ABSTRACT

Title of dissertation: Dynamic Reconfiguration of
Network Topology in Optical Networks

Pedram J. Fard
Doctor of Philosophy, 2007

Dissertation directed by: Professor Steven I. Marcus
Department of Electrical and Computer Engineering

Dynamic reconfiguration of the logical topology of a network is generally used to adjust the shape of the network to adhere to changes in traffic. In this thesis the logical topology reconfiguration problem for a WDM network is combined with the multi-path routing of traffic throughout the network. Changes to the logical topology are made using branch exchanges to minimize the disruption to the network traffic. A Multi-timescale Markov Decision Process (MMDP) model is used to capture the different frequencies of changes to the network topology (slow timescale) and the routing of traffic (fast timescale). Heuristic policies are developed for the slow and fast time scales and rollout is used to improve the performance of the heuristic. To measure the performance of the heuristic and rollout policies, the Model Reference Adaptive Search (MRAS) method is used to find reference topologies that are close to optimal for a given static traffic demand matrix. To implement the MRAS method a random graph generation algorithm is needed. To address this need, a very efficient random graph generation algorithm is developed that competes with existing algorithms in the literature.

Dynamic Reconfiguration of Network Topology in Optical Networks

by

Pedram J. Fard

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Professor Steven I. Marcus, Chair/Advisor
Professor Mark A. Shayman, Co-Advisor
Associate Professor Richard J. La, Co-Advisor
Professor André L. Tits
Associate Professor Subramanian Raghavan

Dedication

To my mother, Tahereh,

and the memory of my father, Iraj,

my first and most important teachers.

Acknowledgments

I wish to thank my advisor, Professor Steven Marcus, for guiding my research with great vision and patience. He helped me keep my focus through years of full-time work and part-time research. I am grateful to have had the opportunity to work with him and learn in the process.

Thanks to Professors Shayman and La for the guidance they provided during the weekly group meetings.

Thanks to Professor Nick Wormald for lengthy and very helpful discussions we had via email about the literature in the graph enumeration and random graph generation.

Finally, I wish to thank my wife Behnaz for her hard work that helped my focus on my studies, and my children Armita and Arman for allowing me to work on my research in the evenings and during the weekends. Daddy is back!

Table of Contents

List of Figures

List of Abbreviations

BE	Branch Exchange
CE	Cross Entropy, or Certainty Equivalence
GTST	Generalized Traveling Salesperson problem
LSP	Label Switched Path
LSR	Label Switch Router
MRAS	Model Reference Adaptive Search
MMDP	Multi-time scale Markov Decision process
MPLS	Multi-Protocol Label Switching
OEO	Optical-Electronic-Optical
OLP	Optical lightpath
OXC	Optical Cross Connect
S-D	Source-Destination
TSP	Traveling Salesperson problem
WDM	Wavelength Division Multiplexing

List of Parameters

$(i-j)$	S-D pair with end nodes i and j
(p)	S-D pair with ID p
$[i-j]$	lightpath connecting nodes i and j
$[p]$	lightpaht with ID p
α	Cost coefficient for a branch exchange
$\varphi^i_{g,p}$	The ith LSP of S-D pair (p), given topology g
$\rho^i_{g,p}$	The reservation made for LSP $\varphi^i_{g,p}$
θ	Traffic vector
Θ	Set of all possible traffic vectors
δ^f	Initialization function for the fast time scale
δ^s	Initialization function for the slow time scale
γ	Discount factor
A	Action space at the fast time scale
a	Action at the fast time scale
$B(g)$	Set of admissible branch exchanges for topology g
b	The action at the slow time scale, or a branch exchange
C	Capacity of a lightpath
c	Cross parameter that identifies the two types of branch exchange
d	degree of each node
D_p	Dropped traffic of node pair p
g	A topology
G	State space at the slow time scale
G^d_N	Set of all valid topologies with N nodes and d interfaces per node
L	A lightpath
$L(g)$	set of all available LSPs given topology g
N	Number of nodes in a network
P	Number of S-D pairs in a network
Q	Number of lightpaths in a topology
S	Number of LSPs considered for each S-D pair
T	Number of fast time scale steps in a slow time scale step
u_q	Utilization of lightpath q
Y_t	Deterministic part of traffic vector at time t
Z_t	Random part of traffic vector at time t

Chapter 1

Introduction

Optimizing the topology of a network has long been a difficult problem to solve. In its most simplified form, the problem reduces to searching graphs with certain properties for one that optimizes a predefined cost function. This is an NP-complete problem (see [13]) and when the number of nodes in the network becomes large, searching this space becomes prohibitive. This problem has applications in many areas of science and technology, including but not limited to various communication networks. More recently, topology/graph search algorithms have been used for social and biological networks.

