binary link encoding appears to be uniform crossover.

It can be seen from these examples that the crossover operator is combining the underlying schemata of the binary encoding using uniform crossover. As interacting genes will not necessarily be placed together in either encoding there would be no advantage in using one-point crossover, and uniform crossover has the advantage of being more likely to create children that differ from their parents and allowing diversity. Thus we have the benefit that all the schemata of the binary encoding that can be combined to give feasible trees are available to the predecessor encoding using this operator.

| LINK MAP | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| LINK | (1,2) | (1,3) | (1,4) | (1,5) | (2,3) | (2,4) | (2,5) | (3,4) | (3,5) | (4,5) |

CROSSOVER ON (a) AND (b) TO GET (c)

| PRED | 1 | **2** | **3** | 4 | 5 |
|---|---|---|---|---|---|
| (a) | 0 | **1** | **1** | 3 | 2 |
| (b) | 0 | **4** | **5** | 1 | 1 |
| (c) | 0 | 1 | 1 | 2 | 3 |

| LINK | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| (a) | **1** | **1** | **0** | **0** | **0** | 0 | 1 | 1 | 0 | 0 |
| (b) | 0 | 0 | 1 | 1 | 0 | **1** | **0** | **0** | **1** | **0** |
| (c) | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

CROSSOVER ON (a) AND (b) TO GET (d)

| PRED | 1 | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| (a) | 0 | **1** | **1** | 3 | 2 |
| (b) | 0 | 4 | 5 | **1** | **1** |
| (d) | 0 | 1 | 1 | 1 | 1 |

| LINK | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| (a) | **1** | **1** | 0 | 0 | 0 | **0** | 1 | 1 | **0** | 0 |
| (b) | 0 | 0 | **1** | **1** | **0** | 1 | **0** | **0** | 1 | **0** |
| (d) | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

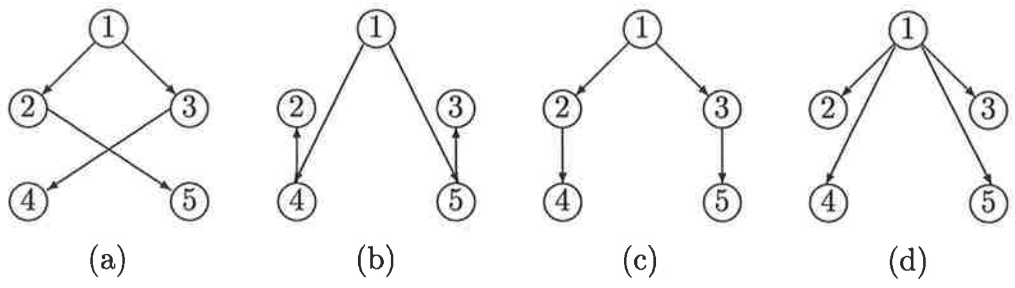


Figure 4.2.4: Predecessor crossover and binary link crossover

### Mutation

The **mutation operator** chooses one parent via roulette selection and creates a child which differs from its parent by selecting a link in the tree to remove and replacing it by an appropriate link not in the tree. This can be done as in the simulated annealing moves, but it is possible to use a variation of this which is slightly more restrictive, but does not require the cycle to be redirected.

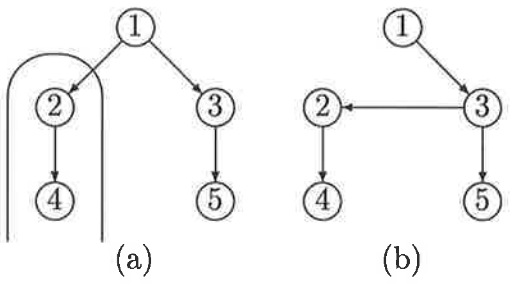
A link $(i, j)$ is chosen to be removed from the tree where pred[j]=i. The difference from the simulated annealing moves is that, when selecting a link from the fundamental cutset to replace link $(i, j)$, only links $(j, k)$ which have node $j$ as an endpoint may be selected. This gives pred[j]=k in the new tree, which is the only change to the predecessor array. The only case in which this move will not be possible is when node one is a leaf node and the link that has been chosen to be removed is $(1, j)$. In this case a new link will be selected to be removed.

This mutation method will always give a feasible tree which differs from its parent, without the need to redirect the tree. However, it does not allow as many mutation possibilities as the simulated annealing moves. Testing two genetic algorithms, one with each mutation type, for 100 problems with between 10 and 55 nodes, showed that the more restrictive mutation described here gave superior results.

If the link that is chosen to be removed is incident with a leaf node, the changes to the tree and its evaluation may be quite minor. In other cases there may be quite considerable changes in both the tree and its evaluation.

In the binary encoding this mutation operator changes one gene from a 1 to a 0 and one from a 0 to a 1 such that a feasible tree is created. In Figure 4.2.5 we can see the results of performing mutation on a tree. The link that has been chosen to be removed is $(1, 2)$. The fundamental cutset is then represented by $(\{2, 4\}, \{1, 3, 5\})$ and consists of the links $(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)$. Only links $(2, 3)$ and $(2, 5)$ can be chosen to replace $(1, 2)$. Link $(2, 3)$ has been chosen in Figure 4.2.5.

| LINK NO.S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| LINK | (1,2) | (1,3) | (1,4) | (1,5) | (2,3) | (2,4) | (2,5) | (3,4) | (3,5) | (4,5) |



| PRED | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| (a) | 0 | **1** | 1 | 2 | 3 |
| (b) | 0 | **3** | 1 | 2 | 3 |

| LINK | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| (a) | **1** | 1 | 0 | 0 | **0** | 1 | 0 | 0 | 1 | 0 |
| (b) | **0** | 1 | 0 | 0 | **1** | 1 | 0 | 0 | 1 | 0 |

Figure 4.2.5: Predecessor mutation and binary link mutation

Thus, the predecessor of 2 becomes 3.

**Heuristic**

The heuristic used is one designed by Salzborn [57]. This is explained in Section 2.3. The starting tree for the heuristic when used with the genetic algorithm is a parent chosen from the current population.

## 4.2.6 Parameters

We noted earlier that the issue of setting parameters is a very difficult one, which is dependent on the problem, the representation and the operators used. We tuned (selected) our parameters by running the genetic algorithm hundreds of times with different combinations, taking note of the diversity of the population, the convergence rate and the final solution.

By choosing the predecessor representation, we were led to choose a form of uniform crossover that always gives legal solutions. It has been acknowledged that uniform crossover is reasonably disruptive, which encourages diversity and hence allows a smaller population size to be used.

As we have a generation gap we can also have reasonably large crossover and mutation rates. This is a trade off between diversity and disruption and keeping good schemata to encourage convergence. Choosing a reasonable size generation gap also suggests that the fitness function for the chromosomes should not lead the fittest to be dominating. The smaller population size allows us to run the genetic algorithm for more generations. Hence we can see the interaction and trade-off between the parameters.

It is worth noting that, for an $n$ node problem, each chromosome has $n-1$ links represented of a possible $\frac{n(n-1)}{2}$ links for the problem. For a population size $p$ there are $p(n-1)$ links represented in the population and hence with $p = n/2$ it is possible to have every link (not every schemata) present in the population.

Using the rank fitness compared to the linear fitness method gave similar results as did using the stochastic selection with replacement compared to stochastic remainder sampling without replacement. The results that will be shown used the rank fitness and stochastic selection with replacement.

We chose the genetic algorithm parameters by running hundreds of test problems. We tried population sizes of $n$ and $n/2$ chromosomes which were kept constant throughout the run, and created $4n/5$ and $2n/5$ children each generation for the respective population sizes. The algorithm was run until $3000n$ children had been created, which is similar to the number of solutions the simulated annealing algorithm will investigate. The crossover and mutation rates were linearly adjusted throughout the run. The mutation rate was set to 30% at the start of the run, finishing at 70%. The crossover rate was set to 70% at the start of the run, finishing at 30%. If the heuristic is used, it has a 1% chance of being used and the mutation rates are reduced by 1%.

# Chapter 5

# Results

For problems in which the number of nodes is larger than say 10, it is impractical to enumerate all the possible solutions to a problem to find the optimum. Hence, in an attempt to investigate the performance of our algorithms when the optimum is known, we ran some problems where the distances were set to one and the traffics were random. In Chapter 2 we noted that there is a polynomial algorithm (see [36]) to find the optimum for this case. It was surprising to note that the optimal solution to all our test problems was a star. We will attempt to ascertain reasons for this in the next section.

## 5.1 Polynomial case

For an $n$ node problem there are $n^{n-2}$ possible trees, where $n$ of these are stars (each node can be the hub of a star). This means that $100n/n^{n-2} = 100/n^{n-3}\%$ of the total solutions are stars. For example, for $n = 10$, $1/10^5\%$ of the total solutions are stars. This is a very small percentage, however when using randomly allocated evenly distributed traffics on 160 random test problems (for $n = 10, \ldots, 50$), the optimal solution was always a star. This is a striking result since star networks form such a small percentage of the possible solutions. If we consider the cost of a star and a non-star, the reasons for this result become clearer.

The cost of a tree, $T$, is

$$C_T = \sum_s l_s^T \nu_s \tag{5.1.1}$$

where $l_s^T = \sum_{j \in p_s} m_j$, which is the cost per unit of traffic of the unique path for stream $s$ in tree $T$. Let $T_S$ be any star tree, $T_N$ be any non-star tree. If the cost per unit of traffic for each link is set to be one, then $l_s^T$ is the number of links in the path the stream $s$ uses. Let this be $h_s^T$, the number of hops. For any tree $T$, if stream $s$ is connected by a direct link in the tree, then $h_s^T = 1$. For a star tree, any stream $s$ not connected by a direct link in the tree will be connected by a two link path, that is $h_s^{T_S} = 2$. For a non-star tree, any stream $s$ not connected by a direct link in the tree will be connected by a path which is two or more links long, that is $h_s^{T_N} \geq 2$. At least one path will consist of more than two links, otherwise the tree will be a star. Thus, there exists a stream $s$ such that $h_s^{T_N} > 2$.

The total number of streams for a network is $\frac{n(n-1)}{2}$. The number of streams with a direct link in a tree is $n-1$, the number of streams with a non-direct link is $\frac{(n-1)(n-2)}{2}$. Let us define $s \in T$ to mean that stream $s$ has a single link path in the tree, and $s \notin T$ to mean that stream $s$ does not have a single link path in the tree.

For $m_j = 1$, for all $j \in \mathcal{J}$, and $\nu_s = \nu$, for all $s \in S$, the cost of a star tree, $T_S$, is

$$\begin{aligned} C_{T_S} &= \nu \sum_{s \in T_S} 1 + 2\nu \sum_{s \notin T_S} 1 \\ &= \nu((n-1) + (n-1)(n-2)) \\ &= \nu(n-1)^2. \end{aligned}$$

The cost of a non-star tree, $T$, is

$$\begin{aligned} C_{T_N} &= \nu \sum_{s \in T_N} 1 + \nu \sum_{s \notin T_N} h_s \\ &= \nu \sum_{s \in T_N} 1 + 2\nu \sum_{s \notin T_N} 1 + \nu \sum_{s \notin T_N} (h_s - 2) \\ &= \nu((n-1) + (n-1)(n-2)) + \nu \sum_{s \notin T_N} (h_s - 2) \end{aligned}$$

$$
\begin{aligned}
&= \nu(n-1)^2 + \nu \sum_{s \notin T_N} (h_s - 2) \\
&= C_{T_S} + \nu \sum_{s \notin T_N} (h_s - 2).
\end{aligned}
$$

Thus, as there is an $h_s > 2$ for at least one stream $s$ in the non-star tree, the non-star tree is more expensive than the star tree.

For $m_j = 1$, for $j \in \mathcal{J}$, and $\nu_s$ random, for $s \in S$, the cost of a star is

$$
\begin{aligned}
C_{T_S} &= \sum_{s \in T_S} \nu_s + 2 \sum_{s \notin T_S} \nu_s && (5.1.2) \\
&= \sum_{s \in S} \nu_s + \sum_{s \notin T_S} \nu_s && (5.1.3) \\
&= 2 \sum_{s \in S} \nu_s - \sum_{s \in T_S} \nu_s. && (5.1.4)
\end{aligned}
$$

The cost of a non-star is

$$
\begin{aligned}
C_{T_N} &= \sum_{s \in T_N} \nu_s + 2 \sum_{s \notin T_N} \nu_s + \sum_{s \notin T_N} (h_s^T - 2)\nu_s && (5.1.5) \\
&= \sum_{s \in S} \nu_s + \sum_{s \notin T_N} \nu_s + \sum_{s \notin T_N} (h_s^T - 2)\nu_s && (5.1.6) \\
&= 2 \sum_{s \in S} \nu_s - \sum_{s \in T_N} \nu_s + \sum_{s \notin T_N} (h_s^T - 2)\nu_s && (5.1.7)
\end{aligned}
$$

where $h_s^T \geq 2$ for all $s \notin T$ and at least one $s \notin T_N$ has $h_s^T > 2$ for the tree to be a non-star.

The percentage of streams which do not have a direct link in the tree is $100\frac{n-2}{n}$. For $n = 10$ this is 80%, for $n = 50$ it is 96%. So since the majority of the total streams are included in the sums $\sum_{s \notin T_N} \nu_s$ and $\sum_{s \notin T_S} \nu_s$ (equations (5.1.6) and (5.1.3)), and the traffics are evenly distributed, it can be expected that these sums are approximately equal. Since the term $\sum_{s \notin T_N} (h_s^T - 2)\nu_s$ (equation (5.1.6)), is the summation of several (in fact it will be shown later, at least $(n-3)$) traffic streams, which may include some multiples, and this term is added only to the non-star's cost, it can be expected that the non-star tree will be more expensive.

So, if we wish to find examples in which a non-star will be less expensive than any star tree we need

$$
C_{T_S} - C_{T_N} = \sum_{s \in T_N} \nu_s - \sum_{s \in T_S} \nu_s - \sum_{s \notin T_N} (h_s^T - 2)\nu_s > 0
$$

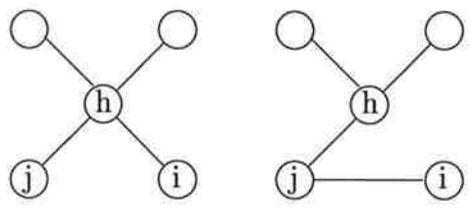


Figure 5.1.1: Changing one link in a star

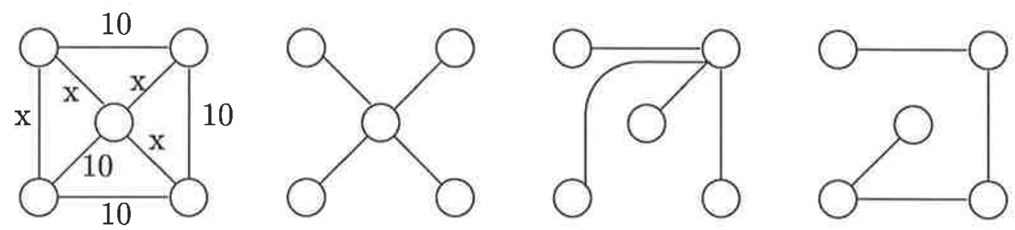
from equations (5.1.4) and (5.1.7). Hence, for a non-star to be less expensive it is necessary that there are some high traffic streams with no end-points in common, so that the majority of them cannot be included together in any star. These streams will then be given a direct link in the non-star tree. The rest of the traffic streams need to be fairly inexpensive such that the sum of the traffics on streams with paths longer than length two in the non-star plus the sum of the traffics that are direct in the star is less than the sum of the traffics that are direct in the non-star.

To see that the sum $\sum_{s \notin T_N}(h_s^T - 2)\nu_s$ (equations (5.1.5)–(5.1.7)) will involve at least $(n-3)$ traffic streams, consider the difference between a star and a non-star that differs by one link only. If link $(i, h)$ is removed from a star tree, where $h$ is the hub node, and is replaced by link $(i, j)$ to create a non-star (see Figure 5.1.1), then equation (5.1.6) gives

$$C_{T_N} = \sum_{s \in S} \nu_s + \sum_{s \notin T_N} \nu_s + \sum_{k=1, k \neq i,j,h} \nu_{ik}$$

where $\sum_{k=1,k\neq i,j,h}^{n} \nu_{ik}$ involves $(n-3)$ terms and the difference in cost between the star and the non-star tree is $C_{T_N} - C_{T_S} = \sum_{k=1,k\neq i,j}^{n} \nu_{ik} - \nu_{ij}$. A non-star which differs from any star by more than one link will have more than $(n-3)$ streams which have paths longer than 2 links.

Figure 5.1.2 gives an example in which a non-star tree is less expensive than any star tree, $(x = 1)$, and an example in which the least expensive star tree has the same cost as the least expensive non-star tree $(x = 4)$. Both examples are for $m_j = 1,\ \forall j \in \mathcal{J}$ and the traffics $\nu_s,\ \forall s \in S$ are as shown in the first diagram, with



| Traffics | $C_{T_{S1}}$ | $C_{T_{S2}}$ | $C_{T_N}$ |
|---|---|---|---|
| x = 1 | 2(44)-13=**75** | 2(44)-21=**67** | 2(44)-40+1(1) +2(1)+1(1)=**52** |
| x = 4 | 2(56)-22=**90** | 2(56)-24=**88** | 2(56)-40+1(4) +2(4)+1(4)=**88** |

Figure 5.1.2: Tree costs using equations (5.1.4) and (5.1.7)

any traffics not shown having value zero. It can be seen that the second star shown is less expensive than the first star in both cases as it includes two of the high cost links in its star. It can also be seen that the traffics in the non-star tree need to be considerably higher than those not in this tree, for the non-star tree to be less expensive.

We created several examples in which the optimal tree would not be a star, such as the one above, for different problem sizes (number of nodes $n = 5, \ldots, 50$) and ran all our algorithms on these examples. The optimal solution was found by using the polynomial algorithm mentioned at the start of this section. In every case the simulated annealing, with and without the heuristic, the genetic algorithm with the heuristic, the heuristic and the heuristic with the greedy algorithm obtained the optimal solution. The genetic algorithm did not perform as well, but was within 1% of the optimal solution in all cases. This gives us some confidence in the performance of our algorithms, when the costs, $m_j$, are not restricted to be uniformly one and thus the optimal solution is not known.

## 5.2 Computational results

We then tested our simulated annealing, genetic algorithm and heuristic (with and without the greedy algorithm) on randomly generated test data, with both Euclidean and non-Euclidean distances. To generate Euclidean distances a list of nodes and their positions on a grid was created and the distances between them calculated. The non-Euclidean distances were randomly generated. The traffic between node pairs was also randomly generated, with the majority of traffic pairs having non-zero traffic. For all the methods, input is the inter-node distance and the traffic matrix. All problems were run on a Sparc IPX.

The simulated annealing, genetic algorithm and heuristic (with and without the greedy algorithm) were run on 100 problems with Euclidean traffic and 100 problems with non-Euclidean traffic. The problems were divided up by the number of nodes $n = 5, \ldots, 50$, with 10 problems being run for each set $n, \ldots, n+5$. The simulated annealing and genetic algorithms were run twice each with different parameters, which were discussed in Sections 3.2.6 and 4.2.6. The best of the two results for each algorithm was recorded. Both algorithms were then run once each incorporating the heuristic. The heuristic and the heuristic with greedy algorithm were also run once each, and finally an algorithm to find the best star cost was run.

For problems with less than 10 nodes we were able to calculate the optimum by enumerating all possible solutions. This can be done in under ten minutes for 9 node problems. However, since the number of possible trees for a problem with $n$ nodes is $n^{n-2}$, the increase in time in using this method for larger problems is impractical.

The best solution ($n > 9$) was taken to be the best solution from all the algorithms. Each solution was then expressed as a percentage of the best (or optimal) solution for each of the ten problems. These percentages were then averaged over the ten problems to give each algorithm's average performance for that problem size. The results can be seen in Tables 5.2.1 and 5.2.2. A quick look at these tables shows that the results of some of the algorithms varies greatly depending on whether

Euclidean or non-Euclidean traffics are used. Thus we will examine the results for the two cases separately.

### 5.2.1 Discussion of results - Euclidean traffic

Table 5.2.1 shows that the genetic algorithm, with the aid of the heuristic, consistently gives the best results. All the algorithms, for all problem sizes, are within approximately $\frac{1}{2}\%$ of the best solution found. The best star is approximately 7% higher than the best solution. For problems with less than 9 nodes, where the exact solution is enumerated, all algorithms found the optimal solution every time.

The simulated annealing algorithm with the heuristic sometimes performs better than the simulated annealing algorithm and sometimes worse. There appears to be little advantage in adding the heuristic to the simulated annealing algorithm.

Adding the heuristic to the genetic algorithm is definitely advantageous. For the larger size problems, the genetic algorithm often finds the worst solution of all the algorithms. However, with the addition of the heuristic, it finds the best solution of all the methods for every problem size.

The performance of the heuristic is comparable with, but slightly better than, the genetic algorithm. Adding the greedy algorithm to the heuristic improves its performance, making it comparable with the simulated annealing algorithm and better than it for larger problem sizes.

For the small to medium problem sizes, all the methods gave their results within approximately 30 seconds. Some of the larger problems took up to 30 minutes to solve.

### 5.2.2 Discussion of results - Non-Euclidean traffic

Table 5.2.2 shows that the algorithms perform rather differently with non-Euclidean traffics. The genetic algorithm with the heuristic is certainly no longer the best method. The simulated annealing algorithm with the heuristic consistently out-

| NO OF NODE | SA AVER % | SA+H AVER % | GA AVER % | GA+H AVER % | HEUR AVER % | H+GR AVER % | STAR AVER % |
|---|---|---|---|---|---|---|---|
| 5-9 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.97 |
| 10-14 | 100.00 | 100.00 | 100.03 | 100.00 | 100.19 | 100.16 | 106.72 |
| 15-19 | 100.01 | 100.05 | 100.04 | 100.04 | 100.12 | 100.07 | 107.09 |
| 20-24 | 100.06 | 100.00 | 100.13 | 100.00 | 100.28 | 100.18 | 107.66 |
| 25-29 | 100.07 | 100.20 | 100.18 | 100.00 | 100.16 | 100.09 | 108.72 |
| 30-34 | 100.00 | 100.01 | 100.15 | 100.00 | 100.22 | 100.11 | 107.49 |
| 35-39 | 100.21 | 100.37 | 100.52 | 100.06 | 100.44 | 100.35 | 107.45 |
| 40-44 | 100.20 | 100.18 | 100.48 | 100.01 | 100.34 | 100.08 | 107.75 |
| 45-49 | 100.15 | 100.05 | 100.37 | 100.00 | 100.25 | 100.07 | 107.04 |
| 50-54 | 100.38 | 100.27 | 100.42 | 100.06 | 100.33 | 100.25 | 106.26 |

Table 5.2.1: Performances of the algorithms with Euclidean traffic.

| NO OF NODE | SA AVER % | SA+H AVER % | GA AVER % | GA+H AVER % | HEUR AVER % | H+GR AVER % | STAR AVER % |
|---|---|---|---|---|---|---|---|
| 5-9 | 100.00 | 100.00 | 100.00 | 100.00 | 108.80 | 100.00 | 169.19 |
| 10-14 | 100.00 | 100.00 | 100.00 | 100.00 | 126.72 | 100.00 | 229.98 |
| 15-19 | 100.00 | 100.00 | 100.00 | 100.50 | 154.90 | 100.00 | 330.43 |
| 20-24 | 100.02 | 100.05 | 100.10 | 100.57 | 171.84 | 100.30 | 392.18 |
| 25-29 | 100.00 | 100.00 | 100.00 | 101.33 | 190.72 | 100.14 | 465.74 |
| 30-34 | 100.00 | 100.07 | 100.00 | 102.76 | 198.95 | 100.00 | 502.18 |
| 35-39 | 100.02 | 100.01 | 100.17 | 107.18 | 224.62 | 100.48 | 566.17 |
| 40-44 | 100.36 | 100.00 | 100.85 | 106.98 | 211.74 | 104.82 | 566.83 |
| 45-49 | 100.20 | 100.00 | 101.22 | 105.76 | 244.81 | 102.95 | 727.82 |
| 50-54 | 100.32 | 100.19 | 101.06 | 108.34 | 284.69 | 105.76 | 832.88 |

Table 5.2.2: Performances of the algorithms with non-Euclidean traffic.

performs the other methods, although its performance is very close to that of the simulated annealing algorithm.