The preliminary research that culminated in this thesis started from a topology optimization problem for a wavelength division multiplexing (WDM) network with dynamic traffic, very similar to the one studied in [43]. One obstacle we faced in representing our results was that we did not have any reference to compare our solutions to. We were able to resolve this issue by implementing the MRAS method ([20]), which is used to find near optimal solutions for static optimization problems.

In order to apply the MRAS method to our topology optimization problem, we needed a random topology generation algorithm. Hence we developed a random topology/graph generation algorithm which turned out to be one of the most efficient compared to similar methods that are in use today.

The rest of this chapter is organized as follows: In Sections 1.1 to 1.3, the WDM network topology design problem is introduced and the MMDP model used in this thesis is compared to other papers in the literature. In Section 1.4 the MRAS method is introduced as the chosen method in this thesis to generate reference solutions for the static topology optimization problem. In Section 1.5 a survey of the random graph generation algorithms in the literature is given. In Section 1.6 the contributions of this thesis to the literature are pointed out. In the conclusion of this chapter we give an outline of the rest of this thesis.

1.1 Motivation

There are two distinct IP/WDM networking architectures: IP over reconfigurable WDM and IP over switched WDM. Under the IP over reconfigurable WDM architecture, the WDM lightpath topology is reconfigurable in response to traffic changes and/or network planning. In the IP over switched WDM architecture, optical headers or labels are attached and transmitted with the payload, and processed at each network switch.

Most papers in the literature concentrate on either WDM reconfigurability or a fixed WDM network with multi-path routing using Multi-Protocol Label Switching (MPLS). In this thesis we consider a combination of network engineering where the WDM reconfigurations are used along with the label switching (MPLS) to route the traffic. The optical topology reconfigurations occur at a much slower pace than the label switching decisions that are made to route the traffic entering the network.

An MMDP is a hierarchically structured sequential decision making process, where decisions in each level of the M-level hierarchy are made in different time scales. This hierarchy, which is deduced from the natural structure of the model, is used to reduce the size of the action space at each level. Instead of searching the action space of all M levels at every step, we only search the action space of one level of the structure at each decision time.

A usual approach to multi-level structured problems is that a slow time-scale subsystem abstracts the details of a fast time-scale dynamics by "average" behavior and then solves its own optimization problem (e.g., pp. 314-344 of [10]). This approach makes sense especially when the hierarchy in the system is structured in a pyramid sense. Our model is based on the same concept and the main difference is that our model is in discrete time and has some added complexities to mimic the specific nature of the networking model.

In this model, the state space and the action space of each level in the hierarchy are non-overlapping with those of the other levels and the hierarchy is structured in a pyramid sense such that a decision made at level m (slower time scale) will affect the evolutionary decision making process of the lower level $m+1$ (faster time scale); however, the lower level decisions do not affect the higher level's transition dynamics. The performance produced by the lower level's decisions will affect the higher level's decisions. A hierarchical objective function is defined such that the finite-horizon cost of following a non-stationary policy at the level $m+1$ over a decision epoch of the level m, plus an immediate cost at the level m, is the single step cost for the level m decision making process.

Optimality of a network topology in the presence of static traffic is easy to define and even though it becomes an NP hard problem to solve, comparing two topologies is straightforward given the cost function. When the traffic in the network is dynamic the concept of optimization becomes much more difficult to define. While we could find the optimum topology for the instantaneous traffic at any time, it is not possible to try to do a complete topology reconfiguration for every instance of the traffic demand.

The use of MDPs aims at finding policies that would transition through sequences of states and lead to better performance rather than finding the absolute best state for static conditions. The cost function is affected by the current state of the network (including the current topology and current traffic in the network) and an action set that is defined based on the state and the policy that selects the actions. The action set could be custom made to fit the behavior of a specific network if special properties exist for the network.

1.2 Our network model

We use an MPLS (Multi-Protocol Label Switching) over WDM (Wavelength Division Multiplexing) infrastructure. Each node consists of an optical cross connect (OXC) and a label switched router (LSR). This model extends to the case where some nodes only include OXCs with no LSR.

The lightpaths (optical channels) set up between the nodes form the logical topology of the network. In the logical topology, a lightpath is shown as a direct

connection between the two end nodes no matter how many OXCs the optical signal goes through to reach the destination node. Each node has a limited number of interfaces that are capable of electronic to optical and optical to electronic (OEO) conversion and traffic grooming. Each interface can handle one channel, and each channel is associated with one wavelength.

In order to reduce the disruption of traffic due to reconfiguration we consider branch exchanges (BEs) only. A branch exchange is a small change in the configuration, where two lightpaths are removed to free up interfaces that are used to set up two different lightpaths. We assume that there are as many wavelengths as necessary and if an interface is available a lightpath can be set up. In other words, the number of interfaces is the limiting factor, not the number of wavelengths. This work could be extended to the case where either the number of wavelengths or the number of interfaces is the limiting factor. Additional restrictions on the availability of wavelengths, or optical impairments will put additional restrictions on the possible BEs that can be carried out at a given topology. These restrictions can be reduced by optimizing the wavelength assignments in the physical network. We do not discuss this problem in this thesis but there are numerous papers in the literature in wavelength assignment optimization, for example refer to [4], [32], [65], [73].

Our goal is to accommodate the traffic demand between all Source-Destination (S-D) pairs while minimizing a cost function that is designed to minimize the utilization of the lightpaths and the dropped traffic. There are two problems that are considered simultaneously in this thesis. The first problem is finding the best distri-

bution of traffic among multiple paths available for each pair of nodes (multi-path routing problem). The second problem is concerned with finding the best transition through the topology space (using BEs), given the starting topology. Readjusting the distribution of the traffic among different paths is done more frequently than changing the logical topology, since it is less disruptive to the network traffic. For this reason the first problem is called the fast time scale problem and the second problem is the slow time scale problem. Using the MMDP model these two problems are solved simultaneously.

We study both static traffic and dynamic traffic. In the static traffic case we can compare the performance of our algorithm to other methods that attempt to find the optimal topology for a given traffic pattern. In the dynamic traffic case the optimal topology changes from step to step because the traffic changes with time. In this case it may not be possible to reconfigure the entire network to the optimal topology for the traffic pattern at each time step. This is because it may take several BEs to transition from the topology that is optimal for the traffic at a time step to the topology that is optimal for the traffic at the following step. Instead, we have to find a transition through several topologies that copes well with the transition in the network traffic. It becomes very difficult to even define an optimal transition, let alone solving such a problem. We use a Markov Decision Process (MDP) model to find a topology transition from an initial topology using the available BEs.

The MDP model employed in this thesis uses the current traffic measurements and the time-of-day patterns to predict the traffic demand in the near future and the future costs. Future look-ahead is needed mainly because we restrict the actions to

BEs. Since only one BE is allowed in each time step, to minimize disruption to the ongoing traffic, the action taken at time n not only determines the topology at time n, but also defines the possible topologies at $n+1$. This implies that a sequence of BEs selected by an algorithm may temporarily choose a logical topology that is not desirable for the current traffic demand as a transient topology in order to reach a good logical topology for the predicted demand.

A Multi-time Scale Markov Decision Process (MMDP) model [18] seems to provide a natural framework for this problem since we can distinguish several different types of actions that are taken with different frequencies. At the slow time scale we allow logical topology reconfigurations of the optical network, while at the fast time scale we allow the Label Switched Path (LSP) reconfigurations that happen more frequently than logical topology reconfigurations. It is possible to include more time scales to include physical topology changes at a slower time scale or online distribution of packets among different LSPs for a Source-Destination (S-D) pair at a faster time scale.

1.3 Survey of the topology reconfiguration literature

Topology reconfiguration in optical networks has been studied under different assumptions. In [66], [46], and [40], a total reconfiguration of the network is studied. A disadvantage of this approach is that a total reconfiguration causes traffic disruption during the transition period. In contrast, in the present work, we use small steps (BEs) to mitigate the adverse effects of a total topology reconfiguration.