The genetic algorithm performs very well for the small problem sizes but not quite as well for the larger problems. The addition of the heuristic to the genetic algorithm causes it to perform worse. This is due to the fact that the performance of the heuristic is extremely bad and thus it misguides the genetic algorithm.

The greedy algorithm improves the performance of the heuristic greatly. The reason for this and the poor performance of the heuristic can be seen in the average star percentage. These show that the solutions of these problems are certainly not star based. The heuristic is based on creating sub-trees which are two-hub trees and hence may lead to star based solutions.

## 5.3 Conclusions

It is worth noting that although extensive testing has been used to determine the parameters for our methods, other choices of parameters may lead to improvements in performance of some methods. Method performance may also depend on the characteristics of the solution space, the problem type, and the data used as input.

In Sections 3.1.4 and 4.1.6 the difficulty of setting the parameters for both solution methods was discussed. It is interesting to note that there are quotes about both methods stating that setting the parameters is more an art than a craft or science. Simulated annealing has fewer parameters than genetic algorithms though and appears to be less dependent on the parameter choice.

The performance of the algorithms is also very dependent on the solution space and therefore no general conclusions should be drawn about which method is better for any problem in general. Ackley [3, p177] performed a large study comparing genetic algorithms and simulated annealing for several problem types and concluded that none of the strategies was the best all the time as their performance is dependent on the search space. Schweitzer *et al.* [59, p944] discussed this dependency on the search space stating that "the Boltzmann strategy is able to detect the appropriate potential minima even in an unknown, rugged landscape as long as the potential barriers between local minima are not too high, which forces the locking in side minima. On the other hand, the Darwin strategy is able to cross high barriers by tunneling if the next minimum is close enough."

It is possible to determine which method is better for a particular problem though if the parameters are chosen carefully. For this problem it appears that the best solution method, if the traffic is Euclidean, is the genetic algorithm combined with the heuristic. If the traffic is non-Euclidean the simulated annealing algorithm in conjunction with the heuristic performs best. However, in both cases the simulated annealing by itself performs very well, and is a good choice as it is reasonably simple to find reasonable parameters and implement.

# Part II

# Maximal Profit Dimensioning and Tariffing

In this part of the thesis an approach to optimal dimensioning and tariffing of communication networks is presented. The link capacities, tariffs and the routing strategy are chosen in order to maximise the profit for the company operating the network. The tariffs and grade of service are subject to regulatory constraints. It is assumed that there is an existing network structure consisting of a set of nodes and physical links. By cross-connecting traffic through nodes at a high bandwidth rather than multiplexing and de-multiplexing it, a logical link (consisting of capacity on several physical links) is created. However, it may be more efficient for an OD pair to take advantage of existing physical links rather than to initiate its own logical link. Several results will be presented. These include a simple formula for the optimal tariff and a result that only one of the possible routes for each OD pair will be used in the optimal solution. A numerical investigation will also be discussed.

# Chapter 6

# Problem formulation

## 6.1 Introduction

Loss networks can be used to model circuit switched telephone networks, along with many other practical networks. A loss network consists of sets of resources accessed by users of different types. If the required resources are not available, one or more alternative sets may be tried, but ultimately a user whose request cannot be satisfied is lost from the system. Many references to research on the dimensioning of loss networks were given in Bean and Taylor [10] (see for example Girard [29] and Kelly [44]). This work is an extension of the work presented there and in Bean *et al.* [9].

As explained in Bean and Taylor [10], the usual method used in loss network dimensioning is to minimise network cost subject to grade of service constraints. However, the approach used there, and also here, maximises the network profit which is the difference between the revenue generated by network users and the cost of providing the network. This takes into account the fact that many telephone companies are now operating in a private enterprise environment.

Although this approach has been discussed previously (see Kelly [43] and Girard [29]), Bean and Taylor [10] and Bean *et al.* [9] incorporated the concept of a traffic elasticity function. This function acknowledges the fact that the traffic offered to the network is a decreasing function of the tariff charged to users. Such a traffic elasticity function could reflect factors such as a competitor's tariff structure or the grade of service offered by the network.

In this thesis we consider the introduction of *logical* links. A logical link consists of reserved capacity on a set of two or more physical links. Traffic using the logical link is cross-connected through intermediate nodes rather than being multiplexed and de-multiplexed at these nodes. For instance, in a possible Australian network (see Figure 6.1.1), traffic travelling from Perth to Melbourne on the logical link would use reserved capacity on the Perth to Adelaide and Adelaide to Melbourne links and be cross-connected through Adelaide. Although this may be an advantage in some cases, it also introduces inefficiencies as the reserved capacity is allocated when the network is configured rather than being used on demand.

In the next sections we define our model and discuss methods of network analysis. We then look at the optimisation procedure and give a numerical example. Chapter 7 contains theoretical results concerned with optimal tariffing and route choice. In Chapter 8 we define a model in which more than two path choices are allowed, and give both theoretical and numerical results.

## 6.2 The model

In this section we introduce the notation and concepts of a circuit-switched telephone network. Both physical and logical links may be used. Note that a list of the notation used is given in Appendix A.

Consider a network in which there is a group of cities or nodes $n \in N$ joined by a set of links $j \in \mathcal{J}$. This set of links $\mathcal{J}$ consists of two subsets: the physical links $\mathcal{J}_P$ and the logical links $\mathcal{J}_L$, which use reserved capacity on the physical links. Each link $j$ comprises $C_j$ circuits and has a link blocking probability $E_j$. Define $S$ to be the set of all streams (origin-destination pairs), $S_P$ to be the set of streams connected by a

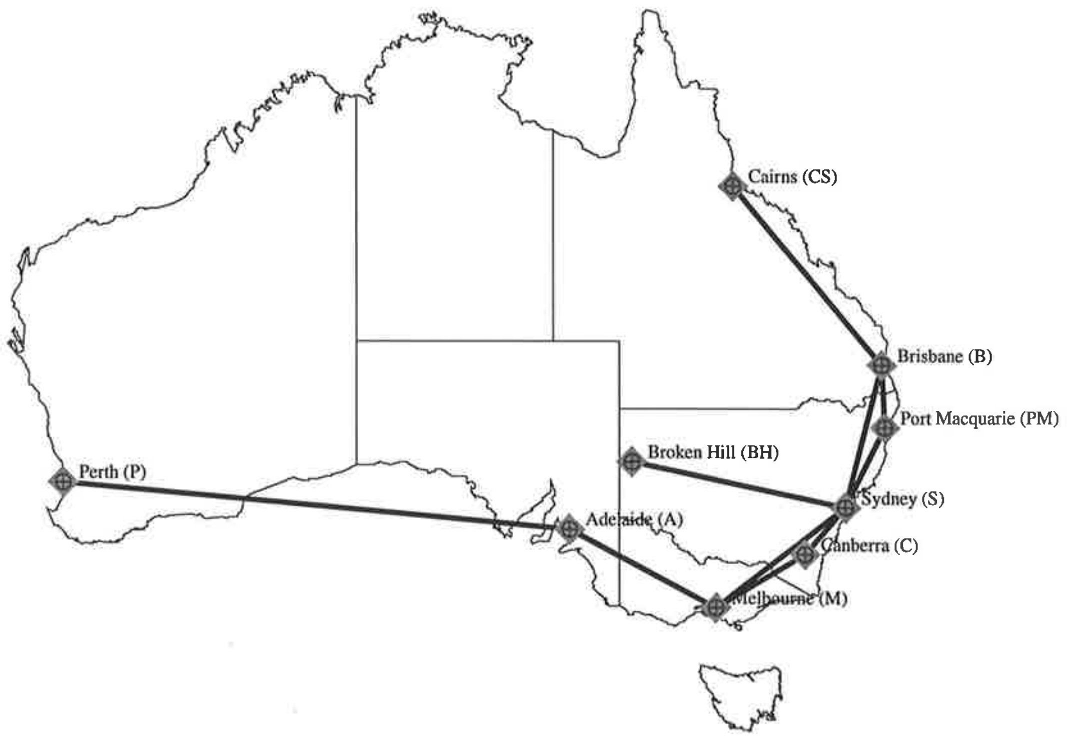


Figure 6.1.1: An example loss network

single physical link (and hence for which no logical link is created) and $S_I$ to be the set of remaining streams which consequently have an indirect physical path as well as a direct logical path. For example, in Figure 6.1.1, Perth-Adelaide is connected by a single physical link and hence belongs to the set $S_P$. Perth-Melbourne is connected by an indirect physical path and hence belongs to the set $S_I$.

For $s \in S$ let $P_s$ be the subset of paths that stream $s$ may use and let $P = \bigcup_s P_s$. The set $P$ can also be partitioned into subsets in another way. Define $P_P$ to be the set of paths consisting of a single physical link, $P_L$ to be the set of logical paths consisting of a single logical link and $P_I$ to be the set of physically indirect paths, that is, paths consisting of more than one physical link. Our routing scheme allows calls for stream $s \in S_I$ to use the physical path with probability $\theta_s$ and the logical path with probability $1 - \theta_s$. The parameter $\theta_s$ is called the splitting probability.

Path $p$ requires $\varsigma_{jp}$ circuits on link $j$. The arrival rate of calls to stream $s$ is a Poisson stream of rate $\nu_s$. A call on stream $s$ requesting path $p \in P_s$ is blocked and lost if there are less than $\varsigma_{jp}$ circuits free on any link $j$. Otherwise, the call is connected and simultaneously holds $\varsigma_{jp}$ circuits from link $j$ for the call holding time. The call holding time for calls on stream $s$ are identically distributed with unit mean and are independent of all earlier arrival times and holding times.

If a call for stream $s$ is connected, it is charged a tariff of $\alpha_s$ per unit of time. Note that regardless of which path $p \in P_s$ a call for stream $s \in S_I$ uses, it must be charged the same tariff since a user does not choose which path their call takes in the network. The arrival rate may depend on the tariff since a higher tariff may lead to fewer calls being made. It may also depend on the grade of service : if the blocking probability is high then the arrival rate might be lower. Therefore, we allow the arrival rate to depend on $\alpha_s$ and $B_s$ and denote it by $\nu_s(\alpha_s, B_s)$, where $B_s$ is the blocking probability of stream $s$, obtained by averaging the blocking probability over all paths $p \in P_s$.

In designing our network we are interested in finding the link capacities $C_j, j \in \mathcal{J}$, the stream tariffs $\alpha_s, s \in S$ and the splitting probabilities $\theta_s, s \in S_I$. Bold face will be used to represent vectors, for instance $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{|S_I|})$, where $|S_I|$ denotes the number of streams $s \in S_I$.

An exact analysis for such a network exists (see Bean and Taylor [10], Kelly [44]). However, although the equilibrium distribution has a very simple form, it has been shown by Louth *et al.* [48] that determination of the normalising constant is #P-complete in the number of distinct routes. This makes the analysis impractical for realistically sized networks. Hence we shall use the well-known Erlang fixed point technique (see Kelly [44]) for approximating the blocking probabilities.

Let $E_j, j \in \mathcal{J}$ be the unique solution to the equations

$$E_j = E(\rho_j, C_j) \;, \tag{6.2.1}$$

where

$$\rho_j = \sum_{s \in S} \sum_{p \in P_s : j \in p} \varsigma_{jp} f_s^p \nu_s(\alpha_s, B_s)(1 - E_j)^{-1} \prod_{i \in p} (1 - E_i)^{\varsigma_{ip}} , \tag{6.2.2}$$

the probabilities $f_s^p$ are given by

$$f_s^p = \begin{cases} 1 , & p \in P_P \cap P_s , \\ 1 - \theta_s , & p \in P_L \cap P_s , \\ \theta_s , & p \in P_I \cap P_s , \end{cases} \tag{6.2.3}$$

and the function $E$ is Erlang's formula

$$E(\rho, C) = \frac{\rho^C}{C!} \left[ \sum_{l=0}^{C} \frac{\rho^l}{l!} \right]^{-1} , \quad j \in \mathcal{J}. \tag{6.2.4}$$

Then the vector $(E_j, j \in \mathcal{J})$ is called the *Erlang fixed point* and an approximation for the acceptance probability on path $p$, $a_p$, is given by,

$$a_p = 1 - b_p = \prod_{j \in \mathcal{J}} (1 - E_j)^{\varsigma_{jp}} \tag{6.2.5}$$

where $b_p$ is the blocking probability on path $p$.

The average blocking probability for a stream $s$ is given by

$$B_s = \sum_{p \in P_s} f_s^p b_p . \tag{6.2.6}$$

The average acceptance probability of calls into the network for stream $s$ is

$$A_s = 1 - B_s = \sum_{p \in P_s} f_s^p a_p, \tag{6.2.7}$$

and is used to describe the grade of service. This then enables $\nu_s(\alpha_s, B_s)$ to be written as $\nu_s(\alpha_s, \boldsymbol{E})$ since equations (6.2.5) and (6.2.6) show that $B_s$ is a function of the link blocking probabilities $\boldsymbol{E}$ giving

$$B_s = 1 - \sum_{p \in P_s} f_s^p \prod_{j \in \mathcal{J}} (1 - E_j)^{\varsigma_{jp}}. \tag{6.2.8}$$

This approximation assumes that all requests for circuits are independent from link to link as are requests for multiple circuits within a single link and for all streams that have paths that use that link. Thus the traffic offered to link $j$ by stream $s$ using path $p$ is Poisson with rate $f_s^p \nu_s(\alpha_s, B_s)$ but will be thinned at every link $i$ on the path by a factor of $(1-E_i)$ for the number of circuits $\varsigma_{ip}$ required, except link $j$, before being offered to link $j$. Kelly's [44] reasoning for this is that the level of carried traffic on link $j$ will be $\sum_{s\in S}\sum_{p\in P_s:j\in p}\varsigma_{jp} f_s^p \nu_s(\alpha_s, B_s)\prod_{i\in p}(1-E_i)^{\varsigma_{ip}}$. The blocking probability on link $j$ given by equation (6.2.1) should then be consistent with the level of carried traffic given by equation (6.2.2), under the Erlang model of a single link offered Poisson single-circuit traffic. Ziedins and Kelly [66] have shown that this approximation is more accurate the more diverse the paths that pass through the link.

## 6.3 Formulation

We noted earlier that we wish to maximise network profit, which is the difference between revenue received from users and the cost of providing the network.

The rate at which calls for stream $s \in S$ are accepted into the network, when the tariff charged is $\alpha_s$, is given by $\nu_s(\alpha_s, \boldsymbol{E})(1-B_s)$. Thus the network provider can expect to receive revenue per unit time for stream $s$ of $\alpha_s \nu_s(\alpha_s, \boldsymbol{E})(1-B_s)$, and the total expected revenue per unit time is

$$\sum_{s\in S} \alpha_s \nu_s(\alpha_s, \boldsymbol{E})(1-B_s) = \sum_{s\in S}\sum_{p\in P_s} f_s^p \alpha_s \nu_s(\alpha_s, \boldsymbol{E}) \prod_{j\in p}(1-E_j)^{\varsigma_{jp}}. \tag{6.3.9}$$

The cost of installing the network consists of a fixed cost, involving such factors as the digging of trenches and laying of cables, and a variable cost, which depends on the capacity. It is possible to ignore the fixed cost in the problem formulation as it can be incorporated after the optimisation procedure is complete.

Define the cost per circuit of cross-connecting to be $x$ and the cost per circuit of multiplexing/de-multiplexing to be $m$. If these costs are capital costs, then they must be ~~be~~ written off over a period of time and so can be described as a charge per unit time. Also, if capacity is leased, these costs are implicitly a charge per unit time. Calls on a physical path are multiplexed at the origin node, de-multiplexed at the destination node and both multiplexed and de-multiplexed at every intermediate node. Assuming that the cost of multiplexing is equivalent to that of de-multiplexing, the number of multiplexes/de-multiplexes on a physical path is twice the number of links on the path. Hence, the cost of providing all the physical paths is $2m\sum_{i\in\mathcal{J}_P} C_i$.

Consider a logical path which consists of the logical link $i$. The number of intermediate nodes through which calls must be cross-connected is given by $\eta_i - 1$, where $\eta_i$ is the number of physical links on which logical link $i$ uses capacity. Calls on a logical path must also be multiplexed at the origin node and de-multiplexed at the destination node. Therefore, the total cost of all the logical paths is $\sum_{i\in\mathcal{J}_L}(2m+(\eta_i-1)x)C_i$.
The total cost of providing the network is therefore

$$2m\sum_{i\in\mathcal{J}_P} C_i + \sum_{i\in\mathcal{J}_L}(2m+(\eta_i-1)x)C_i = 2m\sum_{i\in\mathcal{J}} C_i + x\sum_{i\in\mathcal{J}_L}(\eta_i-1)C_i. \qquad (6.3.10)$$

The network provider wishes to maximise profit and thus wishes to maximise

$$\sum_{s\in S}\alpha_s\nu_s(\alpha_s,\boldsymbol{E})(1-B_s) - 2m\sum_{i\in\mathcal{J}} C_i - x\sum_{i\in\mathcal{J}_L}(\eta_i-1)C_i. \qquad (6.3.11)$$

We shall assume that there are regulatory constraints. That is, for all $s\in S$ the tariff $\alpha_s$ must lie in the interval $[\underline{\alpha}_s,\overline{\alpha}_s]$ and the average stream blocking probability $B_s$ defined in equation (6.2.6) must lie within $[\underline{B}_s,\overline{B}_s]$. Obviously, the link blocking $E_j,\ j\in\mathcal{J}$ and the splitting probability $\theta_s,\ s\in S_I$ must lie in the interval $[0,1]$.

Before presenting our optimisation formulation it is important to note a further feature used by Bean and Taylor [10] which we shall apply. By using link blocking probabilities, rather than link capacities, as variables in our formulation a large efficiency gain is achieved. There are two reasons : first, the network analysis performed by the Erlang fixed point approximation is no longer iterative in nature, and second, it is immediately obvious whether a proposed set of link blocking probabilities is feasible, that is, whether they lie in $[\underline{B_s},\overline{B_s}]$. In contrast it requires a full and iterative network analysis to check whether a given set of capacities lead to feasible blocking probabilities.

In order to make use of this, for fixed $E_j$ we require the inverse function $C : R_+ \to R_+$, such that

$$C_j = C(\rho_j, E_j),\ j \in \mathcal{J}. \tag{6.3.12}$$

$C_j$ is then the capacity required in an Erlang loss system to carry an offered traffic of $\rho_j$ with a blocking probability of $E_j$. We will discuss the method and formulae for calculation of the function $C$ in Section 6.4.

Using the function $C$, the formulation of the non-linear program is as follows.

**Variables**

$$\alpha_s\,, \quad s \in S\,, \tag{6.3.13}$$

$$E_j\,, \quad j \in \mathcal{J}\,, \tag{6.3.14}$$

$$\theta_s\,, \quad s \in S_I\,. \tag{6.3.15}$$

**Objective**

**max :**

$$Z = \sum_{s\in S}\sum_{p\in P_s} f_s^p \alpha_s \nu_s(\alpha_s, \boldsymbol{E}) \prod_{j\in p}(1-E_j)^{\varsigma_{jp}} - 2m\sum_{i\in\mathcal{J}} C_i - x\sum_{i\in\mathcal{J}_L}(\eta_i - 1)C_i\,. \tag{6.3.16}$$

**Constraints**

$$0 \le E_j \le 1\,, \quad j \in \mathcal{J}\,, \tag{6.3.17}$$

$$\underline{\alpha_s} \le \alpha_s \le \overline{\alpha_s}\,, \quad s \in S\,, \tag{6.3.18}$$

$$0 \le \theta_s \le 1\,, \quad s \in S_I\,, \tag{6.3.19}$$

$$\underline{B_s} \le B_s \le \overline{B_s}\,, \quad s \in S\,, \tag{6.3.20}$$

where $\rho_j$, $f_s^p$, $B_s$ and $C_j$ are given by equations (6.2.2), (6.2.3), (6.2.8), and (6.3.12) respectively.

## 6.4 Numerical discussion of $C(\rho, E)$

In Section 6.3 we noted the advantages of using link blocking probabilities, rather than link capacities, as variables in our formulation. We also noted that in order to make use of this, for fixed $E_j$ we require the inverse function $C : R_+ \to R_+$, such that

$$C_j = C(\rho_j, E_j),\ j \in \mathcal{J}. \tag{6.4.21}$$

We need a method of determining $C_0$, such that for a given $\rho_0$ and $E_0$, $E(\rho_0, C_0) = E_0$.

Atkinson and Anido [7] gave a continuous function for $C(\rho)$, for fixed link blocking, $\boldsymbol{E}$. They created this expression by fitting a function to curves generated by using Kaufmann's model [42]. They also give the first and second derivatives of this function with respect to $\rho$. However, it would be preferable to have a function which is dependent on the link blocking $E$ as well as the offered traffic $\rho$.

Bean and Taylor [10] used recursion on the iterative form of Erlang's function

$$E(\rho, C) = \frac{\rho E(\rho, C-1)}{C + \rho E(\rho, C-1)}, \tag{6.4.22}$$

with the boundary condition $E(\rho, 0) = 1$, until $E(\rho, C)$ is less than the required blocking. They then use linear interpolation to find $C_0 = C(\rho_0, E_0)$.