In [38] the shortest path, i.e., the least number of BEs, to go from one topology to another is calculated. This is only useful if the target topology is known and there is no traffic disruption or cost considerations.

In [43] BEs are used for reconfiguration of the logical topology. They make the same assumption of no wavelength limitations as we do. However, there are two main differences between their approach and ours: In their approach, since no prediction of the traffic pattern is involved, the resulting policy is reactive as opposed to the proactive policy proposed here. Their dynamic traffic pattern seems to make a linear transition from one pattern to another, whereas in our traffic pattern we do not make any such assumption.

The problem of designing the logical topology for a known traffic pattern has been studied in [47], [54], and [37]. However, the exact traffic pattern is not known beforehand in practice. We define a traffic model that has two parts. The first part is deterministic and reflects the time-of-day variations in the traffic demand. This could be the average behavior of the network over many days. The second part is random and reflects the uncertainty of our prior knowledge of traffic, which is the day-to-day variation of traffic.

The algorithm given in [26] assumes that there are wavelengths and interfaces available so that it is possible to set up new lightpaths when needed and remove lightpaths when they are significantly under-utilized. In their approach they do not consider congestion in the network and assume they can insert a new lightpath whenever the network gets close to congestion. This approach works well when the network is not congested. However, if we assume we may eventually run out

of resources and the network becomes congested, this scheme does not work very well. If other related lightpaths are not lightly utilized (which is the case during congestion) the algorithm is unable to make any adjustments to the topology to reduce the congestion in the network.

Due to the complexity of the topology optimization problem ([13] shows it is NP-complete), there are a number of papers that use heuristic algorithms to control the reaction of the network to changes in traffic (e.g., [26], [61], [6], [5], [46]). In Chapter 2 we define a heuristic algorithm that with a greedy approach finds the BE that gives the best cost in the next step. This heuristic algorithm improves the topology until it reaches a topology that is locally optimal. We apply a rollout algorithm to this heuristic to improve the performance. In [10], rollout algorithms are defined as an approximation to the Dynamic Programming methodology that results in a near-optimal solution. Rollout algorithms have been used in various optimization problems to find near optimal solutions to complex problems(see [12], [72], [64]).

1.4 The Model Reference Adaptive Search algorithm

Finding a reference optimal solution to the problem has been a challenge in this thesis. In this section we introduce an algorithm that finds near optimal topologies for a given cost function.

The Model Reference Adaptive Search (MRAS, see [29], [20]) algorithm is a variation of Cross Entropy (CE) method ([22], [60]) that finds ϵ-optimal solutions

to static optimization problems. MRAS finds very good solutions to the traveling salesperson problem (TSP), and the similarities between TSP and the topology generation problem pointed us towards MRAS.

In the TSP a salesman is to travel through N cities going to each city only once and ending in the same city that he started from. There is a cost associated with traveling between each pair of cities. The objective of the TSP is to find the path for the traveling salesman that minimizes the total cost. This is equivalent to finding a minimum-cost 2-regular connected graph or a ring network with the cities as vertices of such a graph and the road between them as the edges.

MRAS finds solutions to this problem using a recursive method that in each iteration performs two steps. First it generates a sample population of random paths for the salesman according to a given probability distribution and calculates the cost for each path. Then the cost of each path is used to update the probability distribution used in the first step. The probability distribution is updated such that it gives higher probability to selecting sample paths that had better costs. Under very weak conditions this probability distribution converges almost surely to a degenerate probability distribution that generates the MRAS solution.

Our static topology optimization problem is similar to the TSP, in that it optimizes a graph, however we optimize a more general graph than in the TSP. This similarity prompted us to consider MRAS to generate a reference solution for our problem. In order to do that we need to define a probability distribution for graphs of given size and characteristics, and then generate random graphs according to the probability distribution. It should be noted the static topology optimization

problem is not the same as the generalized TSP (GTSP) problem introduced in [44]. In GTSP the objective of the salesperson is to visit clusters of cities, rather than each city.

Random graph generation is a rich topic in graph theory. According to Bollobás in [16] there are two basic models for generating random graphs. The first model generates graphs with uniform distribution, and the second model that is more interesting to us starts with N vertices and selects edges between vertices independently of the other edges with probability p. This model allows us to define a probability distribution for each edge but has a major problem. Generating a graph with given characteristics using this method is a hit and miss process.