However, in order to obtain better accuracy we have chosen to follow the method given by Farmer and Kaufmann [23]. This method attempts to find the zero of

$$\phi(C) = E(\rho_0, C) - E_0 \tag{6.4.23}$$

by using the regula falsi method [2], which will be discussed shortly, and the following formulae for $E(\rho, C)$. The specific formulae for $E(\rho, C)$ are chosen due to the fact they are defined for $C \in R_+$ and are differentiable with respect to $C$. For $\rho > 1$, $E(\rho, C)$ is calculated using

$$E = \frac{\exp(C \ln(\frac{\rho}{C+1}) - \rho + C + 1 - \frac{1}{12(C+1)} + \frac{1}{360(C+1)^3})}{\sqrt{2\pi(C+1)Q(\xi_0)}} \tag{6.4.24}$$

where

$$\xi_0 \;=\; \xi - \frac{1}{27C}\left[\frac{2}{3\sqrt{C}}(\xi^2 - 1) - \frac{\xi}{4}(\xi^2 - 3)\right],$$

$$\begin{aligned}
\xi &= \left(\left(\frac{\rho}{C+1}\right)^{1/3} + \frac{1}{9(C+1)} - 1\right) 3\sqrt{C+1}, \\
Q(\xi) &= 1/2[1+\xi(d_1+\xi(d_2+\xi(d_3+\xi(d_4+\xi(d_5+\xi d_6)))))]^{-16} + \epsilon(\xi), \\
\|\epsilon(\xi)\| &< 1.5 \times 10^{-7}, 0 \le \xi \le \infty,
\end{aligned}$$

and

$$d_1 = 0.0498673470, \quad d_2 = 0.0211410061, \quad d_3 = 0.0032776263,$$

$$d_4 = 0.0000380036, \quad d_5 = 0.0000488906, \quad d_6 = 0.0000053830.$$

For $\rho < 1$, $E(\rho, C)$ is calculated using

$$E \approx \frac{1}{\frac{\sqrt{2\pi(C+1)}}{\exp(C \ln \frac{\rho}{C+1} - \rho + C + 1 - \frac{1}{12(C+1)} + \frac{1}{360(C+1)^3})} - \sum_{k=1}^{\infty} \frac{\rho^k}{(C+1)(C+2)\ldots(C+k)}}. \tag{6.4.25}$$

For the regula falsi method it is first necessary to choose two initial points, such that they are close as possible to $C_0$, and $\phi(C_1)$ and $\phi(C_2)$ have opposite signs. When $E_0 > \sqrt{\frac{2}{\pi\rho_0}}$, the initial points are chosen such that

$$C_1 = 1, \quad E_1 = \tfrac{\rho_0}{1+\rho_0}$$

$$\text{and } C_2 = \rho_0, \quad E_2 = \sqrt{\tfrac{2}{\pi\rho_0}}.$$

When $E_0 < \sqrt{\frac{2}{\pi\rho_0}}$, the initial points are chosen such that

$$C_1 \approx \frac{1}{4}\left\{\left[2\sqrt{C_0} + \left(2\ln\frac{4}{(4 - E\sqrt{2\pi C_0})E\sqrt{2\pi C_0}}\right)^{1/2}\right]^2 - 3 - \frac{1}{4E_0 + \frac{C_0}{4} + \frac{1}{9}}\right\}$$

and

$$\begin{aligned}
C_2 &= C_1 + 1, \text{when} E(\rho_0, C_1) \ge E_0 \\
\text{or } C_2 &= C_1 - 1, \text{when} E(\rho_0, C_1) < E_0.
\end{aligned}$$

$E(\rho_0, C_2)$ can then be calculated by means of the recurrence relation

$$E(\rho_0, C_2) = E(\rho_0, C_1 + 1) = \frac{\rho_0 E(\rho_0, C_1)}{C_1 + 1 + \rho_0 E(\rho_0, C_1)} \tag{6.4.26}$$

$$\text{or } E(\rho_0, C_2) = E(\rho_0, C_1 - 1) = \frac{C_1}{\rho_0}\frac{E(\rho_0, C_1)}{1 - E(\rho_0, C_1)}. \tag{6.4.27}$$

$C_3$ is now calculated using

$$C_3 = C_2 - \frac{C_2 - C_1}{\phi(C_2) - \phi(C_1)}\phi(C_2). \tag{6.4.28}$$

The next iteration is continued with $C_3$ and either $C_1$ or $C_2$ for which $\phi(C_1)$ or $\phi(C_2)$ is of opposite sign to $\phi(C_3)$.

Farmer and Kaufman stated that using the initial two points $C_1$ and $C_2$, the first regula falsi iteration (6.4.28) yields a very good approximation to the exact value $C_0$, and in general only one or at most two additional iterations are required. In fact the formula for $C_1$, in the case where $E_0 < \sqrt{\frac{2}{\pi\rho_0}}$, yields results which differ from the exact value $C_0$ by no more than $\frac{1}{3}$.

Figures 6.4.2 and 6.4.3 have been plotted using data from Akimaru and Nishimura [6]. Figure 6.4.2 shows the dimensioning curve of required capacity $C$ for offered traffic (Erlangs) $\rho$, with fixed blocking probability $E = 0.01$. Figure 6.4.3 shows the difference between the required capacity $C$, and the offered traffic (Erlangs) $\rho$, with fixed blocking $E = 0.01$, as $\rho$ varies. We have included Figure 6.4.3 to emphasise that although $C$ looks linear in $\rho$ for $\rho > 10$, say, it is in fact quite non-linear.

A further study of the function $C(\rho, E)$ was carried out in Berezner *et al.* [11]. They showed that $\rho(1 - E) < C(\rho, E) < \rho(1 - E) + 1/E$ and moreover that, for fixed $E$, $\lim_{\rho\to\infty} C(\rho, E) = \rho(1 - E)$.

## 6.5 Optimisation procedure

We now have a formulation with a smooth objective function subject to constraints. The constraints consist of simple bounds on the variables and smooth non-linear constraints, (6.3.20). Often, an advantage to solving such optimisation problems is to have first partial derivatives for the objective function and constraint functions.

As $C_j = C(\rho_j(\boldsymbol{\theta}, \boldsymbol{E}, \boldsymbol{\nu}(\boldsymbol{\alpha}, \boldsymbol{E})), E_j)$, the dependency of the function $C$ on both $\alpha$ and $\theta$ is due to $\rho$. Thus, to find the partial derivative of the objective function Z (6.3.16) with respect to either of the variables $\alpha$ or $\theta$, it is necessary to be able to

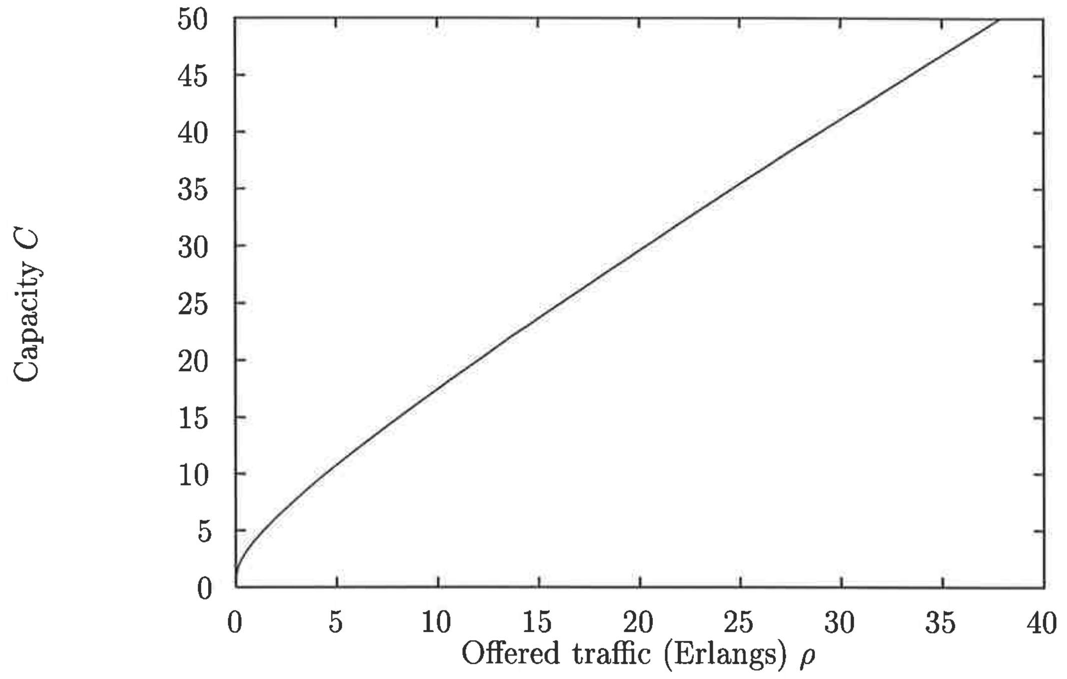


Figure 6.4.2: Dimensioning curve

evaluate $\frac{\partial C}{\partial \rho}$ which can be rewritten as $\frac{\partial E}{\partial \rho}/\frac{\partial E}{\partial C}$.

Bean and Taylor [10] used equations (6.2.4) and (6.4.22) to define a formula for the partial derivative $\frac{\partial E}{\partial C}$, which is not continuous. They also used

$$\frac{\partial E}{\partial \rho} = E\left(\frac{C}{\rho} - 1 + E\right) \tag{6.5.29}$$

which is defined in Jagerman [38] and Akimaru and Nishimura [6]. Use of these two partial derivatives, then gives $\frac{\partial C}{\partial \rho} = \frac{\partial E}{\partial \rho}/\frac{\partial E}{\partial C} = 1 - E$. This is no longer a function of $\rho$, and will prove to be a useful approximation in Chapter 5.

To obtain a formula for $\frac{\partial C}{\partial \rho}$ which is dependent on $\boldsymbol{E}$ and $\rho$, we can use equation (6.5.29), but it is necessary to choose a different formula for $\frac{\partial E}{\partial C}$. By differentiating equation (6.4.24), with respect to $C$, it is possible to obtain a function for $\frac{\partial E}{\partial C}$ with

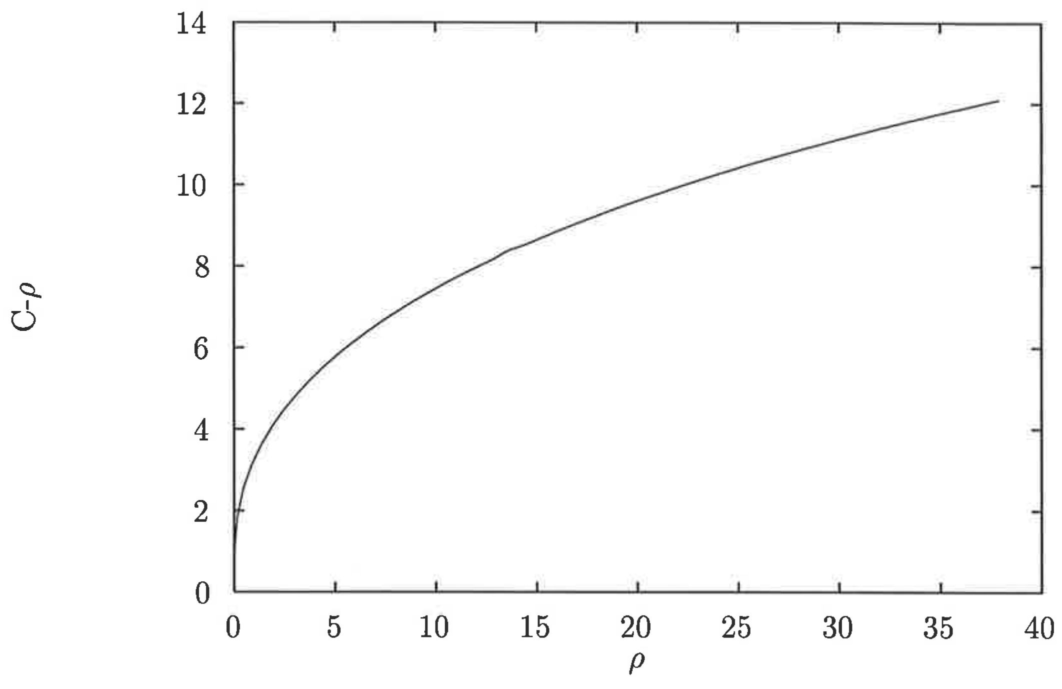


Figure 6.4.3: Dimensioning curve

the aid of Maple. As this function is complicated, it is presented in Appendix B. These two formulae are then used to obtain $\frac{\partial C}{\partial \rho} = \frac{\partial E}{\partial \rho} / \frac{\partial E}{\partial C}$, which is a function of $\rho$.

Figure 6.5.4 shows the marginal capacity increase required for an increase in offered traffic when the link blocking is fixed at $E = 0.01$. This figure was created using data from Akimaru and Nishimura [6], who only gave data for $\rho < 40$. This is a reasonably small arrival rate for a realistic network. As the offered traffic increases, the marginal capacity increase required approaches $1 - E$. However, it can be seen that, for small $\rho$, this approximation is inaccurate. This graph also shows the economy of scale factor : as the offered traffic increases, the marginal capacity increase required decreases.

The derivatives of the non-linear constraint (6.3.20), for $s \in S$ are

$$\frac{\partial B_s}{\partial E_j} = \begin{cases} \theta_s \varsigma_{jp(s)}(1-E_j)^{-1}\prod_{i\in p(s)}(1-E_i)^{\varsigma_{ip(s)}}, & j \in p(s), \\ (1-\theta_s)\varsigma_{jl(s)}(1-E_j)^{\varsigma_{jl(s)}-1}, & j \in l(s), \\ 0, & \text{otherwise,} \end{cases} \tag{6.5.30}$$

$$\frac{\partial B_s}{\partial \alpha_{s'}} = 0, \qquad \forall s' \in S, \tag{6.5.31}$$

$$\frac{\partial B_s}{\partial \theta_{s'}} = \begin{cases} (1-E_{\ell(s)})^{\varsigma_{\ell(s)l(s)}} - \prod_{j\in p(s)}(1-E_j)^{\varsigma_{jp(s)}}, & s = s', \\ 0, & \text{otherwise.} \end{cases} \tag{6.5.32}$$

Thus we now have a formulation with a smooth objective function subject to simple bounds on the variables and smooth non-linear constraints. It is also now possible to obtain formulae for the first partial derivatives of the objective function Z (6.3.16) and the constraints. Keeping these facts in mind, we selected NAG routine E04UCF as our optimisation package. The specification for this routine is given below.

## 6.5.1 NAG routine

This NAG routine is essentially identical to the subroutine SOL/NPSOL described in Gill *et al.* [28]. The NAG documentation [52] described E04UCF in the following way.

"//E04UCF// is designed to minimize an arbitrary smooth function subject to constraints, which may include simple bounds on the variables, linear constraints and smooth nonlinear constraints. (//E04UCF// may be used for unconstrained, bound-constrained and linearly constrained optimization.) The user must provide subroutines that define the objective and constraint functions and as many of their first partial derivatives as possible. Unspecified derivatives are approximated by finite differences. //E04UCF// uses a sequential quadratic programming (SQP) algorithm in which the search direction is the solution of a quadratic programming

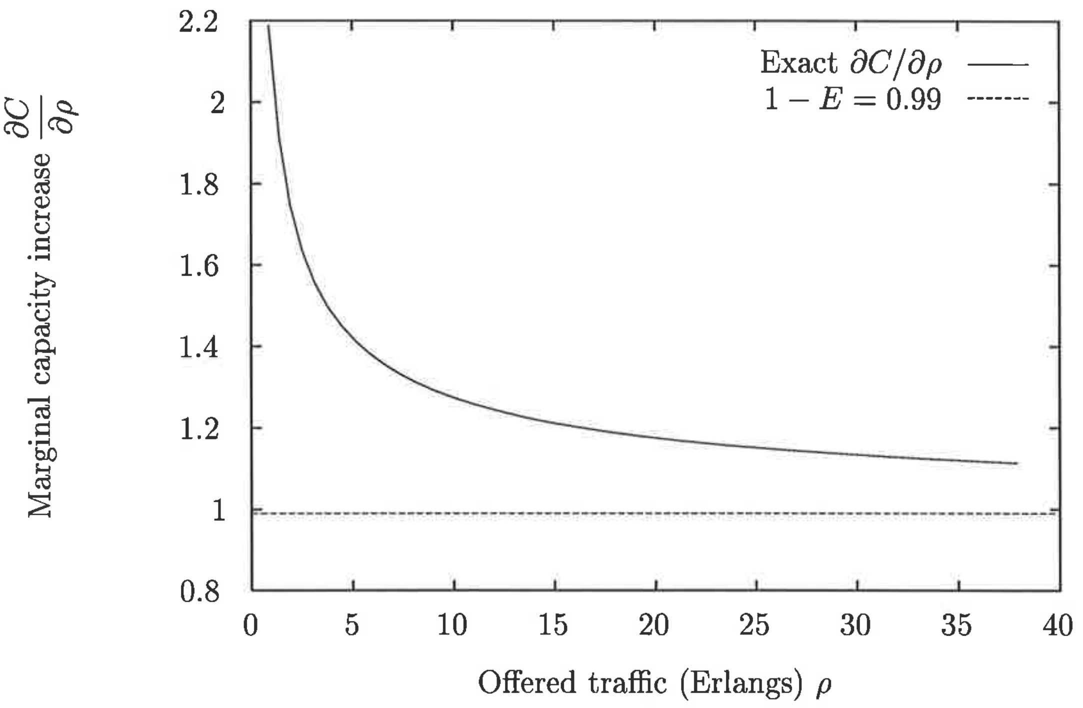


Figure 6.5.4: Marginal capacity increase for an increase in offered traffic

(QP) problem. The algorithm treats bounds, linear constraints and nonlinear constraints separately. "

## 6.6 Example

We will now work through an example, using an artificial network and setting the costs, cost function and constraint bounds artificially also, but such that they satisfy any necessary requirements.

Consider the network shown in Figure 6.1.1. There are 9 nodes and 36 streams, of which 10 have only the direct physical path available and 26 have a choice between physical and logical paths. That is, there are 10 streams in $S_P$, and 26 streams in $S_I$. The logical path consists of the single logical direct link, and the physical path is the unique shortest possible path in terms of the number of links used.

We have set the multiplexing cost coefficient to be $m = 1.01$ and the cross-connecting cost coefficient to be $x = 2.00$. These values have been chosen to encourage calls on some streams to use their physical paths and on others their logical paths. Since the multiplexing coefficient always occurs in multiples of two, the two costs are approximately equal making neither path highly desirable, while still reflecting the fact that multiplexing is more expensive than cross-connecting.

The average stream blocking probability for all streams is bounded above by 0.01, with no lower bound imposed, giving $0 \leq B_s \leq 0.01, s \in S$. The tariff has not been bounded above or below, hence $0 \leq \alpha_s \leq \infty$. The number of circuits required by a stream, $s \in S$, using path $p \in P_s$ on link $j \in \mathcal{J}$ has been set to be zero or one; that is $\varsigma_{jp} = 0/1$. This implies that each path $p$ requires either zero circuits or one circuit on link $j$ and so provides the physical description of the path $p$.

The function $\nu_s(\alpha_s, B_s)$ has been defined such that $\alpha_s \nu_s(\alpha_s, B_s)$ decays rapidly as $\alpha_s \to \infty$ for all $s \in S$, to avoid the network being highly profitable whilst accepting a very small number of calls, and is bounded above as $\alpha_s \to 0$ for all $s \in S$, to avoid an infinite number of calls being accepted. "Economic models for such a function use a sigmoid shape, representing the fact that under competition there will usually be a marked reduction (increase) in demand if your tariff is more (less) expensive that your competitor's" [10]. The function we use is

$$\nu_s(\alpha_s, B_s) = \begin{cases} \tilde{\nu_s}\exp\frac{1}{3}(\tilde{\alpha_s} - \alpha_s), & \text{if } \frac{1}{3}\alpha_s \geq \tilde{\alpha_s}, \\ \tilde{\nu_s}(2 - \exp\frac{1}{3}(\alpha_s - \tilde{\alpha_s})), & \text{otherwise} \end{cases} \tag{6.6.33}$$

which is similar to that used by Bean and Taylor [10]. This function is continuous and differentiable with respect to $\alpha_s \in (0, \infty)$, and is bounded above by $\tilde{\nu_s}(2 - \exp(-\tilde{\alpha_s}/3))$ when $\alpha_s = 0$.

We have defined the tariff of the competitor $\tilde{\alpha_s}$, $s \in S$, to be 3. The point of inflexion will then be at $\alpha_s = 3$. We define the values of the base traffic demand, $\tilde{\nu_s}$, which is the traffic demand if the tariff is $\tilde{\alpha_s}$, as in Bean and Taylor. These demands are based on approximate subscriber figures and are given in Table 6.6.1.

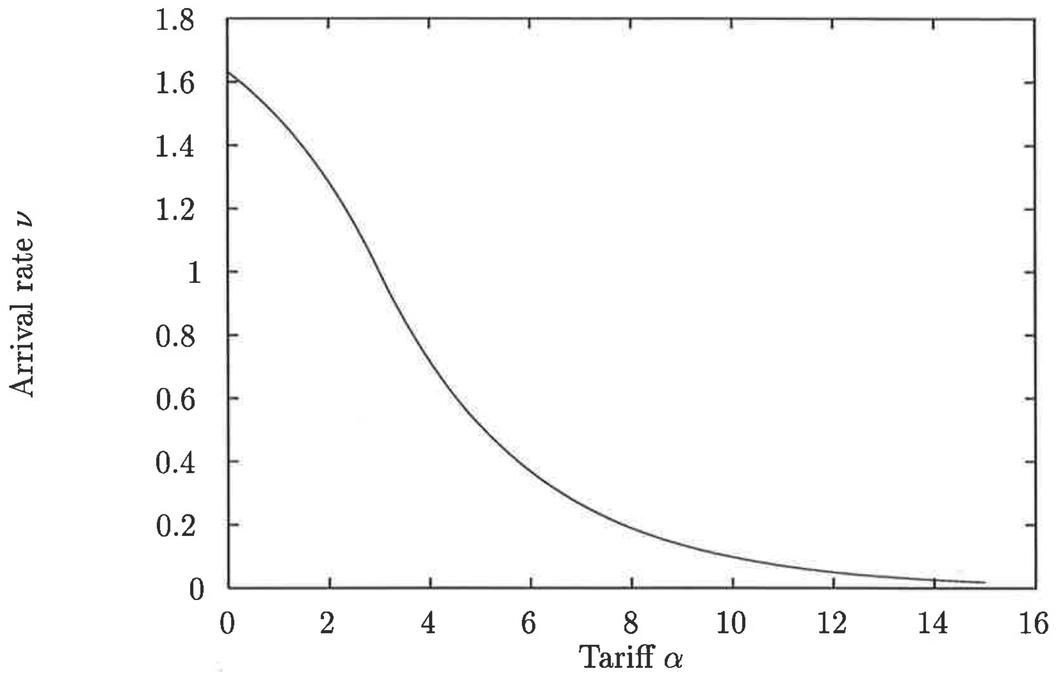


Figure 6.6.5: Elasticity function $\nu_s(\alpha_s)$ for $\tilde{\alpha_s} = 3, \tilde{\nu_s} = 1$

## 6.7 Results

To find the optimal tariffs, splitting probabilities and link blocking probabilities we have used the standard routine available in the NAG Library of routines discussed earlier. The values of the tariffs and the splitting probabilities are given in Table 6.7.2. The full names of the cities abbreviated in Table 6.7.2 can be seen in Figure 6.1.1. The splitting probability in the table is the probability that the *physical* path is used. The streams, $s \in S_I$, have been ordered from those with the lowest accepted traffic (CS-C) to those with the highest (B-M).