For our topology optimization problem, we are interested in generating graphs with a given number of nodes and a given degree sequence (the sequence of numbers corresponding to the number of interfaces of each node) for those nodes. For example if we want to generate a topology with N nodes and all nodes having a degree of 3, this is equivalent to generating a 3-regular connected graph. Using the above basic model, the probability of generating a graph that is 3-regular is very low. As a result we have to generate many random graphs until one of them is 3-regular.

There are numerous papers in the random graph literature that improve the performance of this basic algorithm by increasing the probability of generating a graph with given degree sequence (see Section 1.5). However the probability of generating a graph not matching the given degree sequence is not eliminated. Most of the random graph generation algorithms are based on a counting algorithm that counts the total number of graphs with given properties.

In this thesis we introduce a new algorithm that generates random graphs with a given degree sequence. This algorithm generates a graph with the given degree sequence in every iteration with certainty. This improves the speed of the graph generation algorithm. This is critical for the speed of the MRAS algorithm since in each iteration of the algorithm we need to generate a large random sample of such graphs.

In order to implement the MRAS method we use the MMDP cost function to evaluate each topology and use our random graph generation algorithm to generate the sample population based on a probability distribution. The algorithm starts from an initial probability distribution, which is chosen to be the uniform distribution, and in each iteration finds a probability distribution that favors the topologies with better cost function. The objective of the algorithm is to reach a degenerate probability distribution (that generates the same topology with probability one). The topology corresponding to the degenerate probability distribution is the solution found by MRAS algorithm.

1.5 Survey of the graph theory literature

Enumeration of graphs has been studied for more than a century. In [27], Gropp gives evidence of work dating back to more than 100 years ago regarding enumeration of graphs. In their doctoral theses, Read (in [48]) and Wormald (in [67]) introduce a number of counting algorithms and formulae including recursive formulae for counting labeled cubic graphs and labeled connected cubic graphs.

They developed recurrence relations for these graphs and also a differential equation that is satisfied by the generating function of these graphs. There is a book ([28]) and there are numerous other papers in the literature for counting different kinds of graphs and graphs with special properties, see for example, [49], [50], [51], [53], [69], [68].

In [52], the authors introduce an algorithm that can count a 10-node graph with a given degree sequence. In this algorithm they sort the degree sequence of the nodes in the graph and start from the highest degree to assign neighbors to each node. Our counting algorithm given in Section 3.1 is very similar to this algorithm except that we sort the nodes in the opposite order and start from the lowest degree node. In this way, we show how to enumerate the number of *connected* graphs resulting from the given degree sequence.

The theory of random graphs was founded by Erdős and Rényi in [24]. As described in [16], there are two basic models in random graph theory, namely $\mathcal{G}(N, M)$ and $\mathcal{G}\{N, P(edge) = p\}$. The former starts with N nodes and selects M edges at random from all available edges. The latter starts with N nodes and assumes each possible edge is chosen with probability p independently of the selection of all the other edges. It is very easy to generate random graphs using these two models but the graphs generated using these two models do not have the same degree sequence, hence they are not relevant to the type of graphs that we intend to construct in this thesis.

Random d-regular graphs (each node of the graph having d interfaces, see [15]) were first generated implicitly in enumeration papers by Bollobas [14], Bender and

Canfield [9], and Wormald [67]. In [70], Wormald gave a sequential algorithm that would generate random graphs. Random d-regular graphs are often generated using a *pairing* model that starts with Nd points (Nd even) in N groups. It then chooses a random pairing of these points. Finally, an edge between nodes i and j is created if there is a pair that has points in groups i and j. In its very basic form, this model generates a large number of duplicate pairings that correspond to multiple edges between the same nodes. As a result, we have to obtain several pairings before one of them results in a valid graph and hence this is a very slow algorithm (run time of $\mathcal{O}(Nde^{(d^2-1)/4})$ per graph). In [62], there are references to several versions of this algorithm that give polynomial expected running time under very specific conditions. Such algorithms are often very hard to implement and much harder to analyze. In [34] and [62], the authors achieve a faster run time of $\mathcal{O}(Nd^2)$, by generating an approximately uniform distribution for regular graphs of N nodes and degree d. Wormald has an extensive survey of the random graph literature in [71]. Recently, a new method was introduced in [8], that can extend the algorithm in [62] to graphs with general degree sequence.