An inspection of Table 6.7.2 reveals that the optimal tariffs are close to integers. In the simple model, in which the logical path is not offered as an alternative, Bean

| Streams $s \in S_I$ | Base Traffic | Opt. Traffic | Streams $s \in S_I$ | Base Traffic | Opt. Traffic | Streams $s \in S_P$ | Base Traffic | Opt. Traffic |
|---|---|---|---|---|---|---|---|---|
| CS-C | 475.0 | 63 | A-BH | 8603.4 | 1135 | C-S | 1744.0 | 882 |
| C-P | 660.0 | 86 | CS-PM | 5840.9 | 1508 | C-M | 2622.4 | 1326 |
| C-PM | 562.0 | 144 | A-B | 12556.8 | 1656 | B-CS | 8967.0 | 4555 |
| BH-C | 580.7 | 149 | BH-PM | 7138.0 | 1848 | A-P | 9776.6 | 4966 |
| A-C | 677.7 | 174 | B-BH | 10764.6 | 2788 | B-PM | 10419.7 | 5296 |
| B-C | 848.2 | 217 | P-S | 25086.7 | 3344 | PM-S | 21378.1 | 10877 |
| CS-P | 6858.0 | 232 | CS-M | 27147.1 | 3620 | BH-S | 22084.1 | 11238 |
| A-CS | 7040.3 | 471 | CS-S | 18081.2 | 4712 | B-S | 32191.0 | 16383 |
| P-PM | 8111.9 | 543 | A-S | 25750.9 | 6717 | A-M | 38632.8 | 19663 |
| BH-P | 8380.6 | 561 | M-PM | 32086.4 | 8371 | M-S | 98224.4 | 50024 |
| BH-CS | 6034.6 | 793 | BH-M | 33143.9 | 8648 | | | |
| B-P | 12232.0 | 819 | M-P | 37638.5 | 9823 | | | |
| A-PM | 8327.5 | 1098 | B-M | 48263.7 | 12600 | | | |

Table 6.6.1: Table of base traffic demands and optimal traffic demands.

and Taylor [10] showed that these optimal tariffs consist of a term representing the cost of carrying a call and a term depending only on the elasticity function of the traffic $\nu_s(\alpha_s, \boldsymbol{E})$. It appears that this is still the case, with the addition of logical links.

For instance, from Table 6.7.2, it can be seen that traffic for stream Cairns to Canberra (CS-C) uses its physical path ($\theta_{CS-C} = 1$). Hence, traffic for this stream is multiplexed at Cairns, de-multiplexed and then re-multiplexed at Brisbane and Sydney (as these are not the traffic's destination) and is finally de-multiplexed at Canberra. Given that the cost of multiplexing is assumed to be equivalent to that of de-multiplexing, we have the cost of multiplexing/de-multiplexing traffic on this path as $6m = 6.06$. By guessing that the form of the term depending on the elasticity function will be as in Bean and Taylor [10], this term is equal to $-\nu_s \left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} = 3$. Hence traffic for the stream CS-C using its physical path, would have a cost of 9.06.

| Streams $s \in S_I$ | Tariff $\alpha_s$ | Split Prob $\theta_s$ | Streams $s \in S_I$ | Tariff $\alpha_s$ | Split Prob $\theta_s$ | Streams $s \in S_P$ | Tariff $\alpha_s$ |
|---|---|---|---|---|---|---|---|
| CS-C | 9.05 | 1.00 | A-BH | 9.07 | 1.00 | C-S | 5.04 |
| C-P | 9.10 | 1.00 | CS-PM | 7.06 | 1.00 | C-M | 5.04 |
| C-PM | 7.07 | 1.00 | A-B | 9.08 | 1.00 | B-CS | 5.03 |
| BH-C | 7.07 | 1.00 | BH-PM | 7.05 | 1.00 | A-P | 5.03 |
| A-C | 7.07 | 1.00 | B-BH | 7.05 | 1.00 | B-PM | 5.03 |
| B-C | 7.08 | 1.00 | P-S | 9.05 | 0.00 | PM-S | 5.03 |
| CS-P | 13.15 | 1.00 | CS-M | 9.04 | 0.00 | BH-S | 5.03 |
| A-CS | 11.11 | 1.00 | CS-S | 7.03 | 0.00 | B-S | 5.03 |
| P-PM | 11.11 | 1.00 | A-S | 7.03 | 0.00 | A-M | 5.03 |
| BH-P | 11.11 | 1.00 | M-PM | 7.03 | 0.00 | M-S | 5.03 |
| BH-CS | 9.09 | 1.00 | BH-M | 7.03 | 0.00 | | |
| B-P | 11.11 | 1.00 | M-P | 7.03 | 0.00 | | |
| A-PM | 9.08 | 1.00 | B-M | 7.03 | 0.00 | | |

Table 6.7.2: Table of optimal tariffs and splitting probabilities.

This is extremely close to the value of 9.05 given by the numerical results found by NAG in Table 6.7.2.

It can also be seen from Table 6.7.2 that traffic for stream Perth to Sydney (P-S) uses its logical path ($\theta_{P-S} = 0$). Hence, traffic for this stream is multiplexed at Perth, cross-connected at Adelaide and then at Melbourne and is finally de-multiplexed at the destination, Sydney. Thus the cost of multiplexing and cross-connecting traffic on this stream is $2m + 2x = 2.02 + 4 = 6.02$. Adding the cost due to the elasticity function gives the cost of traffic for stream P-S using its logical path as 9.02, which is close to the value of 9.05 given by the numerical results found by NAG in Table 6.7.2.

This seems to suggest that there is a simple formula which the tariffs $\alpha_s, s \in S$ obey. It can also be seen in Table 6.7.2 that the splitting probabilities $\theta_s, s \in S_I$ all have value zero or one, indicating that all the traffic for the streams $s \in S_I$ uses either the logical or physical path. We shall investigate these conjectures in the next chapter.

# Chapter 7

# Analytical results

## 7.1 Exact optimisation

We now wish to use analytical methods to find either explicit formulae for the tariffs $\alpha_s^*, s \in S$ and the splitting probabilities $\theta_s^*, s \in S_I$ or equations which they must satisfy. By doing this we hope to be able to investigate whether the same properties which arose from the numerical results in the previous chapter hold in general.

As we know that our objective function Z, (6.3.16), is differentiable, we will use simple differential calculus in an attempt to find any stationary points for $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ and classify them. However, it is necessary to discuss some notation and concepts that will be used in the analysis first.

## 7.2 Further notation

In the following work, it is necessary to know the functional dependencies of the variables. For notational convenience we will suppress these dependencies but list them here for future reference.

$$\nu_s(\alpha_s, \boldsymbol{E}),\ C(\rho_j, E_j),\ \rho_j(\boldsymbol{\theta}, \boldsymbol{E}, \boldsymbol{\nu}),\ a_{p(s)}(\boldsymbol{E}),\ a_{l(s)}(\boldsymbol{E}),\ A_s(\boldsymbol{E}, \theta_s),\ f_s^p(\theta_s). \quad (7.2.1)$$

For stream $s \in S_P$, there is a unique direct physical path, say $d(s)$, which consists of a single link $\delta(s)$. For stream $s \in S_I$ there is a unique physical path, say $p(s)$, and a unique logical path, say $l(s)$, which consists of a single logical link $\ell(s)$. Although this notation may appear superfluous, it will turn out to be necessary to avoid unnecessary summations over a single link.

Hence, using equation (6.2.5), for a stream $s \in S_P$, the acceptance probability for the unique path $d(s)$ is given by $a_{d(s)} = (1 - E_{\delta(s)})^{\varsigma_{\delta(s)d(s)}}$. For stream $s \in S_I$, the acceptance probabilities on the logical and physical paths are given by $a_{l(s)} = (1 - E_{\ell(s)})^{\varsigma_{\ell(s)l(s)}}$ and $a_{p(s)} = \prod_{j\in p(s)}(1 - E_j)^{\varsigma_{jp(s)}}$ respectively.

The expressions below will arise frequently during the work to follow, and thus it will be beneficial to have defined them and to have an understanding of their physical interpretation. For $E_j < 1$, let us define

$$M_p(\tfrac{\partial \boldsymbol{C}}{\partial \boldsymbol{\rho}}, \boldsymbol{E}) = \sum_{j\in\mathcal{J}\in p} 2m\varsigma_{jp}\frac{\partial C}{\partial \rho_j}(1-E_j)^{-1} + \sum_{j\in\mathcal{J}_L\in p} x(\eta_j - 1)\varsigma_{jp}\frac{\partial C}{\partial \rho_j}(1-E_j)^{-1}. \quad (7.2.2)$$

Hence,

$$\begin{aligned}
M_{\delta(s)}(\tfrac{\partial C}{\partial \rho_{\delta(s)}}, E_{\delta(s)}) &= 2m\varsigma_{d(s)\delta(s)}\frac{\partial C}{\partial \rho_{\delta(s)}}(1 - E_{\delta(s)})^{-1}, \\
M_{l(s)}(\tfrac{\partial C}{\partial \rho_{\ell(s)}}, E_{\ell(s)}) &= (2m + x(\eta_{\ell(s)} - 1))\varsigma_{\ell(s)l(s)}\frac{\partial C}{\partial \rho_{\ell(s)}}(1 - E_{\ell(s)})^{-1}, \\
M_{p(s)}(\tfrac{\partial \boldsymbol{C}}{\partial \boldsymbol{\rho}}, \boldsymbol{E}) &= \sum_{j\in p(s)} 2m\varsigma_{jp(s)}\frac{\partial C}{\partial \rho_j}(1 - E_j)^{-1}.
\end{aligned}$$

We will write $M_{\delta(s)}$, $M_{l(s)}$ and $M_{p(s)}$ in future.

The quantity $\frac{\partial C}{\partial \rho_j}$ is the marginal extra capacity required to support a unit of extra traffic on link $j$ and $M_p$ is the cost of this marginal extra capacity on all the links used by path $p$. If traffic behaved like a fluid flow, then that marginal extra capacity would be $(1 - E_j)$. Therefore, $\frac{\partial C}{\partial \rho_j}(1 - E_j)^{-1}$ can be thought of as the penalty to be paid for the stochastic nature of the link. Berezner *et al.* [11] showed that this ratio is bounded below by 1.

Berezner *et al.* also showed that the marginal extra capacity required to support extra traffic on the link is given by $(1 - E_j)$ in the limit as $\rho_j$ tends to $\infty$. Therefore, this ratio can also be thought of as the penalty that has to be paid due to the finiteness of the traffic on the link. Under this interpretation, the ratio plays the role of a measure of the inefficiency of the link.

### 7.2.1 Optimal tariffs

We will assume that $\nu_s(\alpha_s, \boldsymbol{E})$ is a *strictly* decreasing function of $\alpha_s$ and hence $\frac{\partial \nu_s}{\partial \alpha_s} < 0$. Only a minor technical adjustment is needed if this assumption is not made. In general, demand functions tend to zero as $\alpha \to \infty$ but do not equal zero at any finite value, giving $\nu_s > 0$. It is further possible to argue that we can assume that $\nu_s \neq 0$ since if there is no traffic for a stream, that stream can be ignored.

To find the optimal tariff $\alpha_s^*$ which maximises network profit we will assume we have fixed, feasible link blocking probabilities $\boldsymbol{E}$ and splitting probabilities $\boldsymbol{\theta}$. The objective function (6.3.16) is then differentiated with respect to $\alpha_s$. For $s \in S_P$ this gives

$$\frac{\partial Z}{\partial \alpha_s} = \nu_s(1 - E_{\delta(s)})^{\varsigma_{\delta(s)d(s)}} + \alpha_s \frac{\partial \nu_s}{\partial \alpha_s}(1 - E_{\delta(s)})^{\varsigma_{\delta(s)d(s)}} - 2m \frac{\partial C}{\partial \rho_{\delta(s)}} \frac{\partial \rho_{\delta(s)}}{\partial \nu_s} \frac{\partial \nu_s}{\partial \alpha_s}$$

where

$$\frac{\partial \rho_{\delta(s)}}{\partial \nu_s} = \varsigma_{\delta(s)d(s)}(1 - E_{\delta(s)})^{\varsigma_{\delta(s)d(s)}-1}.$$

Any stream $s \in S_P$ has only one possible path which consists of the single link $\delta(s)$. If $E_{\delta(s)} = 1$, no traffic will be accepted on the link. This implies that no traffic will be accepted for stream $s$, and hence its tariff $\alpha_s$ is inconsequential.

Using our earlier arguments in this section, we can assume that $\frac{\partial \nu_s}{\partial \alpha_s} \neq 0$ and from above, we can assume $E_{\delta(s)} < 1$. To find the stationary point $\alpha_s$, $\frac{\partial Z}{\partial \alpha_s}$ is set to zero. Hence the optimal tariff $\alpha_s^*$, $s \in S_P$, must satisfy

$$\alpha_s = -\nu_s \left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + M_{\delta(s)}. \tag{7.2.3}$$

$M_{\delta(s)}$ is given by equation (7.2.2), and, as discussed in Section 7.2, is the cost of marginal extra capacity required to support extra traffic on path $\delta(s)$. The term $-\nu_s(\frac{\partial \nu_s}{\partial \alpha_s})^{-1}$ depends only on the elasticity function of the traffic $\nu_s$.

For $s \in S_I$,

$$\begin{aligned}\frac{\partial Z}{\partial \alpha_s} &= \sum_{p\in P_s} f_s^p \nu_s \prod_{j\in p}(1-E_j)^{\varsigma_{jp}} + \sum_{p\in P_s} f_s^p \alpha_s \frac{\partial \nu_s}{\partial \alpha_s}\prod_{j\in p}(1-E_j)^{\varsigma_{jp}} \\ &\quad -2m\sum_{j\in p(s)}\frac{\partial C}{\partial \rho_j}\frac{\partial \rho_j}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s} - 2m\frac{\partial C}{\partial \rho_{\ell(s)}}\frac{\partial \rho_{\ell(s)}}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s} - x(\eta_{\ell(s)}-1)\frac{\partial C}{\partial \rho_{\ell(s)}}\frac{\partial \rho_{\ell(s)}}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s}\end{aligned}$$

where

$$\frac{\partial \rho_j}{\partial \nu_s} = \sum_{p\in P_s : j\in p} \varsigma_{jp} f_s^p (1-E_j)^{-1}\prod_{i\in p}(1-E_i)^{\varsigma_{ip}} \tag{7.2.4}$$

giving

$$\frac{\partial \rho_j}{\partial \nu_s} = \begin{cases} \varsigma_{jp(s)}\theta_s(1-E_j)^{-1}\prod_{i\in p(s)}(1-E_i)^{\varsigma_{ip(s)}}, & j\in p(s),\ \ s\in S_I, \\ \varsigma_{jl(s)}(1-\theta_s)(1-E_j)^{\varsigma_{jl(s)}-1} & j\in l(s),\ \ s\in S_I, \\ 0 & \text{otherwise.}\end{cases} \tag{7.2.5}$$

Hence, setting $\frac{\partial Z}{\partial \alpha_s} = 0$ and rearranging gives

$$\alpha_s \frac{\partial \nu_s}{\partial \alpha_s} A_s = -\nu_s A_s + 2m\sum_{j\in p(s)}\frac{\partial C}{\partial \rho_j}\frac{\partial \rho_j}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s} + (2m + x(\eta_{\ell(s)}-1))\frac{\partial C}{\partial \rho_{\ell(s)}}\frac{\partial \rho_{\ell(s)}}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s}.$$

As $\boldsymbol{\theta}$ and $\boldsymbol{E}$ have already been set such that they are feasible solutions, there are only two cases in which $A_s$ could have the value zero. The first possibility is that there is at least one link in both the physical and the logical paths for which the link blocking probability is one, and hence $a_{p(s)} = a_{l(s)} = 0$. The second is that only one of the two paths has at least one link for which the link blocking probability is one, but that that same path carries all the traffic. In both cases the stream acceptance probability will be zero. If this is the case, there will no need to determine the optimal tariff $\alpha_s$ for that stream.

Thus, using this assumption and our earlier assumption that $\frac{\partial \nu_s}{\partial \alpha_s} \neq 0$, the optimal tariff $\alpha_s^*$, $s \in S_I$, must satisfy

$$\alpha_s = -\nu_s\left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + M_{p(s)}\frac{\theta_s a_{p(s)}}{A_s} + M_{l(s)}\frac{(1-\theta_s)a_{l(s)}}{A_s}. \tag{7.2.6}$$

The first term again depends only on the elasticity function of the traffic $\nu_s$. $M_{p(s)}$ and $M_{l(s)}$ are given by equation (7.2.2) and, as discussed in Section 7.2, represent the cost of marginal extra capacity required to support extra traffic on the physical and logical paths respectively. Thus, the next two terms are the cost of marginal extra capacity, required for an increase in traffic, for the physical and logical paths weighted by the proportion of the traffic that is accepted on the path.

It is interesting to note that if all the traffic for stream $s \in S_I$ uses its logical path, $l(s)$, then $\theta_s = 0$ and $A_s = a_{l(s)}$. This causes equation (7.2.6) which the optimal tariff $\alpha_s^*$, $s \in S_I$, must satisfy, to be $\alpha_s = -\nu_s\left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + M_{l(s)}$. Similarly, if all the traffic for stream $s \in S_I$ uses its physical path, $p(s)$, then $\theta_s = 1$ and $A_s = a_{p(s)}$. In this case, equation (7.2.6) which the optimal tariff $\alpha_s^*$, $s \in S_I$, must satisfy becomes $\alpha_s = -\nu_s\left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + M_{p(s)}$.

These equations are similar to the forms we expected from the numerical results in Section 6.7. Recall that from our numerical results we were expecting the value of the optimal tariffs to depend on a term due to the elasticity function and a term due to the cost of multiplexing and/or cross-connecting the traffic. The only difference between this and the formulae above is that the cost terms are weighted by the cost of marginal extra traffic required to support extra traffic on each link, which is $\frac{\partial C}{\partial \rho}(1-E)$. Using the bounds for $C(\rho, E)$ given by Berezner *et al.* [11] (see Section 6.4), we know that as $\rho$ increases, $\frac{\partial C}{\partial \rho}$ approaches $1-E$, and hence this marginal weighting approaches 1.

Note that if we place upper and lower bounds on the tariff, then the optimal tariff occurs at the upper bound if $\alpha_s^* > \overline{\alpha}_s$ or at the lower bound if $\alpha_s^* < \underline{\alpha}_s$. This can be shown by using similar arguments on the Lagrangian relaxation of the formulation.

### 7.2.2 Optimal splitting probabilities

We now turn to the problem of determining the optimal splitting probabilities given that the tariffs, $\boldsymbol{\alpha}$, and the link blocking probabilities, $\boldsymbol{E}$, are fixed and feasible. We wish to show that there is only one stationary point for the revenue as a function of $\boldsymbol{\theta}$, and to classify this stationary point.

Differentiating the objective function (6.3.16) with respect to $\theta_s$, $s \in S_I$ gives,

$$\frac{\partial Z}{\partial \theta_s} = \alpha_s \nu_s (a_{p(s)} - a_{l(s)}) - 2m \sum_{j \in p(s)} \frac{\partial C}{\partial \rho_j}\frac{\partial \rho_j}{\partial \theta_s} - (2m + x(\eta_{\ell(s)} - 1))\frac{\partial C}{\partial \rho_{\ell(s)}}\frac{\partial \rho_{\ell(s)}}{\partial \theta_s}$$

where

$$\frac{\partial \rho_j}{\partial \theta_s} = \begin{cases} \varsigma_{jp(s)}\nu_s a_{p(s)}(1-E_j)^{-1} & j \in p(s), \quad s \in S_I \\ -\varsigma_{jl(s)}\nu_s a_{l(s)}(1-E_j)^{-1} & j \in l(s), \quad s \in S_I \\ 0 & \text{otherwise.} \end{cases}$$

Hence

$$\frac{\partial Z}{\partial \theta_s} = \nu_s a_{p(s)}(\alpha_s - M_{p(s)}) - \nu_s a_{l(s)}(\alpha_s - M_{l(s)}). \tag{7.2.7}$$

The $\boldsymbol{\alpha}$ which satisfies equation (7.2.6), maximises Z for any feasible $\boldsymbol{\theta}$. Thus we wish to find the $\boldsymbol{\theta}$ that maximises Z using the fixed $\boldsymbol{\alpha}$ which satisfies equation (7.2.6).

Substituting $\alpha_s^*, s \in S_I$, from equation (7.2.6) into this equation gives

$$\begin{aligned} \frac{\partial Z}{\partial \theta_s} &= \nu_s a_{p(s)}\Big( - \nu_s \frac{\partial \nu_s}{\partial \alpha_s}^{-1} + (M_{l(s)} - M_{p(s)})\frac{(1-\theta_s)a_{l(s)}}{A_s}\Big) \\ &\quad - \nu_s a_{l(s)}\Big( - \nu_s \frac{\partial \nu_s}{\partial \alpha_s}^{-1} + (M_{p(s)} - M_{l(s)})\frac{\theta_s a_{p(s)}}{A_s}\Big). \end{aligned}$$

Hence,

$$\frac{\partial Z}{\partial \theta_s} = (\nu_s)^2 \left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} \left(a_{l(s)} - a_{p(s)}\right) - \left(\frac{\nu_s a_{p(s)} a_{l(s)}}{A_s}\right)\left(M_{p(s)} - M_{l(s)}\right). \tag{7.2.8}$$

We now set $\frac{\partial Z}{\partial \theta_s}$ to zero in equation (7.2.8) to find the stationary points. In solving this equation, we assume $\theta_s$ is no longer restricted to $[0, 1]$ and it is possible that $A_s$ will equal zero. From equation (6.2.7) this gives a vertical asymptote of $Z$ at $\theta_s = \Upsilon_s = \frac{-a_{l(s)}}{a_{p(s)} - a_{l(s)}}$. We shall investigate this possibility later in our proof, but for the moment we shall assume that $A_s \neq 0$. Note that if $a_{l(s)} = 0$, $\Upsilon_s = 0$ and if $a_{p(s)} = 0$, $\Upsilon_s = 1$. If $a_{p(s)} = a_{l(s)} = 0$, no traffic will be accepted from stream $s$ and hence the splitting probability for that stream will be irrelevant.

Using our arguments in Section 7.2.1, we can assume that $\frac{\partial \nu_s}{\partial \alpha_s} \neq 0$ and $\nu \neq 0$. If $a_{p(s)} - a_{l(s)} = 0$, the traffic for stream $s$ will simply use the path $p \in P_s$ with lower cost rendering this analysis unnecessary for stream $s$. Thus we can rearrange equation (7.2.8) to find

$$\begin{aligned} A_s &= (1-\theta_s)a_{\ell(s)} + \theta_s a_{p(s)} \\ &= \frac{-\nu_s a_{p(s)} a_{l(s)} \left(M_{p(s)} - M_{l(s)}\right)}{\left(\nu_s\right)^2 \left(\dfrac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} \left(a_{p(s)} - a_{l(s)}\right)}. \end{aligned}$$

Hence, for all $s \in S_I$, $\theta_s^*$ must satisfy

$$\theta_s = \frac{-a_{l(s)}}{a_{p(s)} - a_{l(s)}} \left( \frac{\left(M_{p(s)} - M_{\ell(s)}\right) a_{p(s)} \left(\dfrac{\partial \nu_s}{\partial \alpha_s}\right)}{\left(a_{p(s)} - a_{l(s)}\right) \nu_s} + 1 \right). \tag{7.2.9}$$

The optimal splitting probability $\theta_s^*$ is not given explicitly by equation (7.2.9) as $M_{p(s)}$ and $M_{l(s)}$ are both functions of $\dfrac{\partial C}{\partial \rho(\boldsymbol{\theta})}$, which is the cost of marginal extra capacity, required for an increase in traffic. The traffic on each link $j$ is dependent on the percentage of the traffic for each stream, $f_s^p$, $s \in S_I$ which uses each path $p \in P_s$ where link $j \in p(s)$. Since as $\rho$ increases, $\frac{\partial C}{\partial \rho} \to 1 - E$, this dependency is weak.

In this section, we have assumed that the tariffs, $\alpha_s$, are fixed and feasible. However, it is possible that the tariff $\alpha_s^*$, at which we evaluate the optimal splitting probabilities, is not feasible. In Section 7.2.1 we stated that if the tariff $\alpha_s^*$ is not feasible, the optimal tariff occurs at the upper bound if $\alpha_s^* > \overline{\alpha}_s$ or at the lower bound if $\alpha_s^* < \underline{\alpha}_s$. In this case the analysis becomes more involved, as it is not be possible to rearrange $\frac{\partial Z}{\partial \theta}$ to find an equation for $\theta_s^*$. We will discuss this eventuality in detail in Section 7.4.2, in which we use an approximation.

We would now like to proceed by finding the second derivative of the objective function (6.3.16) with respect to $\theta_s$, at $\theta^*$, if equation (7.2.9) gives a single stationary point. This would allow us to determine whether the objective function evaluated at $\boldsymbol{\theta}^*$ gives a maximum or minimum. If a maximum were obtained, it would then be necessary to investigate whether the vector $\boldsymbol{\theta}$ was feasible or not. If a minimum were obtained, then an end-point of the feasible region for $\boldsymbol{\theta}$ would be chosen for each $\theta_s^*$. We are unable to proceed in this manner as we do not have an explicit expression for $\theta_s^*$ at which we could evaluate the second derivative.

In Section 7.4 we will use an approximation for $\frac{\partial C}{\partial \rho}$ that removes the dependency from equation (7.2.8). This will enable us to proceed as described above, and thus determine the optimal vector $\boldsymbol{\theta}^*$. However, before doing so, we will examine equation (7.2.9) to see what information can be gleaned.

### 7.2.3 Investigation of $\theta$ equation

When solving the optimisation problem there is a trade-off between the route with the best performance (that is, lowest blocking) and the route with the best cost. If the acceptance probabilities on the routes are equal, then it would be expected that the route choice problem would simply come down to choosing the best cost route. If the cost coefficients are equal it would be expected that the traffic would simply be routed on the path with the higher acceptance probability. Furthermore, if both the acceptance probabilities and the costs are equal then it is expected that the route choice (choice of splitting probabilities) is arbitrary. We now wish to investigate whether we can show that this is so.

If $0 < \theta_s < 1$ and there is only one stationary point, we need to know whether the stationary point gives a maximum or a minimum. If there is only one stationary point $\boldsymbol{\theta}$ and $\theta_s < 0$ or $\theta_s > 1$, that is, $\theta_s$ is not feasible, we know to choose an endpoint of the feasible region as $\theta_s^*$. However, we need more information about the solution space to know which end-point to choose.

Note that, rather than stating that one of the points $\theta_s = 0$ or $\theta_s = 1$ will be chosen, we state that we will choose an end-point of the feasible region for $\theta_s$. This is due to the fact that, in some rare cases, $\theta_s$ will be constrained to a tighter region than $[0, 1]$, by constraint (6.3.20) in the problem formulation. We will investigate this possibility in the next section, for the moment continuing to use a term such as *within the feasibility region.*

Earlier we defined the point at which $A_s = 0$, and hence at which there is a vertical asymptote of $Z$, as

$$\Upsilon_s = \frac{-a_{l(s)}}{a_{p(s)} - a_{l(s)}}. \tag{7.2.10}$$

For $a_{p(s)} - a_{l(s)} > 0$ we have $\Upsilon_s < 0$. For $a_{p(s)} - a_{l(s)} < 0$ we have $\Upsilon_s > 1$.

Let us now define

$$k_s = \frac{-a_{l(s)}a_{p(s)}(\frac{\partial \nu_s}{\partial \alpha_s})}{(a_{p(s)} - a_{l(s)})^2 \nu_s}. \tag{7.2.11}$$

Using our arguments from Section 7.2.1, we may assume $\nu_s > 0$ and $\frac{\partial \nu_s}{\partial \alpha_s} < 0$. In Section 7.2.2 we explained that it is possible to assume $a_{p(s)} - a_{l(s)} \neq 0$. Hence, using these assumptions, $k_s \geq 0$. Thus equation (7.2.9) can be written as

$$\theta_s = \left(M_{p(s)} - M_{\ell(s)}\right) k_s + \Upsilon_s. \tag{7.2.12}$$

Note that if $a_{p(s)} = 0$, then $k_s = 0$ and $\theta_s = \Upsilon_s = 1$. Also, if $a_{l(s)} = 0$, then $k_s = 0$ and $\theta_s = \Upsilon_s = 0$. In both these cases the splitting probability is set and no further analysis is necessary. Thus, we will exclude these two cases in the following, giving $k_s > 0$.

In order to investigate equation (7.2.12) further, we will assume we have found $\hat{\theta}_s$ that satisfies this equation for $s \in S_I$. Although $\boldsymbol{\theta}$ will maximise the objective function, it will not necessarily be the optimal $\boldsymbol{\theta}^*$ for our problem formulation as one or more of the $\hat{\theta}_s$ values may not be feasible, due to not satisfying the constraints (6.3.19). Thus, we will investigate several cases in turn, and in each case attempt to determine whether or not $\hat{\theta}_s$, $s \in S_I$, is feasible. This will enable us to make some assertions about the value of $\theta^*_s$, $s \in S_I$.

- Case 1 : $M_{p(s)} - M_{\ell(s)} > 0$ and $a_{p(s)} - a_{l(s)} < 0$

  In this case the cost of marginal extra capacity on the physical path, $p(s)$, is more expensive than on the logical path, $l(s)$, and the acceptance probability on the physical path is lower than that on the logical path. Thus we would expect to use the logical path as much as possible within feasibility. Using equation (7.2.12), and the fact that $\Upsilon_s > 1$, gives $\hat{\theta}_s > 1$ which is not feasible. Therefore the lower end-point of the feasible region will be chosen.

- Case 2 : $M_{p(s)} - M_{\ell(s)} < 0$ and $a_{p(s)} - a_{l(s)} > 0$

  In this case the cost of marginal extra capacity on the logical path, $l(s)$, is more expensive than on the physical path $p(s)$, and the acceptance probability on the logical path is lower than that on the physical path. Thus we would expect to use the physical path as much as possible within feasibility. Using equation (7.2.12), and the fact that $\Upsilon_s < 0$, gives $\hat{\theta}_s < 0$ which is not feasible. Therefore the upper end-point of the feasible region will be chosen.

- Case 3 : $M_{p(s)} - M_{\ell(s)} < 0$ and $a_{p(s)} - a_{l(s)} < 0$ or

  $M_{p(s)} - M_{\ell(s)} > 0$ and $a_{p(s)} - a_{l(s)} > 0$

  In this case the cost of marginal extra capacity on one path is more expensive, but that path has a higher acceptance probability. Hence which path we use, or whether we use a combination of the paths, would be expected to be dependent on the magnitude of the differences in the marginal costs and the differences in the acceptance probabilities. Using equation (7.2.12) it is not possible to know whether $\hat{\theta}_s$ is feasible or not without knowing the magnitudes of the terms. Thus, we have no information about which path will be chosen or whether the traffic will be split.

- Case 4 : $M_{p(s)} - M_{l(s)} = a_{p(s)} - a_{l(s)}$

  In this case the difference in the costs of marginal extra circuits for the paths is equivalent to the difference in the acceptance probabilities. Thus, we may expect that, as there is no advantage in using one path over another, we could possibly have the traffic split over the paths. Using equation (7.2.12) we find that

  1. if $a_{p(s)} - a_{l(s)} < 0$, then $0 < \hat{\theta}_s < 1$ if $a_{p(s)} < -\nu_s(\frac{\partial \nu_s}{\partial \alpha_s})^{-1} < a_{l(s)}$.
  2. if $a_{p(s)} - a_{l(s)} > 0$, then $0 < \hat{\theta}_s < 1$ if $a_{l(s)} < -\nu_s(\frac{\partial \nu_s}{\partial \alpha_s})^{-1} < a_{p(s)}$.

  Since $a_{p(s)} \leq 1$ and $a_{l(s)} \leq 1$, if $-\nu_s(\frac{\partial \nu_s}{\partial \alpha_s})^{-1} \geq 1$ then $\hat{\theta}_s$ will not be feasible, that is, $\hat{\theta}_s \leq 0$ or $\hat{\theta}_s \geq 1$. In the example Section 6.6, $-\nu_s(\frac{\partial \nu_s}{\partial \alpha_s})^{-1} = 3$ and hence $\hat{\theta}_s$ would not be feasible and $\theta_s^*$ would be an end-point.

- Case 5 : $a_{p(s)} - a_{l(s)} = \epsilon, \epsilon \to 0$

  If the acceptance probabilities are nearly equivalent we would expect to use the path with lower cost as much as possible. Equation (7.2.12) shows that $\hat{\theta}_s \to \pm\infty$ depending on the sign of $M_{p(s)} - M_{\ell(s)}$. Hence $\hat{\theta}_s$ is not feasible, and again an end-point of the feasible region would be chosen.

- Case 6 : $a_{p(s)} - a_{l(s)} = 0$

  If the acceptance probabilities on the physical and logical paths are equal for any stream $s \in S_I$, then choice of splitting probability would simply be a matter of choosing the path with the lower cost, and analysis would not be necessary.

- Case 7 : $M_{p(s)} - M_{l(s)} = \epsilon, \epsilon \to 0$

  If the costs are nearly equal we would expect to use the path with the higher acceptance probability as much as possible. In this case equation (7.2.12) gives $\hat{\theta}_s \to \Upsilon_s$ which is either $< 0$ or $> 1$ depending on the sign of $a_{p(s)} - a_{l(s)}$. Hence $\hat{\theta}_s$ is not feasible again and an end-point of the feasible region will be chosen.

- Case 8: $M_{p(s)} - M_{l(s)} = 0$ If the costs of marginal extra capacity on the physical and logical paths are equal for any stream $s \in S_I$, traffic for the stream would use the path with the higher acceptance probability and the analysis would not be necessary.

Thus, even without an explicit equation for $\theta_s^*$, we have found that there are many cases in which $\hat{\theta}_s$ will not be feasible and therefore an end-point of the feasible region for $\theta_s$ will be selected as $\theta_s^*$. In Section 7.4 we will be able to extend this to show that $\theta_s^*$ will always be an end-point of the feasible region. However, first we will investigate what are the end-points of the feasible region for $\theta_s$.

## 7.3 End-points of the feasible region for $\theta_s$

In attempting to find the optimal splitting probabilities we have assumed that the link blocking probabilities, $\boldsymbol{E}$, and tariffs, $\boldsymbol{\alpha}$, are fixed. Thus it is possible to rearrange constraint (6.3.20) to find the feasible region for $\theta_s$, $s \in S_I$.

Using equations (6.2.5) and (6.2.6) gives

$$B_s(\theta_s, \boldsymbol{E}) = (1-\theta_s)(1-a_{l(s)}) + \theta_s(1-a_{p(s)}),$$

which we will write as $B_s$ in future. Hence inequality (6.3.20) becomes

$$a_{l(s)} - (1-\underline{B}_s) \le \theta_s(a_{l(s)} - a_{p(s)}) \le a_{l(s)} - (1-\overline{B}_s).$$

Thus there are now two possibilities. The first is when $a_{l(s)} - a_{p(s)} > 0$ and hence

$$\frac{a_{l(s)} - (1-\underline{B}_s)}{a_{l(s)} - a_{p(s)}} \le \theta_s \le \frac{a_{l(s)} - (1-\overline{B}_s)}{a_{l(s)} - a_{p(s)}}.$$

This can constrain the value of $\theta_s$ to an interval tighter than the interval $[0,1]$. Specifically

$$\begin{aligned} &\frac{a_{l(s)} - (1-\underline{B}_s)}{a_{l(s)} - a_{p(s)}} > 0 \quad \text{if} \quad a_{l(s)} > (1-\underline{B}_s) \\ \text{and } &\frac{a_{l(s)} - (1-\overline{B}_s)}{a_{l(s)} - a_{p(s)}} < 1 \quad \text{if} \quad a_{p(s)} < (1-\overline{B}_s). \end{aligned}$$

The second possibility is when $a_{l(s)} - a_{p(s)} < 0$ and

$$\frac{a_{l(s)} - (1-\overline{B}_s)}{a_{l(s)} - a_{p(s)}} \le \theta_s \le \frac{a_{l(s)} - (1-\underline{B}_s)}{a_{l(s)} - a_{p(s)}}.$$

Again, this can constrain the value of $\theta_s$ to a tighter interval than $[0,1]$, where

$$\begin{aligned} &\frac{a_{l(s)} - (1-\overline{B}_s)}{a_{l(s)} - a_{p(s)}} > 0 \quad \text{if} \quad a_{l(s)} < (1-\overline{B}_s) \\ \text{and } &\frac{a_{l(s)} - (1-\underline{B}_s)}{a_{l(s)} - a_{p(s)}} < 1 \quad \text{if} \quad a_{p(s)} > (1-\underline{B}_s). \end{aligned}$$

In both cases, if the acceptance probabilities on the logical and physical paths, $a_{l(s)}$ and $a_{p(s)}$, are between $1-\overline{B}_s$ and $1-\underline{B}_s$ then $0 \le \theta_s \le 1$ and there are no tighter bounds imposed on $\theta_s$ than those in constraint (6.3.19) in the original formulation.

## 7.4 Optimisation using an approximation

In Sections 7.2.1 and 7.2.2 we found formulae that the optimal tariffs and splitting probabilities must obey. However, as the formula for $\theta_s$ was not an explicit equation for $\theta_s^*$, we were only able to form certain theories about the optimal value of $\theta_s$, $s \in S_I$, rather than fully analyse it. In Section 7.2.2 we noted that the dependency on $\boldsymbol{\theta}$ in equation (7.2.9) is weak and hence removing this dependency would allow us to obtain further information about $\boldsymbol{\theta}^*$.

Bean and Taylor [10] and Berezner *et al.* [11] stated that a good approximation for $\frac{\partial C}{\partial \rho_j}$, when $\rho_j$ is large, is

$$\frac{\partial C}{\partial \rho_j} \approx 1 - E_j. \tag{7.4.13}$$

This approximation has already been mentioned in Section 6.5 and Figure 6.5.4 shows a comparison of $\frac{\partial C}{\partial \rho_j}$ to this approximation.

Using this approximation, $M_p$ in equation (7.2.2) becomes a fixed cost, and will now be called $F_p$. Thus,

$$F_p = 2m \sum_{j \in \mathcal{J} \in p} \varsigma_{jp} + x \sum_{j \in \mathcal{J}_L \in p} (\eta_j - 1)\varsigma_{jp}, \tag{7.4.14}$$

and hence, $F_{\delta(s)} = 2m\varsigma_{d(s)\delta(s)}$, $F_{l(s)} = (2m+x(\eta_{\ell(s)}-1)\varsigma_{\ell(s)l(s)}$ and $F_{p(s)} = \sum_{j \in p(s)} 2m\varsigma_{jp(s)}$. These costs are no longer dependent on the marginal capacity increase due to extra traffic as equation (7.4.13) does not vary as $\rho$ varies. Hence $F_p$ now represents the cost of the circuits required by path $p$, regardless of the amount of traffic on the path.

### 7.4.1 Optimal tariffs using an approximation

We now wish to examine the changes that using this approximation causes in the analytical work of Section 7.2.1, where we found an equation that the optimal tariffs must satisfy, without the approximation.

For fixed link blocking probabilities $\boldsymbol{E}$ and splitting probabilities $\boldsymbol{\theta}$, the optimal tariff $\alpha_s^*$, $s \in S_P$ must now satisfy

$$\alpha_s = -\nu_s \left( \frac{\partial \nu_s}{\partial \alpha_s} \right)^{-1} + F_{\delta(s)}. \tag{7.4.15}$$

The term $-\nu_s(\frac{\partial \nu_s}{\partial \alpha_s})^{-1}$ depends only on the elasticity function of the traffic $\nu_s$, as in equation (7.2.3), but the second term $F_{\delta(s)}$ is now the fixed cost of carrying the call on the physical path, consisting of a single link, rather than being a marginal cost, as in equation (7.2.3).

For $s \in S_I$, the optimal tariff $\alpha_s^*$ must satisfy

$$\alpha_s = -\nu_s \left( \frac{\partial \nu_s}{\partial \alpha_s} \right)^{-1} + F_{p(s)} \frac{\theta_s a_{p(s)}}{A_s} + F_{l(s)} \frac{(1-\theta_s) a_{l(s)}}{A_s}. \tag{7.4.16}$$

Once again the term $-\nu_s(\frac{\partial \nu_s}{\partial \alpha_s})^{-1}$ depends only on the elasticity function of the traffic $\nu_s$. The difference between this equation and equation (7.2.6) is that the next two terms now represent the fixed weighted cost of carrying a call on the physical and logical paths respectively, as against a marginal cost. It is interesting to note that this is the form of the equation we observed for $\alpha_s$ from our numerical results in Section 6.7.

## 7.4.2 Optimal splitting probabilities using an approximation

We now wish to examine the changes that using this approximation causes in the analytical work of Section 7.2.2, where we found an equation that the optimal splitting probabilities must satisfy, without the approximation.

As $\theta_s$ is no longer a function of $\frac{\partial C}{\partial \rho}$, instead of having equation (7.2.9) which $\theta_s^*$ must satisfy, we now have an explicit equation for $\theta_s^*$, $s \in S_I$. Thus

$$\theta_s^* = \frac{-(F_{p(s)} - F_{l(s)}) a_{p(s)} a_{l(s)} \left( \frac{\partial \nu_s}{\partial \alpha_s} \right)}{\left( a_{p(s)} - a_{l(s)} \right)^2 \nu_s} - \frac{a_{l(s)}}{a_{p(s)} - a_{l(s)}} \tag{7.4.17}$$

where $F_{p(s)} - F_{l(s)}$ is the difference in the fixed costs of the physical and logical paths. This replaces the term $M_{p(s)} - M_{l(s)}$ from Section 7.2.2, which was a marginal cost difference. Although $\theta_s^*$ is a critical point for revenue, it is may not be a viable physical solution with $0 \le \theta_s^* \le 1$, for all $s \in S_I$.

Using the approximation (7.4.13), equation (7.2.7) becomes

$$\frac{\partial Z}{\partial \theta_s} = \nu_s a_{p(s)}(\alpha_s - F_{p(s)}) - \nu_s a_{l(s)}(\alpha_s - F_{l(s)}). \tag{7.4.18}$$

In Section 7.2.1, we found a formula for the optimal splitting probability, without approximation (7.4.13). There we discussed the possibility that the tariff $\alpha_s^*$, at which we evaluate the splitting probability, may not be feasible. We noted that the optimal tariff occurs at the upper bound if $\alpha_s^* > \overline{\alpha}_s$ or at the lower bound if $\alpha_s^* < \underline{\alpha}_s$. In these cases, we know from equation (7.4.18) that $\frac{\partial Z}{\partial \theta_s} = c$, where $c$ is a constant. Thus the second derivative $\frac{\partial^2 Z}{\partial \theta_s^2}$ will have value zero and $\theta_s$ will be a point of inflexion. Since there is no maximum or minimum, the optimal splitting probability $\theta_s^*$ will be an endpoint of the feasible region for $\theta_s$.

For feasible $\alpha_s^*$, substitution of (7.4.16) into (7.4.18) gives

$$\frac{\partial Z}{\partial \theta_s} = (\nu_s)^2 \left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} \left(a_{l(s)} - a_{p(s)}\right) - \left(\frac{(F_{p(s)} - F_{l(s)})\nu_s a_{p(s)} a_{l(s)}}{A_s}\right). \tag{7.4.19}$$

Differentiating this with respect to $\theta_s$ and noting that $\frac{\partial A_s}{\partial \theta_s} = a_{p(s)} - a_{l(s)}$ gives

$$\frac{\partial^2 Z}{\partial \theta_s^2} = A_s^{-2} \nu_s a_{p(s)} a_{l(s)} (F_{p(s)} - F_{l(s)})(a_{p(s)} - a_{l(s)}). \tag{7.4.20}$$

Using equation (7.4.17) for $\theta_s^*$ and substituting it into the equation for the average stream acceptance gives

$$A_s(\theta_s^*) = \theta_s^*(a_{p(s)} - a_{l(s)}) + a_{l(s)} = -\frac{(F_{p(s)} - F_{l(s)}) a_{p(s)} a_{l(s)} \frac{\partial \nu_s}{\partial \alpha_s}}{(a_{p(s)} - a_{l(s)})\nu_s} \tag{7.4.21}$$

and hence the second derivative of the objective function, with respect to $\theta_s$, at $\theta_s^*$, is

$$\left.\frac{\partial^2 Z}{\partial \theta_s^2}\right|_{\theta_s = \theta_s^*} = \frac{a_{p(s)} - a_{l(s)}}{(F_{p(s)} - F_{l(s)})} \left[\frac{(a_{p(s)} - a_{l(s)})^2 \left(\nu_s\right)^3}{a_{p(s)} a_{l(s)} \left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^2}\right]. \tag{7.4.22}$$

Let $g_s(\nu_s(\alpha_s, \boldsymbol{E})) = -\nu_s^2 \left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1}$ and $h_s(\boldsymbol{E}, \nu_s(\alpha_s, \boldsymbol{E})) = \nu_s a_{p(s)} a_{l(s)}$. Recall from Section 7.2.3, that if $a_{p(s)} = 0$, then $\theta_s^* = 1$ or if $a_{l(s)} = 0$, then $\theta_s^* = 0$. Using these facts and the arguments in Section 7.2.1, we can assume that $g_s$ and $h_s$ are strictly positive. Hence, suppressing the dependencies we have

$$\theta_s^* = (F_{p(s)} - F_{l(s)})k_s + \Upsilon_s \qquad \text{and} \tag{7.4.23}$$
$$\frac{\partial Z}{\partial \theta_s} = g_s(a_{p(s)} - a_{l(s)}) - h_s\frac{(F_{p(s)} - F_{l(s)})}{A_s} \tag{7.4.24}$$

where $k_s, g_s$ and $h_s$ are positive and $k_s$ and $\Upsilon_s$ are given by equations (7.2.11) and (7.2.10) respectively.