The algorithm introduced in Section 3.3 of this thesis takes a straightforward approach to generating topologies. In our algorithm, we select one node, i, at a time and select the connect subset of node i (i.e., the set of nodes which connect to it) in one shot. Given the starting degree sequence, there are a limited number of subsequences that a partially connected graph may have. We determine beforehand the number of possible graphs that could be built with all possible partial degree sequences, and we keep this information in a database. When building a random

graph, we know each selection of the connect subset of node i results in a residual degree sequence. Using the database, we lookup the number of possible graphs for each branch in our decision tree, we assign a weight to that branch. We select each branch with a probability proportional to its weight. We will show in Chapter 3 that our algorithm can generate graphs with runtime of $\mathcal{O}(Nd\log(d))$.

In summary, the advantages of this algorithm are as follows. It has a very fast running time of $\mathcal{O}(Nd\log(d))$, which is faster than all algorithms in the literature. Our algorithm generates samples that are exactly uniformly distributed (no approximation) and it generates one graph with each iteration with certainty. Furthermore, this algorithm lends itself to generating random graphs with a distribution that is a projection of a parameterized family of distributions. The latter property of this algorithm makes it suitable to be used for the MRAS method to find ϵ-optimal solutions to static topology optimization problems. Finally, our algorithm can be applied to a general degree sequence and is not limited to graphs with certain properties. The drawback of the algorithm is that for large networks the size of the database can grow large. In our simulations we generated graphs of up to 40 nodes.

1.6 Contributions to the literature

There are three different fields that this thesis has contributed to. These fields are random graph theory, static optimization, and topology optimization in optical networks. Here is a summary of these contributions.

- A very fast algorithm for generating random graphs with uniform distribution

with prescribed degree sequence. The algorithm introduced in this thesis is faster than any algorithm available in the literature. This algorithm in its most precise variation can generate samples with uniform distribution for graphs of moderate size.

- The traveling salesman problem (TSP) has been used to evaluate many optimization algorithms, including the CE and MRAS methods. The TSP is equivalent to optimizing a ring or a 2-regular graph for a given cost function. The random graph generation algorithm introduced in this thesis is extended to a variation that allows us to generate sample graphs that are concentrated on graphs with better cost. This variation of the random graph generation algorithm enables us to extend the MRAS and CE methods to more general graphs.

- We apply the MMDP method to the problem of topology optimization and apply the appropriate approximations to make it feasible to be used in real time network optimization for moderate sized networks.

- We use rollout to improve the performance of a greedy heuristic algorithm that uses branch exchanges in a WDM network.

1.7 Outline of Thesis

In Chapter 2, the details of the MMDP model and its application to the problem of reconfiguration of network topology is presented. A heuristic is defined

for each time scale and specifics of a rollout policy that improves the heuristic algorithm is given.

The objective of Chapter 3 is to define a reference solution that can be used to measure the goodness of solutions found by algorithms of the heuristic and rollout algorithms. The Model Reference Adaptive Search (MRAS) algorithm that is used to generate ϵ-optimal solutions to static optimization problems is used to generate the reference solution. In the absence of a tractable way to find the optimal solution for large networks the MRAS solution is very useful. Since MRAS is a generic algorithm, this dissertation provides the building blocks needed to implement this algorithm for the network topology design problem.

In Chapter 4, the numerical results for various simulations supporting the theoretical work in Chapters 2 and 3 is given.

Chapter 2

Staging the problem as an MMDP

In this chapter the two problems of multi-path routing and the topology optimization are respectively fit into fast and slow time scales of an Multi-timescale Markov Decision Process (MMDP). Heuristic algorithms have been provided for both fast and slow time scales as well as an exact solution for the fast time scale and an improved policy (using rollout) for the slow time scale.

2.1 Terminology and Assumptions

Consider N nodes in the MPLS/WDM network. Each node uses its interfaces and the underlying physical network to connect to other nodes using lightpaths. Each node is assumed to be an integrated node, which includes an optical cross connect and is capable of optical to electronic conversion. Each lightpath connects the two end nodes with optical-only signals. Source-destination (S-D) pairs that are not connected directly with a lightpath go through two or more lightpaths with OEO (optical-electronic-optical) conversion in between.