It is now necessary to determine whether the critical value $\theta_s^*$ maximises or minimises revenue as a function of $\boldsymbol{\theta}$. If $\theta_s^*$ gives a minimum then the value of $\theta_s$ that maximises the revenue will be one of the end-points of the feasible region for $\theta_s$. If $\theta_s^*$ gives a maximum then the optimal value of $\theta_s$ will still be one of the end-points of the feasible region for $\theta_s$ if $\theta_s^*$ is outside this feasible region.

If, for any stream $s \in S$, the fixed cost of the physical path, $p(s)$, is equal to the fixed cost of the logical path, $l(s)$, that is $F_{p(s)} - F_{l(s)} = 0$, traffic for that stream will simply use the path with the higher acceptance probability. Thus for that stream the analytical analysis will not be necessary. Similarly, if the acceptance probabilities on the two paths are equal, that is $a_{p(s)} - a_{l(s)} = 0$, traffic for that stream will use the path with lower cost. Again, the analytical analysis for this stream will not be necessary. Thus we can assume that $F_{p(s)} - F_{l(s)} \neq 0$ and $a_{p(s)} - a_{l(s)} \neq 0$.

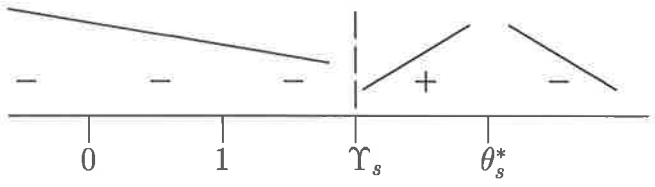
Using these arguments and the arguments in Section 7.2.1 we can assume that the denominator and numerator of the expression in the square brackets in equation (7.4.22) are strictly positive. Thus, there are two cases in which $\frac{\partial^2 Z}{\partial \theta_s^2}$ is negative and hence $\theta_s^*$ gives a maximum.

Before considering the two cases, it is worth noting that the only difference between these two cases and Case 1 and Case 2 in Section 7.2.3 is that the costs are fixed rather than marginal. We also know that these cases cause $\theta_s^*$ to be a maximum. As we have an equation for $\frac{\partial Z}{\partial \theta_s}$ we are able to determine which end-point will be used. We now know that all the other cases in Section 7.2.3 give a $\theta_s^*$ which minimises the objective function. Thus, whether $\theta_s^*$ was feasible in these cases or not was irrelevant as an end-point of the feasible region would be chosen regardless. We will now examine the two cases.

- Case 1 : $F_{p(s)} - F_{l(s)} > 0$ and $a_{p(s)} - a_{l(s)} < 0$

  In this case the fixed cost of the physical path, $p(s)$, is more expensive than that of the logical path, $l(s)$, and the acceptance probability on the physical path is lower than that on the logical path. Thus we would expect to use the logical path as much as possible within feasibility.

  Using equation (7.4.23) gives $1 < \Upsilon_s < \theta_s^*$. Therefore, $\theta_s^*$ is infeasible and we know that the optimal splitting probability will be one of the end-points of the feasible region. Knowing that $\theta_s^*$ gives a maximum, we know what the sign diagram of $\frac{\partial Z}{\partial \theta_s}$ looks like for $\theta_s > \Upsilon_s$. For $\theta_s < \Upsilon_s$, say $\theta_s = \Upsilon_s - b$, for some positive number $b$,
  $A_s = (\frac{-a_{l(s)}}{a_{p(s)}-a_{l(s)}} - b)(a_{p(s)} - a_{l(s)}) + a_{l(s)} = -b(a_{p(s)} - a_{l(s)}) > 0.$
  Hence using equation (7.4.24) we know that $\frac{\partial Z}{\partial \theta_s} < 0$ for $\theta_s < \Upsilon_s$. Thus the sign diagram, for $\frac{\partial Z}{\partial \theta_s}$ is

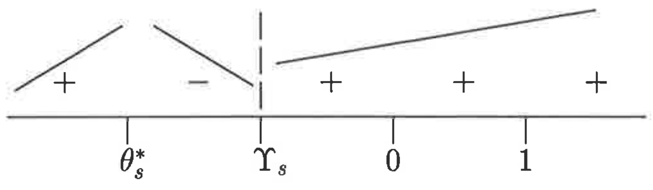
  

  Hence, the maximum value of $Z(\theta_s)$ is obtained at the lower bound, which indicates that as much of the traffic uses the logical path as is possible within the feasibility conditions as expected.

- Case 2 : $F_{p(s)} - F_{l(s)} < 0$ and $a_{p(s)} - a_{l(s)} > 0$

  In this case the fixed cost of the logical path, $l(s)$, is more expensive than that of the physical path $p(s)$, and the acceptance probability on the logical path is lower than that on the physical path. Thus we would expect to use the physical path as much as possible within feasibility.

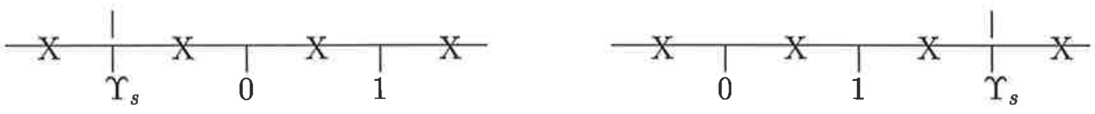
Using equation (7.4.23) gives $\theta_s^* < \Upsilon_s < 0$. Therefore, $\theta_s^*$ is infeasible and we know that the optimal splitting probability will be one of the end-points of the feasible region. Knowing that $\theta_s^*$ gives a maximum, we know what the sign diagram of $\frac{\partial Z}{\partial \theta_s}$ looks like for $\theta_s < \Upsilon_s$. For $\theta_s > \Upsilon_s$, say $\theta_s = \Upsilon_s + b$, for some positive number $b$,
$A_s = (\frac{-a_{l(s)}}{a_{p(s)}-a_{l(s)}} + b)(a_{p(s)} - a_{l(s)}) + a_{l(s)} = b(a_{p(s)} - a_{l(s)}) > 0.$
Hence using equation (7.4.24) we know that $\frac{\partial Z}{\partial \theta_s} > 0$ for $\theta_s > \Upsilon_s$. Thus the sign diagram, for $\frac{\partial Z}{\partial \theta_s}$ is



Hence, the maximum value of $Z(\theta_s)$ is obtained at the upper bound, indicating that as much of the traffic uses the physical path as is possible within the feasibility conditions as expected.

Hence the value of $\theta_s$ that maximises the revenue as a function of $\boldsymbol{\theta}$ is one of the end-points of the feasible region of $\theta_s$.

We will now investigate which end-point will be used in all other cases. This will be dependent on the relation of the value of $\theta_s^*$ to $\Upsilon_s, 0$ and 1. All eight possibilities are shown below, where the possible value of $\theta_s^*$ is represented by X.



By using these possibilities and the following conditions

$$\Upsilon_s < 0 \text{ iff } a_{p(s)} - a_{l(s)} > 0,$$

$$\Upsilon_s > 0 \text{ iff } a_{p(s)} - a_{l(s)} < 0,$$

$$\theta_s^* < \Upsilon_s \text{ iff } F_{p(s)} - F_{l(s)} < 0,$$

$$\theta_s^* > \Upsilon_s \text{ iff } F_{p(s)} - F_{l(s)} > 0$$

and

$$\begin{aligned} Z'' \not> 0 \ \text{ iff } & a_{p(s)} - a_{l(s)} < 0 \text{ and } F_{p(s)} - F_{l(s)} < 0 \\ \text{or } & a_{p(s)} - a_{l(s)} > 0 \text{ and } F_{p(s)} - F_{l(s)} > 0, \\ Z'' \not< 0 \ \text{ iff } & a_{p(s)} - a_{l(s)} > 0 \text{ and } F_{p(s)} - F_{l(s)} < 0 \\ \text{or } & a_{p(s)} - a_{l(s)} < 0 \text{ and } F_{p(s)} - F_{l(s)} > 0, \end{aligned}$$

we obtain Table 7.4.2.

Table 7.4.2 informs us that if $\theta_s^* < 0$, the feasible $\theta_s^*$ that maximises revenue is at the upper bound of the feasible region for $\theta_s$. If $\theta_s^* > 1$, the feasible $\theta_s^*$ that maximises revenue is at the lower bound of the feasible region for $\theta_s$. If $0 < \theta_s^* < 1$ it represents a minimum and so it will be necessary to substitute each end-point of the feasible region into the objective function to find which end-point gives a larger value of $Z$.

| Sign $\theta_s^*$ | Position | $a_{p(s)} - a_{l(s)}$ | $F_{p(s)} - F_{l(s)}$ | $Z''$ | $\theta_s^*$ |
|---|---|---|---|---|---|
| $\theta_s^* < 0$ | $\theta_s^* < \Upsilon_s < 0$ | $+$ | $-$ | $-$ : max | upper bound |
| | $\Upsilon_s < \theta_s^* < 0$ | $+$ | $+$ | $+$ : min | upper bound |
| | $\theta_s^* < 0 < 1 < \Upsilon_s$ | $-$ | $-$ | $+$ : min | upper bound |
| $\theta_s^* > 1$ | $1 < \Upsilon_s < \theta_s^*$ | $-$ | $+$ | $-$ : max | lower bound |
| | $1 < \theta_s^* < \Upsilon_s$ | $-$ | $-$ | $+$ : min | lower bound |
| | $\Upsilon_s < 0 < 1 < \theta_s^*$ | $+$ | $+$ | $+$ : min | lower bound |
| $0 < \theta_s^* < 1$ | $\Upsilon_s < 0 < \theta_s^* < 1$ | $+$ | $+$ | $+$ : min | ? |
| | $0 < \theta_s^* < 1 < \Upsilon_s$ | $-$ | $-$ | $+$ : min | ? |

Table 7.4.1: Feasible optimal values of $\theta_s^*$.

## 7.5 Threshold theory

Observing Table 6.7.2 again, and noting that the traffic streams $s \in S_I$ are ordered from the lowest to the highest amount of traffic accepted, it appears that the splitting probability for each stream depends on the magnitude of traffic accepted by that stream. This is logical since there is an *Economy of Scale* trade-off for the capacity function $C(\rho_j, E_j)$ defined in equation (6.4.21). As traffic on the logical links does not mix with other traffic, using the logical path is less efficient but cheaper (in our example) than using the physical path. Hence, there is a certain threshold at which the amount of traffic accepted by a stream is high enough to justify creating a logical path.

The fact that this threshold appears to depend only on the magnitudes of the accepted traffic is in part due to the Quality of Service requirements being uniform for all paths, and the costs coefficients being approximately equal in our example. If the grades of service or cost coefficients were significantly different, it would be expected that whether a stream satisfies the threshold at which a logical link is created would also depend on the path blocking probabilities for the stream and the cost coefficients.

For instance, if a stream has a very low upper bound on the stream blocking (i.e. $\overline{B_s} \to 0$), a higher level of traffic will need to be accepted to take advantage of the Economy of Scale trade-off for each path, $p \in P_s$, than if the upper bound is less stringent. This is due to the derivative $\sum_{j \in p} \frac{\partial C}{\partial \rho_j}$.

If the cost of multiplexing is much higher than the cost of cross-connecting, the gains due to the cheaper cost of using the logical path may outweigh its inefficiency which is due to traffic being unable to mix. Thus there are many trade-offs for each stream which determine if they are above the threshold at which creating a logical link is cost effective. The trade-off of these terms can be seen in equation (7.2.9) which is the equation that $\theta_s^*$ must satisfy. The numerical example in Section 6.6 gives a clear illustration of the physical/logical threshold when the effects of the costs and the path blocking probabilities are insignificant and the traffic magnitude is the dominating factor.

# Chapter 8

# Multi-path formulation

## 8.1 Theory

In Chapter 4 we introduced the concept of *logical links* which have their own link blocking probability but use reserved capacity on some set of physical links. In the work to date, it has been assumed that traffic on logical paths, which consist of a single logical link, is not able to interact with traffic from other streams. Hence each stream $s \in S_I$ has been assumed to have a logical path associated with it, which is solely for its use. This logical path is in effect a direct link between the origin and the destination nodes, with its own blocking probability and capacity.

Due to the fact that the traffic on the logical links was unable to interact, the logical links were unable to make use of the *economy of scale* trade off, unlike the physical links. If cross-connecting is assumed to be less expensive than multiplexing/demultiplexing, then there is a trade-off between the marginal cost of circuits and the difference in the costs of cross-connecting and multiplexing, as to whether each logical link is cost effective.

This leads one to wonder what the effect would be on the network if traffic on the logical links was allowed to interact. By allowing streams to use paths which are a combination of logical and physical links, this is a possibility. For example, traffic for the stream Perth-Sydney could use the physical link Perth-Adelaide, and then use the logical link from Adelaide-Sydney. This logical link may also carry, for instance, traffic from the stream Adelaide-Sydney. This now means that the logical links are able to take advantage of the economy of scale by carrying traffic from more than one stream, making the logical links more cost-effective. Thus the question now is whether there is an advantage in streams splitting traffic over more than one path or whether they will continue to make use of a single path.

Allowing traffic to mix on the logical links means that in effect we have a fully meshed network, where each link has its own blocking probability and capacity. We will now investigate this extension, firstly discussing the changes in the definitions and some new notation. Most of the definitions and formulation remain unchanged.

In Section 6.2 we defined $s \in S_I$ to be the set of streams which have an indirect physical path as well as a direct logical path. These streams may now also use paths which are a combination of logical and physical links. We also defined $P_I$ to be the set of physically indirect paths, which consist of more than one physical link. This set now includes paths which are a combination of physical and logical links. Note that this does not include logical direct paths, as these are in the set $P_L$. In the previous work, for each stream $s \in S_I$, the set $P_s \cap P_I$ contained one path only. However, with our new definition there will be some streams for which this is no longer true. Thus we will now define the set of paths that stream $s \in S_I$ may use from the set of paths $P_I$ as $P_s^I = P_s \cap P_I$.

The routing scheme now allows calls for stream $s \in S_I$ to use each path $p \in P_s^I$ with probability $\theta_s^p$ and the logical direct path with probability $1 - \sum_{p \in P_s^I} \theta_s^p$. The $\theta_s$ variables from (6.3.15) become

$$\theta_s^p \;, \quad s \in S_I \;, \quad p \in P_s^I \;, \tag{8.1.1}$$

constraint (6.3.19) becomes

$$0 \le \theta_s^p \le 1 \;, \quad s \in S_I \;, \quad p \in P_s^I \;, \tag{8.1.2}$$

$$\sum_{p\in P_s^I} \theta_s^p \le 1\,, \qquad s \in S_I\,, \tag{8.1.3}$$

and the splitting probabilities given by (6.2.3) become

$$f_s^p = \begin{cases} 1\,, & p \in P_P\,, \\ 1 - \sum\limits_{p\in P_s^I} \theta_s^p\,, & p \in P_L\,, \\ \theta_s^p\,, & p \in P_I\,. \end{cases} \tag{8.1.4}$$

The derivatives of the non-linear constraint (6.3.20), for $s \in S$ are now

$$\frac{\partial B_s}{\partial E_j} = \begin{cases} \sum_{p\in P_s : j \in p} f_s^p \varsigma_{jp} (1-E_j)^{-1} \prod_{i\in p} (1-E_i)^{\varsigma_{ip}}, & j \in p \in P_s, \\ 0, & \text{otherwise}, \end{cases} \tag{8.1.5}$$

$$\frac{\partial B_s}{\partial \alpha_{s'}} = 0, \qquad \forall s' \in S, \tag{8.1.6}$$

$$\frac{\partial B_s}{\partial \theta_{s'}^p} = \begin{cases} (1-E_{\ell(s)})^{\varsigma_{\ell(s)l(s)}} - \prod_{j\in p(s)} (1-E_j)^{\varsigma_{jp(s)}}, & p \in P_s, \\ 0, & \text{otherwise}. \end{cases} \tag{8.1.7}$$

The rest of the formulation remains unchanged except that the summations over the paths now include a larger number of paths. We still have a logical direct path $l(s)$, consisting of a single logical link and a path $p(s)$ which consists solely of physical links for each stream $s \in S_I$. The expressions for the marginal costs, $M_p$, and the fixed costs, $F_p$ are still as in equations (7.2.2) and (7.4.14), but there are more possible paths. In fact, for a stream $s$, there are $2^{\eta_{\ell(s)}-1}$ possible paths.

In the next two sections we will make some assumptions similar to those in the previous chapter. Rather than re-state these assumptions for this formulation, we will take them as given.

## 8.1.1 Optimal tariffs

To find the optimal tariff $\alpha_s^*$ which maximises network profit, we will again assume we have fixed link blocking probabilities $\boldsymbol{E}$ and splitting probabilities $\boldsymbol{\theta}$. Differentiating the objective function (6.3.16) with respect to $\alpha_s, s \in S_P$ gives the same equation that $\alpha_s^*$ must satisfy as in Section 7.2.1, which is equation (7.2.3).

The objective function (6.3.16) differentiated with respect to $\alpha_s$, $s \in S_I$ gives

$$\begin{aligned}\frac{\partial Z}{\partial \alpha_s} &= \sum_{p\in P_s} f_s^p \nu_s \prod_{j\in p}(1-E_j)^{\varsigma_{jp}} + \sum_{p\in P_s} f_s^p \alpha_s \frac{\partial \nu_s}{\partial \alpha_s}\prod_{j\in p}(1-E_j)^{\varsigma_{jp}} - 2m\sum_{j\in\mathcal{J}}\frac{\partial C}{\partial \rho_j}\frac{\partial \rho_j}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s} \\ &\quad -x\sum_{j\in\mathcal{J}_L}(\eta_j-1)\frac{\partial C}{\partial \rho_j}\frac{\partial \rho_j}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s}.\end{aligned}$$

As in Section 7.2.1, as $\boldsymbol{\theta}$ and $\boldsymbol{E}$ have already been set such that they are feasible solutions, we have $A_s > 0$. Hence, setting $\frac{\partial Z}{\partial \alpha_s} = 0$ and rearranging, gives

$$\alpha_s = -\nu_s\left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + \left(2m\sum_{j\in\mathcal{J}}\frac{\partial C}{\partial \rho_j}\frac{\partial \rho_j}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s} + x\sum_{j\in\mathcal{J}_L}(\eta_j-1)\frac{\partial C}{\partial \rho_j}\frac{\partial \rho_j}{\partial \nu_s}\frac{\partial \nu_s}{\partial \alpha_s}\right)\left(\frac{\partial \nu_s}{\partial \alpha_s}A_s\right)^{-1}.$$

Substituting $\frac{\partial \rho_j}{\partial \nu_s}$, which is given by equation (7.2.4), into this equation and noting that $\frac{\partial \rho_j}{\partial \nu_s}$ is non-zero only if there is a path $p \in P_s$ such that the link $j$ is in the path $p$ and $f_s^p \neq 0$ gives

$$\begin{aligned}\alpha_s = &- \nu_s\left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + \left(2m\sum_{p\in P_s} f_s^p \prod_{i\in p}(1-E_i)^{\varsigma_{ip}} \sum_{j\in\mathcal{J}\in p}\varsigma_{jp}\frac{\partial C_j}{\partial \rho_j}(1-E_j)^{-1}\right)A_s^{-1} \\ &+ \left(x\sum_{p\in P_s} f_s^p \prod_{i\in p}(1-E_i)^{\varsigma_{ip}} \sum_{j\in\mathcal{J}_L\in p}\varsigma_{jp}(\eta_j-1)\frac{\partial C_j}{\partial \rho_j}(1-E_j)^{-1}\right)A_s^{-1}.\end{aligned}$$

Using equation (7.2.2) for $M_p$, the optimal tariff $\alpha_s^*$ must satisfy

$$\alpha_s = -\nu_s\left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + \left(\sum_{p\in P_s} f_s^p a_p M_p\right)A_s^{-1}. \qquad (8.1.8)$$

This differs from the equation (7.2.6), only in that there are now more terms due to the extra paths. Again, the first term depends only on the elasticity function of the traffic $\nu_s$. The summation term is the cost of the marginal extra capacity, required for an increase in traffic, for each path weighted by the proportion of traffic that is accepted on the path.

## 8.1.2 Optimal splitting probabilities

We now wish to determine the optimal splitting probabilities given that the tariffs $\boldsymbol{\alpha}$, and link blocking probabilities $\boldsymbol{E}$ are constant. In Section 7.2.2 we found that the optimal splitting probabilities were such that one of the two available paths, physical or logical, was used as much as possible subject to the feasibility conditions. Now that traffic is allowed to mix on the logical links and there are extra paths we wish to determine if this is still the case.

Differentiating the objective function (6.3.16) with respect to $\theta_s^{p'}$, $s \in S_I$, $p' \in P_s^I$ gives

$$\frac{\partial Z}{\partial \theta_s^{p'}} = \alpha_s \nu_s (a_{p'} - a_{l(s)}) - 2m \sum_{j \in \mathcal{J}} \frac{\partial C_j}{\partial \rho_j} \frac{\partial \rho_j}{\partial \theta_s^{p'}} - x \sum_{j \in \mathcal{J}_L} (\eta_j - 1) \frac{\partial C_j}{\partial \rho_j} \frac{\partial \rho_j}{\partial \theta_s^{p'}}$$

where

$$\frac{\partial \rho_j}{\partial \theta_s^{p'}} = \begin{cases} \varsigma_{jp'} \nu_s (1-E_j)^{-1} a_{p'} & j \in p' \\ -\varsigma_{\ell(s)l(s)} \nu_s (1-E_j)^{-1} a_{l(s)} & j = \ell(s) \\ 0 & \text{otherwise.} \end{cases}$$

Thus, substituting $\frac{\partial \rho_j}{\partial \theta_s^{p'}}$ into the equation for $\frac{\partial Z}{\partial \theta_s^{p'}}$ gives

$$\begin{aligned} \frac{\partial Z}{\partial \theta_s^{p'}} &= \alpha_s \nu_s (a_{p'} - a_{l(s)}) - \nu_s a_{p'} 2m \sum_{j \in \mathcal{J} \in p'} \varsigma_{jp'} \frac{\partial C_j}{\partial \rho_j} (1-E_j)^{-1} \\ &- \nu_s a_{p'} x \sum_{j \in \mathcal{J}_L \in p'} \varsigma_{jp'} (\eta_j - 1) \frac{\partial C_j}{\partial \rho_j} (1-E_j)^{-1} \\ &+ \nu_s a_{l(s)} \big(2m + x(\eta_{l(s)} - 1)\big) \varsigma_{\ell(s)l(s)} \frac{\partial C_{l(s)}}{\partial \rho_{l(s)}} (1-E_{l(s)})^{-1}. \end{aligned}$$

Using equation (7.2.2) for $M_p$ gives

$$\frac{\partial Z}{\partial \theta_s^{p'}} = \nu_s (a_{p'} - a_{l(s)}) \alpha_s - \nu_s a_{p'} M_{p'} + \nu_s a_{l(s)} M_{l(s)}. \tag{8.1.9}$$

As in Section 7.2.2, we wish to find $\boldsymbol{\theta}$ at the point $\boldsymbol{\alpha}^*$. Thus substituting equation (8.1.8), that $\alpha_s^*$ must satisfy, for $\alpha_s$ gives

$$\begin{aligned} \frac{\partial Z}{\partial \theta_s^{p'}} = \frac{\nu_s}{A_s} \Bigg\{ & \left(a_{p'} - a_{l(s)}\right) \left( -A_s \nu_s \left( \frac{\partial \nu_s}{\partial \alpha_s} \right)^{-1} + \sum_{p \in P_s^I} f_s^p a_p M_p + \Big(1 - \sum_{p \in P_s^I} \theta_s^p\Big) a_{l(s)} M_{l(s)} \right) \\ & - a_{p'} M_{p'} A_s + a_{l(s)} M_{l(s)} \left( \sum_{p \in P_s^I} f_s^p a_p + \Big(1 - \sum_{p \in P_s^I} \theta_s^p\Big) a_{l(s)} \right) \Bigg\}. \end{aligned}$$

In solving this equation, we shall assume that $\theta_s$ is no longer restricted to $[0,1]$ and thus it is possible that $A_s$ will equal zero. From equation (6.2.7) this gives a vertical asymptote of Z at $\theta_s^{p'} = \Upsilon_s^{p'} = \frac{-a_{l(s)} - \sum_{p\in P_s^I, p\neq p'} \theta_s^p (a_p - a_{l(s)})}{a_{p'} - a_{l(s)}}$. For the moment we will assume that $A_s \neq 0$.

It is possible to cancel $\theta_s^{p'}$ from all the terms except $A_s$ giving

$$\begin{aligned}\frac{\partial Z}{\partial \theta_s^{p'}} = &- (\nu_s)^2 \Big(a_{p'} - a_{l(s)}\Big)\left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} + \frac{\nu_s}{A_s}\Bigg\{\Big(a_{p'} - a_{l(s)}\Big)\Big(\sum_{p\in P_s^I, p\neq p'} f_s^p a_p M_p\Big) \\ &+ a_{l(s)} M_{l(s)} \Big(\sum_{p\in P_s^I, p\neq p'} f_s^p a_p + \Big(1 - \sum_{p\in P_s^I, p\neq p'} f_s^p\Big) a_{p'}\Big) \\ &- a_{p'} M_{p'} \Big(\sum_{p\in P_s^I, p\neq p'} f_s^p a_p + \Big(1 - \sum_{p\in P_s^I, p\neq p'} f_s^p\Big) a_{l(s)}\Big)\Bigg\}.\end{aligned}$$

Similar to the previous work, if there are two paths for which $a_{p'} - a_{l(s)} = 0$, the path with higher cost will not be considered. Setting $\frac{\partial Z}{\partial \theta_s^{p'}}$ to zero and assuming as in Section 7.2.1 that $\frac{\partial \nu_s}{\partial \alpha_s} \neq 0$ and $\nu_s \neq 0$ gives

$$\begin{aligned}\theta_s^{p'} = &\frac{\frac{\partial \nu_s}{\partial \alpha_s} \sum_{p\in P_s^I, p\neq p'} f_s^p \Big(a_p a_{p'} (M_p - M_{p'}) + a_{l(s)} a_p (M_{l(s)} - M_p) + a_{p'} a_{l(s)} (M_{p'} - M_{l(s)})\Big)}{\nu_s \Big(a_{p'} - a_{l(s)}\Big)^2} \\ &- \frac{\sum_{p\in P_s^I, p\neq p'} f_s^p (a_p - a_{l(s)})}{\Big(a_{p'} - a_{l(s)}\Big)} - \frac{\frac{\partial \nu_s}{\partial \alpha_s} a_{p'} a_{l(s)} (M_{p'} - M_{l(s)})}{\nu_s \Big(a_{p'} - a_{l(s)}\Big)^2} - \frac{a_{l(s)}}{\Big(a_{p'} - a_{l(s)}\Big)}.\end{aligned}$$

Note that if $f_s^p = 0$ for all $p \in P_s^I, p \neq p'$ then this equation is equivalent to (7.2.9) in Section 7.2.2 except for the fact that path $p'$ may not be the path which consists of physical links only, but may be a combination of logical and physical links. This is then a choice between two paths, the logical and path $p' \in P_s^I$.

Using approximation (7.4.13), which causes the marginal costs to become fixed costs, as they are no longer a function of $\boldsymbol{\theta}$, gives

$$\theta_s^{p'} = \sum_{p\in P_s^I, p\neq p'} \theta_s^p Y_{s,p,p'}(\nu_s, \boldsymbol{E}) + X_{s,p'}(\nu_s, \boldsymbol{E}) \qquad (8.1.10)$$

where

$$Y_{s,p,p'}(\nu_s, \boldsymbol{E}) = \frac{\frac{\partial \nu_s}{\partial \alpha_s}\left(a_p a_{p'}(F_p - F_{p'}) + a_{l(s)}a_p(F_{l(s)} - F_p) + a_{p'}a_{l(s)}(F_{p'} - F_{l(s)})\right)}{\nu_s\left(a_{p'} - a_{l(s)}\right)^2} - \frac{a_p - a_{l(s)}}{a_{p'} - a_{l(s)}}$$

and

$$X_{s,p'}(\nu_s, \boldsymbol{E}) = -\frac{\frac{\partial \nu_s}{\partial \alpha_s} a_{p'}a_{l(s)}(F_{p'} - F_{l(s)})}{\nu_s\left(a_{p'} - a_{l(s)}\right)^2} - \frac{a_{l(s)}}{\left(a_{p'} - a_{l(s)}\right)}.$$

In the case where the stream has only two paths it may use, that is, the physical and logical paths, equation (8.1.10) becomes equation (7.4.17) and hence, from Section 7.4 we know that in this case the optimal feasible $\theta_s^*$ is an end-point of the feasible region for $\theta_s$.

For streams $s \in S_I$, which have more than two possible paths, we could solve the equations (8.1.10) for $p \in P_s^I$ to find a formula for each $\theta_s^p$. This will be an explicit formula and hence we could substitute the $\boldsymbol{\theta}_s$ into the second derivative given by

$$\frac{\partial Z'}{\partial \theta_s^{p'}} = -\frac{\nu_s}{A_s^2}\left(a_{p'} - a_{l(s)}\right)\Bigg\{ \sum_{p \in P_s^I, p \neq p'} f_s^p\Big(a_p a_{p'}(M_p - M_{p'}) + a_{l(s)}a_p(M_{l(s)} - M_p) + a_{p'}a_{l(s)}(M_{p'} - M_{l(s)})\Big) - a_{p'}a_{l(s)}(M_{p'} - M_{l(s)})\Bigg\} \qquad (8.1.11)$$

to determine whether $\boldsymbol{\theta}_s$ maximises or minimises the objective function. Using Maple gives a solution for $\theta_s^p$ but it is complicated.

For a particular stream $s \in S_I$, for which $\eta_{\ell(s)} = 3$, we have

$$\theta^{p'} = \sum_{p \in P_s^I, p \neq p'} \theta^p Y_{p,p'} + X_{p'}$$

There are four possible paths for this streams. Let path 1 be the logical path, and the paths $p \in P_s^I$ be path 2, 3, and 4. Solving equations (8.1.10) using Maple gives

$$\theta_4 = \frac{Y_{23}Y_{32}X_4 - Y_{24}Y_{32}X_3 - Y_{34}Y_{23}X_2 - Y_{24}X_2 - Y_{34}X_3 - X_4}{W}$$

$$\begin{aligned}
\theta_3 &= \frac{Y_{24}Y_{42}X_3 - Y_{43}Y_{24}X_2 - Y_{23}Y_{42}X_4 - Y_{23}X_2 - Y_{43}X_4 - X_3}{W} \\
\theta_2 &= \frac{Y_{34}Y_{43}X_2 - Y_{32}Y_{43}X_4 - Y_{42}Y_{34}X_3 - Y_{32}X_3 - Y_{42}X_4 - X_2}{W} \\
W &= -1 + Y_{24}Y_{32}Y_{43} + Y_{34}Y_{23}Y_{42} + Y_{24}Y_{42} + Y_{23}Y_{32} + Y_{34}Y_{43}
\end{aligned}$$

Thus each $\theta_s^{p'}$ is in terms of the acceptance probabilities $a_p$ and the fixed costs $F_p$, and the choice of path will again be a trade off between these values.

Substituting the explicit equations for $\theta_s^p$ into the second derivative we could attempt to find whether these values maximise or minimise the objective function. However, it is unlikely that this will allow us to determine whether the traffic for a stream uses a combination of paths or a single path. Thus we will attempt to get further information by using a numerical example instead.

## 8.2 Example

We will again consider the network shown in Figure 6.1.1. As in Section 6.6, there are 9 nodes and 36 streams, of which 10 have only the direct physical path available. Some of the other 26 streams now have a larger number of paths from which to choose. There are 13 streams with 2 paths, 8 streams with 4 paths, 4 streams with 8 paths and 1 stream with 16 paths. The possible path choices can be seen in Appendix C. The logical path consists of the single logical direct link, and the physical path is the unique shortest possible path in terms of the number of links used. The other paths are a combination of physical and logical links.

To find the optimal tariffs, splitting probabilities and link blocking probabilities we have used the standard routine E04UCF available in the NAG Library of routines discussed earlier.

We have set the multiplexing coefficient to be $m = 1.0001$ and the cross-connecting coefficient to be $x = 2.0$. These values have been chosen to encourage calls on some streams to use their physical paths and on others their logical paths. Since the multiplexing coefficient always occurs in multiples of two, the two costs are approximately equal making neither path highly desirable. The costs are closer than in the example in Section 6.6 as both physical and logical links can now take advantage of the economy of scale trade-off.

The average stream blocking probability for all streams is bounded above by 0.01, with no lower bound imposed, giving $0 \leq B_s \leq 0.01, s \in S$. The tariff has not been bounded above or below, hence $0 \leq \alpha_s \leq \infty$. The number of circuits required by a stream, $s \in S$, using path $p \in P_s$ on link $j \in \mathcal{J}$ has been set to be zero or one; that is $\varsigma_{jp} = 0/1$.

We use the same traffic elasticity function as in the example in Section 6.6. This defines the tariff of the competitor $\tilde{\alpha_s}$, $s \in S$, to be 3, and thus the point of inflexion will be at $\alpha_s = 3$. The values of the base traffic demand, which is the traffic demand if the tariff is $\tilde{\alpha_s}$, are again as in Bean and Taylor. These demands are based on approximate subscriber figures and are given in Table 8.2.1.

| Streams $s \in S_I$ | Base Traffic | Opt. Traffic | Streams $s \in S_I$ | Base Traffic | Opt. Traffic | Streams $s \in S_P$ | Base Traffic | Opt. Traffic |
|---|---|---|---|---|---|---|---|---|
| CS-C | 475.0 | 63 | A-BH | 8603.4 | 1161 | C-S | 1744.0 | 887 |
| C-P | 660.0 | 88 | CS-PM | 5840.9 | 1530 | C-M | 2622.4 | 1336 |
| C-PM | 562.0 | 146 | A-B | 12556.8 | 1693 | B-CS | 8967.0 | 4589 |
| BH-C | 580.7 | 151 | BH-PM | 7138.0 | 1875 | A-P | 9776.6 | 5002 |
| A-C | 677.7 | 177 | B-BH | 10764.6 | 2829 | B-PM | 10419.7 | 5333 |
| B-C | 848.2 | 221 | P-S | 25086.7 | 3382 | PM-S | 21378.1 | 10958 |
| CS-P | 6858.0 | 242 | CS-M | 27147.1 | 3657 | BH-S | 22084.1 | 11321 |
| A-CS | 7040.3 | 486 | CS-S | 18081.2 | 4748 | B-S | 32191.0 | 16505 |
| P-PM | 8111.9 | 560 | A-S | 25750.9 | 6773 | A-M | 38632.8 | 19807 |
| BH-P | 8380.6 | 579 | M-PM | 32086.4 | 8437 | M-S | 98224.4 | 50390 |
| BH-CS | 6034.6 | 813 | BH-M | 33143.9 | 8716 | | | |
| B-P | 12232.0 | 845 | M-P | 37638.5 | 9894 | | | |
| A-PM | 8327.5 | 1123 | B-M | 48263.7 | 12695 | | | |

Table 8.2.1: Table of base traffic demands and optimal traffic demands.

Due to the extra complexity of the new formulation the NAG routine has difficulty in obtaining the optimal solution. It is possible to run the routine several times and obtain solutions in which the splitting probabilities differ greatly. In some solutions the splitting probabilities are non-integer, in others they are zero/one. It is interesting to note that if the set of links that is used by the paths is similar in two solutions, then whether the splitting probabilities are integer or not seems to have a relatively small influence on the objective function value. The set of links that is used appears to be more important than the splitting probability values.

To encourage NAG to find the optimal solution, it is possible to put tighter bounds on the tariffs to help guide the routine without disallowing the optimal solution. Using equation (8.1.8) it is possible to find an upper and lower bound on the value for the tariff $\alpha_s$ for any stream $s \in S$. These bounds can then be loosened to guarantee the optimal tariff will be within the chosen bounds. This guides the NAG routine enough to obtain optimal solutions or near optimal solutions. Using this approach it appears that the optimal splitting probabilities are zero/one, that is, only one path from the set of possible paths for a stream is used.

The optimal tariffs can be seen in Table 8.2.2. These tariffs are what we expect from equation (8.1.8). They consist of a term representing the cost of carrying a call and a term depending only on the elasticity function of the traffic $\nu_s(\alpha_s, \boldsymbol{E})$. As the multiplexing and cross-connecting coefficients are almost equal, the fixed costs for using any path for a stream are almost equivalent. The term depending on the elasticity function is equal to $-\nu_s \left(\frac{\partial \nu_s}{\partial \alpha_s}\right)^{-1} = 3$. Thus for $s \in S$, the expected optimal tariff, for this example, is approximately $3 + 2\eta_{l(s)}$.

Due to the large number of possible paths for some streams, and the fact that only one path for each stream had a non-zero splitting probability, only the path that is used is shown, rather than a list of splitting probabilities. The path is represented by the list of links that it uses, with logical links shown in bold.

The streams are in the same order as in the example in Section 6.6 and again are in the order of the magnitude of the accepted traffic for each stream.

| Stream $s \in S_I$ | Tariff $\alpha_s$ | Chosen path $\theta_s^{p'} = 1$ | Stream $s \in S_I$ | Tariff $\alpha_s$ | Chosen path $\theta_s^{p'} = 1$ | Stream $s \in S_P$ | Tariff $\alpha_s$ |
|---|---|---|---|---|---|---|---|
| CS-C | 9.03 | C-S **CS-S** | A-BH | 9.01 | A-M M-S BH-S | C-S | 5.03 |
| C-P | 9.02 | **M-P** C-M | CS-PM | 7.02 | B-PM B-CS | C-M | 5.02 |
| C-PM | 7.04 | C-S PM-S | A-B | 9.01 | A-M M-S B-S | B-CS | 5.01 |
| BH-C | 7.04 | C-S BH-S | BH-PM | 7.01 | BH-S PM-S | A-P | 5.01 |
| A-C | 7.02 | A-M C-M | B-BH | 7.01 | BH-S B-S | B-PM | 5.01 |
| B-C | 7.03 | B-S C-S | P-S | 9.01 | **M-P** M-S | PM-S | 5.00 |
| CS-P | 13.02 | **M-P** M-S **CS-S** | CS-M | 9.01 | M-S **CS-S** | BH-S | 5.00 |
| A-CS | 11.02 | A-M M-S **CS-S** | CS-S | 7.01 | **CS-S** | B-S | 5.00 |
| P-PM | 11.02 | **M-P** M-S PM-S | A-S | 7.01 | A-M M-S | A-M | 5.00 |
| BH-P | 11.01 | **M-P** M-S BH-S | M-PM | 7.01 | M-S PM-S | M-S | 5.00 |
| BH-CS | 9.01 | BH-S **CS-S** | BH-M | 7.01 | M-S BH-S | | |
| B-P | 11.01 | **M-P** M-S B-S | M-P | 7.01 | **M-P** | | |
| A-PM | 9.01 | A-M M-S PM-S | B-M | 7.01 | M-S B-S | | |

Table 8.2.2: Table of optimal tariffs and splitting probabilities.

In Table 8.2.2 there are only two logical links that are used : all the paths that are a combination of physical and logical links use these two logical links only and a selection of the physical links. The choice of the two logical links that are chosen is interesting. They are both from the extreme nodes of the network (Perth and Cairns) to a node further towards the center of the network (Melbourne and Sydney respectively). All the streams that have paths which can use these links do actually use the path that uses one or both of these links. Thus, the network becomes as in Figure 8.2.1, where the two logical links are represented by bolder lines, and the traffic is routed via the shortest path.

It is also interesting to note that the objective function value of the example in Section 6.6, for which there is no traffic mixing on the logical links, is $Z = 585978$ with $m = 1.01$ and $x = 2.0$. Running the same example, but setting $m = 1.0001$ and $x = 2.0$ gave an objective function value of $Z = 590462$. The example in this section, where traffic can mix on the logical links, gave an objective function value of $Z = 592567$ with $m = 1.0001$ and $x = 2.0$. Hence there is an increase in profit for the operating company caused by allowing traffic to mix on the logical links. Although it appears that this increase in profit is small, a large portion of the objective function value is fixed. This is due to streams $s \in S_P$ which only have one possible path but carry a large amount of traffic. Hence, this change in profit is significant.

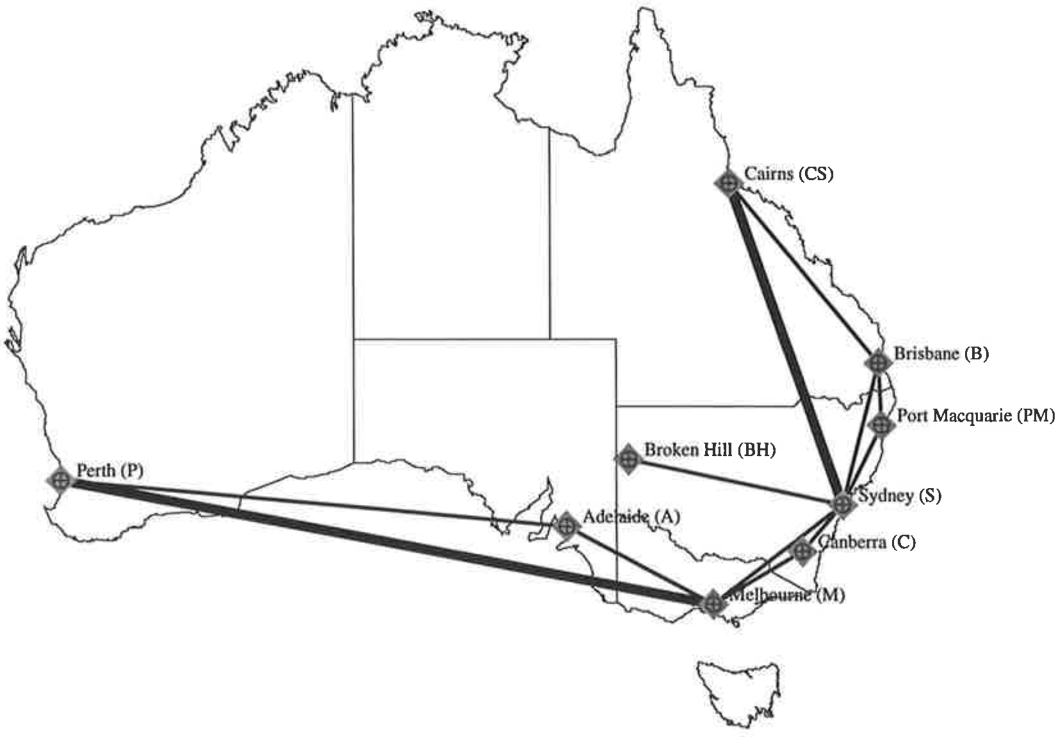


Figure 8.2.1: Optimal loss network incorporating logical links.

# Chapter 9

# Conclusions

In both parts of this thesis we were able to formulate general strategies for the design and optimisation of the network problems considered.

In Part I of the thesis we used three solution methods to solve the optimal communication spanning tree problem. We also applied these methods to the optimum requirement spanning tree, in which the costs are all equal to one, but the traffic demands are arbitrary. Using analytical and numerical approaches, we conjectured that, for random evenly distributed traffics, the optimal network will be a star in this case.

We then presented the results of testing the three solution methods, and combinations of the solution methods, on both Euclidean and non-Euclidean traffic. In general, for Euclidean traffic the best star solution appears to be approximately 7% higher than the optimal, or best known, solution. For non-Euclidean traffic, the best star solution is up to 800% higher than the optimal, or best known, solution.

It was noted that the performance of some of the solution methods varied greatly depending on the traffic type. The difficulty of choosing suitable parameters for the two random search techniques, simulated annealing and genetic algorithms was noted. For this problem it appears that the best solution method, if the traffic is Euclidean, is the genetic algorithm combined with the heuristic. If the traffic is non-Euclidean the simulated annealing algorithm in conjunction with the heuristic performs best. However, in both cases the simulated annealing by itself performs very well, and is a good choice as it is reasonably simple to find parameters and implement it.

In Part II of this thesis we presented a formulation for the determination of the optimal capacities, tariffs and splitting probabilities for a loss network in order to maximise the operating company's profit. We allowed the traffic arrival rate on a stream to depend on the tariff charged for that stream and the blocking probability of the links used by that stream. Our formulation used the Erlang fixed point approximation and used link blocking probabilities rather than link capacities as variables, which provided large computational savings. As a result we were able to solve problems of a realistic size.

By using this formulation we proved a result that explains the optimal tariff as the cost to the network of carrying the call plus a term depending only on the elasticity function. We also proved that, if the acceptance probabilities on the physical and logical paths are within the regulatory constraints set for the acceptance probabilities, then for any given stream either all calls will use the physical path or all calls will use the logical path. We also noted that if the quality of service requirements are similar for all streams, there is a threshold which is dependent on the magnitude of traffic on a stream, at which a stream switches from using its physical path to using its logical path.

Finally, we extended the problem formulation to allow a larger number of path choices. Traffic in this formulation was allowed to use a combination of logical and physical links. We again found a simple formula for the optimal tariff and a formula for the optimal, but not necessarily feasible, splitting probability. Using a numerical example, we were led to conjecture that the splitting probabilities in this formulation will again be zero/one, with only one path for each stream being used. In fact, in the chosen network design which incorporates logical links, the routing will be shortest path. This network design gave the network provider a higher profit than the two path formulation.

Although the NAG routine had some difficulty finding the optimal solution, it was possible to guide the routine's search by placing tighter bounds on the tariffs. The NAG routine was then able to find the optimal solutions for both formulations, thus proving to be a suitable solution technique.

# Appendix A

# Notation

| | | |
|---|---|---|
| $N$ | | Set of nodes. |
| $\mathcal{J}$ | | Set of links. |
| $\mathcal{J}_P$ | | Set of physical links. |
| $\mathcal{J}_L$ | | Set of logical links. |
| $S$ | | Set of all OD pairs. |
| $S_P$ | | Set of OD pairs connected by a single physical link. |
| $S_I$ | | Set of OD pairs not connected by a single physical link. |
| $P$ | | Set of all paths. |
| $P_P$ | | Set of paths which consist of a single physical link. |
| $P_L$ | | Set of paths which consist a single logical link. |
| $P_I$ | | Set of paths which consist of more than one link. |
| $P_s$ | $s \in S$ | Set of paths that stream $s$ may use. |
| $P_s^I$ | $s \in S_I$ | Set of paths stream $s$ may use from the set $P_I$; $P_s^I \equiv P_s \cap P_I$. |
| $\varsigma_{jp}$ | $j \in \mathcal{J},\ p \in P$ | Number of circuits path $p$ requires on link $j$. |
| $C_j$ | $j \in \mathcal{J}$ | Number of circuits on link $j$. |
| $E_j$ | $j \in \mathcal{J}$ | Link blocking probability on link $j$. |

| | | |
|---|---|---|
| $m$ | | The cost per circuit of multiplexing/de-multiplexing. |
| $x$ | | The cost per circuit of cross-connecting. |
| $\alpha_s$ | $s \in S$ | Tariff for stream $s$ per unit of traffic sent. |
| $\nu_s$ | $s \in S$ | Arrival rate to stream $s$. |
| $\theta_s$ | $s \in S_I$ | Probability that stream $s$ uses its physical path. |
| $\rho_j$ | $j \in \mathcal{J}$ | Reduced load on link $j$. |
| $\eta_j$ | $j \in \mathcal{J}_L$ | No of physical links on which logical link $i$ uses capacity. |
| $a_p$ | $p \in P$ | Acceptance probability on path $p$. |
| $b_p$ | $p \in P$ | Blocking probability on path $p$. |
| $A_s$ | $s \in S_I$ | Acceptance probability of calls into the network for stream $s$ averaged over all paths $p \in P_s$. |
| $B_s$ | $s \in S$ | Blocking probability for stream $s$ averaged over all paths $p \in P_s$. |
| $d(s)$ | $s \in S_P$ | Unique direct physical path for stream $s$. |
| $\delta(s)$ | $s \in S_P$ | Physical link which forms unique direct physical path for stream $s$. |
| $l(s)$ | $s \in S_I$ | Unique logical direct path for stream $s$. |
| $\ell(s)$ | $s \in S_I$ | Logical link which forms unique logical direct path for stream $s$. |
| $p(s)$ | $s \in S_I$ | Unique physical path for stream $s$. |
| $\Upsilon_s$ | $s \in S_I$ | Vertical asymptote of objective function at $\theta_s = \frac{-a_{l(s)}}{a_{p(s)}-a_{l(s)}}$ giving $A_s = 0$. |
| $M_p$ | $p \in P$ | Cost of marginal extra capacity required to support extra traffic on path $p$. |
| $F_p$ | $p \in P$ | Fixed cost of path p with approximation $\frac{\partial C}{\partial \rho} = 1 - E$. |
| $\theta_s^p$ | $s \in S_I,\ p \in P_s^I$ | Probability that stream $s$ uses path $p$. |

# Appendix B

# Derivative of $C(\rho, E)$

The derivative of $E$, given by equation (6.4.24), with respect to $C$ found using Maple is now presented.

$$
\begin{aligned}
\frac{\partial E}{\partial C} \;=\; & 2\left(ln\left(\frac{\rho}{C+1}\right) - \frac{C}{C+1} + 1 + \frac{12}{(12(C+1))^2} - \frac{3}{360}\frac{1}{(C+1)^4}\right)H \\
& \frac{(1+p_8(d_1+p_8p_{12}))^{16}}{(6.283185307(C+1))^{\frac{1}{2}}} - 6.283185307\frac{H(1+p_8(d_1+p_8p_{12}))^{16}}{(6.283185307(C+1))^{1.5}} \\
& + \frac{32H(1+p_8(d_1+p_8p_{12}))^{15}}{(6.283185307(C+1))^{\frac{1}{2}}}\Bigg(p_{10}(d_1+p_8p_{12}) + p_8\Bigg(p_{10}p_{12} + p_8 \\
& \Bigg(p_{10}p_{11} + p_8\Bigg(p_{10}(d_4+p_8p_9) + p_8\Bigg(p_{10}p_9 + p_8\Bigg(0.000016149p_2(C+1)^{\frac{1}{2}} \\
& +0.80745\times 10^{-5}\frac{p_1}{(C+1)^{\frac{1}{2}}} + 0.1993703704\times 10^{-6}\frac{p_7}{C^2} \\
& -0.1993703704\times 10^{-6}p_6\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)
\end{aligned}
$$

where

$$H = \exp\left(\mathrm{C}\ \ln\left(\frac{\rho}{C+1}\right) - \rho + C + 1 - \frac{1}{12(C+1)} + 0.00277777778\frac{1}{(C+1)^3}\right)$$

and

$$
\begin{aligned}
d_1 &= 0.049867347 \\
d_2 &= 0.0211410061
\end{aligned}
$$

$$
\begin{aligned}
d_3 &= 0.0032776263 \\
d_4 &= 0.0000380036 \\
d_5 &= 0.0000488906 \\
d_6 &= 0.0000053830
\end{aligned}
$$

and

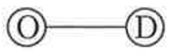
$$
\begin{aligned}
p_1 &= \left(\frac{\rho}{C+1}\right)^{\frac{1}{3}} + \frac{1}{9(C+1)} - 1 \\
p_2 &= -\frac{1}{3}\frac{\rho^{\frac{1}{3}}}{(C+1)^{\frac{4}{3}}} - \frac{9}{(9(C+1))^2} \\
p_3 &= 18p_1p_2(C+1) + 9p_1^2 \\
p_4 &= 9p_1^2(C+1) - 3 \\
p_5 &= 9p_1^2(C+1) - 1 \\
p_6 &= \left(\frac{2}{3}\frac{p_3}{C^{\frac{1}{2}}} - \frac{1}{3}\frac{p_5}{C^{\frac{3}{2}}} - 0.75p_2p_4(C+1)^{\frac{1}{2}} - 0.375\frac{p_1p_4}{(C+1)^{\frac{1}{2}}} - 0.75p_1p_3(C+1)^{\frac{1}{2}}\right)/C \\
p_7 &= \frac{2}{3}\frac{p_5}{C^{\frac{1}{2}}} - 0.75p_1p_4(C+1)^{\frac{1}{2}} \\
p_8 &= 3p_1(C+1)^{\frac{1}{2}} - \frac{37}{999}\frac{p_7}{C} \\
p_9 &= d_5 + 1.6149\times10^{-5}p_1(C+1)^{\frac{1}{2}} - 0.1993703704\times10^{-6}\frac{p_7}{C} \\
p_{10} &= 3p_2(C+1)^{\frac{1}{2}} + 1.5\frac{p_1}{(C+1)^{\frac{1}{2}}} + \frac{37}{999}\frac{p_7}{C^2} \\
p_{11} &= d_3 + p_8(d_4 + p_8p_9) \\
p_{12} &= d_2 + p_8p_{11}
\end{aligned}
$$

# Appendix C

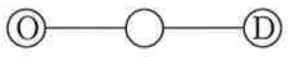
# Possible paths for multi-path formulation

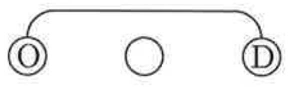
In Chapter 8, streams were allowed to use any combination of physical and logical links. All possible paths for streams with $\eta_{\ell(s)}$ links in the physical path are shown. A straight line represents a physical link, a curved line represents a logical link.

Paths for streams with $\eta_{\ell(s)} = 1$

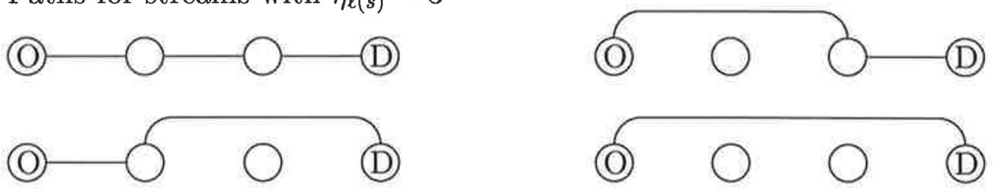


Paths for streams with $\eta_{\ell(s)} = 2$

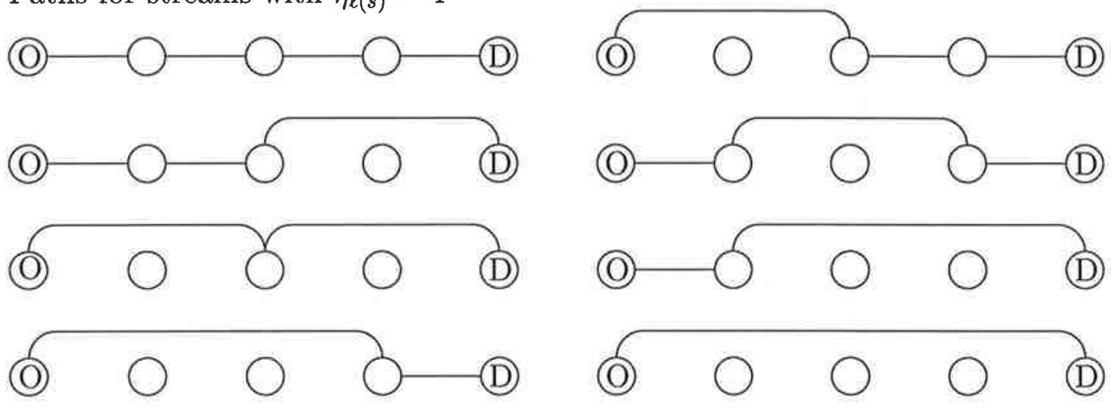




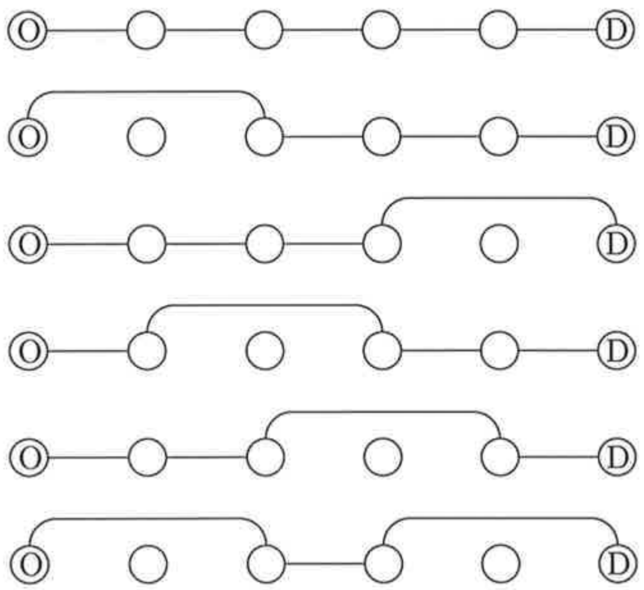
Paths for streams with $\eta_{\ell(s)} = 3$



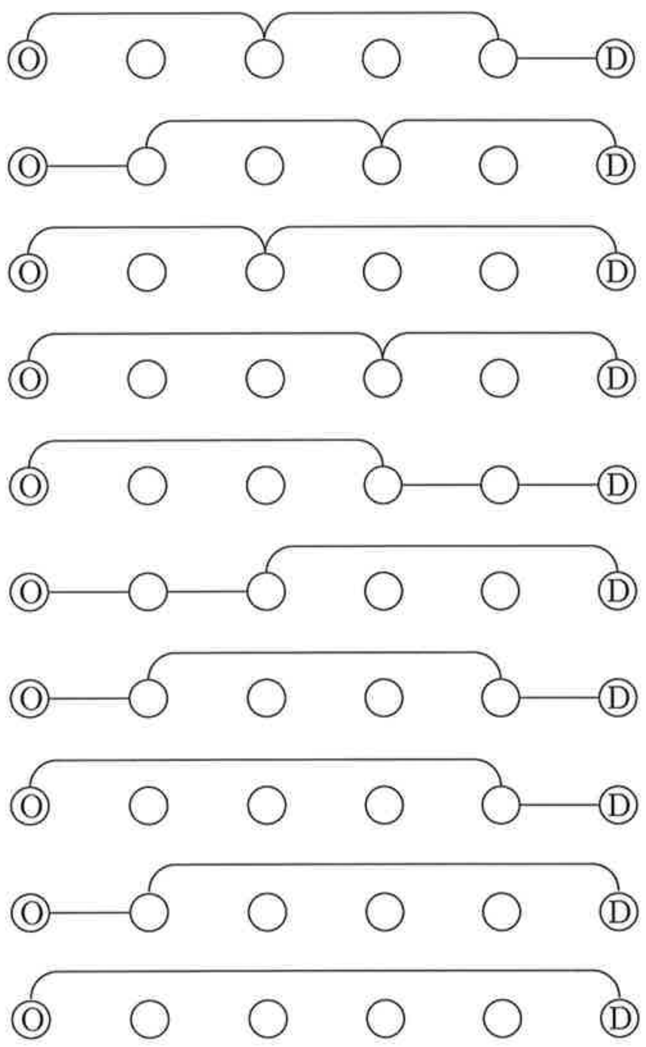
Paths for streams with $\eta_{\ell(s)} = 4$



Paths for streams with $\eta_{\ell(s)} = 5$



Continued on next page


O
D

# Bibliography

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: a Stochastic Approach to Combinatorial Optimization and Neural Computing.* Wiley, 1987.

[2] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions.* Dover Publications, New York, 1964.

[3] D.H. Ackley. An empirical study of bit vector function optimization. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 13, pages 170–205. Pitman, London, 1987.

[4] S. Agarwal, A. K. Mittal, and P. Sharma. Constrained optimum communication trees and sensitivity analysis. *SIAM Journal of Computing*, 13(2):315–328, 1984.

[5] R. K. Ahuja and V. V. S. Murty. Exact and heuristic algorithms for the optimum communication spanning tree problem. *Transportation Science*, 21(3):163–170, August 1987.

[6] H. Akimaru and T. Nishimura. The derivatives of Erlang's B Formula. *Review of the Electrical Communication Laboratory*, 11(9–10):428–445, 1963.

[7] D.J. Atkinson and G.J. Anido. Constant time approximation for the bandwidth requirements of multirate CBR connections. In *Australian Telecommunication Networks and Applications Conference*, volume 11, pages 235–240, 1996.

[8] J.E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their Applications*, pages 101–111, 1985.

[9] N.G. Bean, D.R. Brown, and P.G. Taylor. Maximal profit dimensioning and tariffing of loss networks with cross-connects. *To appear in Mathematical and Computer Modelling*, 1997.

[10] N.G. Bean and P.G. Taylor. Maximal profit dimensioning and tariffing of loss networks. *Probability in the Engineering and Informational Sciences*, 9:323–340, 1995.

[11] S.A. Berezner, A.E. Krzesinski, and P.G. Taylor. On the inverse of Erlang's function. *To appear in Journal of Applied Probability*, 1997.

[12] H. Braun. On solving travelling salesman problems by genetic algorithms. In H.P. Schwefel and R. Manner, editors, *Parallel Problem Solving from Nature, 1st Workshop in Lecture Notes in Computer Science*, pages 129–133. Springer-Verlang, October 1990.

[13] D. R. Brown, F. J. M. Salzborn, and K. White. The communication spanning tree problem : Comparison of an heuristic, search methods and lower bounding. In D. L. Hoffman, editor, *ASOR The 13th National Conference*, pages 15–42, September 1995.

[14] P. M. Camerini, L. Fratta, and F. Maffioli. Some results on the design of tree-structured communications networks. In *Proceedings of the Ninth International Teletraffic Congress*, 1979.

[15] L. Davis. Adapting operator probabilities in genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, June 1989.

[16] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[17] L. Davis and M. Steenstrup. Genetic algorithms and simulated annealing : An overview. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 1, pages 1–11. Pitman, London, 1987.

[18] L. Davis (ed). *Genetic Algorithms and Simulated Annealing.* Pitman, London, 1987.

[19] K.A. De Jong and W.M. Spears. An analysis of the interacting roles of population size and crossover in genetic algorithms. In H.P. Schwefel and R. Manner, editors, *Parallel Problem Solving from Nature, 1st Workshop in Lecture Notes in Computer Science*, pages 38–47. Springer-Verlang, October 1990.

[20] R. Dionne and M. Florian. Exact and approximate algorithms for optimal network design. *Networks*, 9:37–59, 1979.

[21] C. Ersoy and S. S. Panwar. Topological design of interconnected LAN/MAN networks. *IEEE Journal on Selected Areas in Communications*, 11:1172–1182, 1993.

[22] L.J. Eshelman, R.A. Caruana, and J.D. Schaffer. Biases in the crossover landscape. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 10–19. Morgan Kaufmann, June 1989.

[23] R.F. Farmer and I. Kaufman. On the numerical evaluation of some basic traffic formulae. *Networks*, 8:153–186, 1978.

[24] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[25] H.N. Gabow. A good algorithm for smallest spanning trees with a degree constraint. *Networks*, 8:201–208, 1978.

[26] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, 1984.

[27] A. Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, New York, 1985.

[28] P.E. Gill, S.J. Hammarling, W. Murray, M.A. Saunders, and M.H. Wright. User's guide for LSSOL (Version 1.0). Report SOL 86-1, Department of Operations Research, Standford University, 1986.

[29] A. Girard. *Routing and Dimensioning in Circuit Switched Networks.* Addison Wesley, Reading, Massachusetts, 1990.

[30] D.E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, Reading, Massachusetts, 1989.

[31] J. J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 4, pages 42–60. Pitman, London, 1987.

[32] B. Hajek. Cooling schedules for optimal annealing. *Math. Operat. Res.*, 13:311–329, 1988.

[33] J. Hesser, R. Männer, and O. Stucky. Optimization of Steiner trees using genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 231–236. Morgan Kaufmann, June 1989.

[34] J.H. Holland. Hierarchical descriptions of universal spaces and adaptive systems. Technical Report ORA Projects 01252 and 08226, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1968.

[35] J.H. Holland. Adaptation in natural and artificial systems. Technical report, University of Michigan Press,Ann Arbor, MI, 1975.

[36] T.C. Hu. Optimum communication spanning trees. *SIAM Journal of Computing*, 3:188–195, 1974.

[37] H. Inayoshi and B. Manderick. The weighted graph bi-partitioning problem : A look at GA performance. In H.P. Davidor, Y. Schwefel and R. Manner, editors, *Parallel Problem Solving from Nature, 3rd Workshop in Lecture Notes in Computer Science.*, pages 617–625. Springer-Verlang, October 1994.

[38] D.L. Jagerman. Some properties of the Erlang loss function. *Bell System Technical Journal*, 53:525–551, 1974.

[39] P. Jog, J.Y. Suh, and D. Van Gucht. The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 110–115. Morgan Kaufmann, June 1989.

[40] D.S. Johnson, J.K. Lenstra, and A.H. Rinnooy Kan. The complexity of the network design problem. *Networks*, 8:279–286, 1978.

[41] B.A. Julstrom. A genetic algorithm for the rectilinear Steiner problem. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 474–479. Morgan Kaufmann, July 1993.

[42] J.S. Kaufman. Blocking in a shared resource environment. *IEEE Transactions on Communications*, 29(10):1474–1481, 1981.

[43] F.P. Kelly. Routing in circuit-switched networks: Optimization, shadow prices and decentralization. *Advances in Applied Probability*, 20:112–144, 1988.

[44] F.P. Kelly. Loss networks. *The Annals of Applied Probability*, 1:319–378, 1991.

[45] A. Kershenbaum. Computing capacitated minimal spanning trees efficiently. *Networks*, 4:299–310, 1974.

[46] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[47] J. Lienig and H. Brandt. An evolutionary algorithm for the routing of multichip modules. In H.P. Davidor, Y. Schwefel and R. Manner, editors, *Parallel Problem Solving from Nature, 3rd Workshop in Lecture Notes in Computer Science.*, pages 588–597. Springer-Verlang, October 1994.

[48] G. Louth, M. Mitzenmacher, and F.P. Kelly. Computational complexity of loss networks. *Theoretical Computer Science*, 125:45–59, 1994.

[49] T. L. Magnanti and L. A. Wolsey. *Optimal Trees*, volume 7 of *Handbooks in OR and MS*, chapter 9, pages 503–615. Elsevier Science B.V., 1995.

[50] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[51] Y. Myung, C. Lee, and D. Tcha. On the generalized minimum spanning tree problem. *Networks*, 26:231–241, 1995.

[52] The Numerical Algorithms Group Limited. *The NAG Fortran Library Manual*, Mark 15 edition, 1991.

[53] R.H.J.M. Otten and L.P.P.P. van Ginneken. *The Annealing Algorithm.* The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1989.

[54] C.C. Palmer and A. Kershenbaum. An approach to a problem in network design using genetic algorithms. *Networks*, 26:151–163, 1995.

[55] C.H. Papadimitriou. The complexity of the capacitated tree problem. *Networks*, 8:217–230, 1978.

[56] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C - The Art of Scientific Computing*, chapter 10, pages 444–455. Cambridge University Press, second edition, 1994.

[57] F.J.M. Salzborn. The communication spanning tree problem: an heuristic algorithm. *Telecommunication Services for Developing Economies, Filipiak, J. (Ed.), Elsevier*, pages 199–206, 1991.

[58] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 51–60. Morgan Kaufmann, June 1989.

[59] F. Schweitzer, W. Ebeling, H. Rosé, and O. Weiss. Network optimization using evolutionary strategies. In H-M. Voigt, W. Ebeling, I. Rechenberg, and H-P. Schwefel, editors, *International Conference on Evolutionary Computation - The 4th International Conference on Parallel Problem Solving from Nature*, pages 940–949, September 1996.

[60] B. Selman and G. Hirst. Parsing as an energy minimization problem. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 11, pages 141–155. Pitman, London, 1987.

[61] D. W. F. Standingford. *Optimal Lifting Surfaces.* PhD thesis, The University of Adelaide, 1997.

[62] G. Syswerda. Uniform crossover in genetic algortihms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, June 1989.

[63] D.S. Touretzky and G.E. Hinton. Pattern matching and variable binding in a stochastic neural network. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 12, pages 155–170. Pitman, London, 1987.

[64] G. von Laszewski and H. Mühlenbein. Partitioning a graph with a parallel genetic algorithm. In H.P. Schwefel and R. Manner, editors, *Parallel Problem Solving from Nature, 1st Workshop in Lecture Notes in Computer Science*, pages 165–169. Springer-Verlang, October 1990.

[65] G. Zhang and J. Indulska. A new heuristic for the optimum cost spanning tree networks design problem with link capacity constraints. In *Australian Telecommunication Networks and Applications Conference*, pages 241–246, 1996.

[66] I.B. Ziedins and F.P. Kelly. Limit theorems for loss networks with diverse routing. *Advances in Applied Probability*, 21:804–830, 1989.