In this thesis we investigate the problem of designing the logical topology and are not concerned with the problem of optical routing and wavelength assignment in the physical (optical) infrastructure. Each node is allowed to set up lightpaths to d other nodes, where d is the number of available interfaces at each node. There

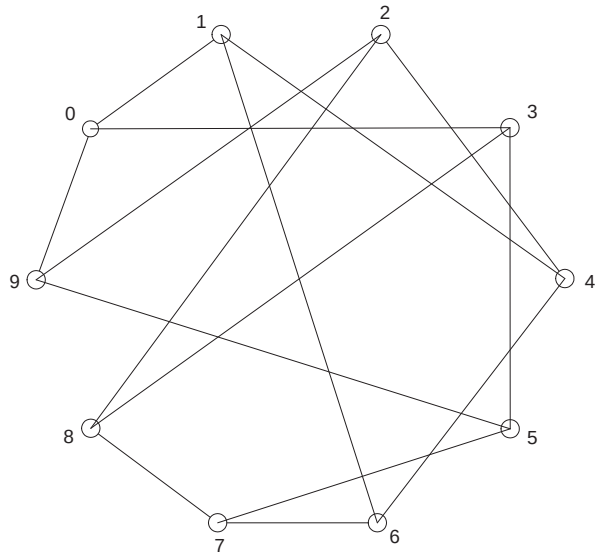

Figure 2.1: A sample topology for a 10 node network with a 3 interface limit for each node.

may be several paths (LSPs) connecting the nodes of an S-D pair. The paths could share the load of traffic demand for that S-D pair.

Reconfiguration of the logical topology is allowed only through BEs that are defined on two lightpaths. A BE is formed by swapping one of the two end nodes of a lightpath with one of the end nodes of another lightpath. Hence, each pair of lightpaths has two potential BEs. We assume the lightpaths are bidirectional and the source and destination of a lightpath are interchangeable. The interfaces include one transmitter and one receiver, and the traffic is assumed to be symmetric.

We make some assumptions about the number of interfaces available at each node and how we can form a valid topology by connecting the nodes to each other. Our definitions in the next section are based on these assumptions, which are specific to this problem but can be extended to more general types of network.

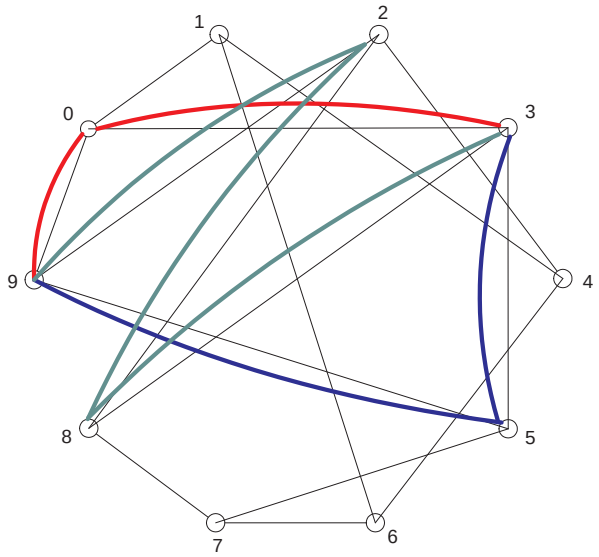

Figure 2.2: A sample topology with LSPs displayed for one S-D pair.

We use Label Switched Paths (LSP) to route the traffic from one end node to another. An LSP is a sequence of lightpaths that connects the end nodes of an S-D pair. To limit the search for the LSPs we always search for a fixed number of LSPs with least number of hops. In this study we search for best (least hops) 3 LSPs for each S-D pair unless otherwise stated. Figure 2.1 shows a sample topology of a 10-node network and Figure 2.2 shows three LSPs set up between nodes 9 and 3.

Figure 2.3 shows the tear down of two lightpaths and set up of two lightpaths that results in a branch exchange.

2.2 Model definition

We start this section by some definitions that set up the notation throughout the thesis.

Definition 2.1 *Consider two nodes i and j (where $i < j$) of an N node network.*

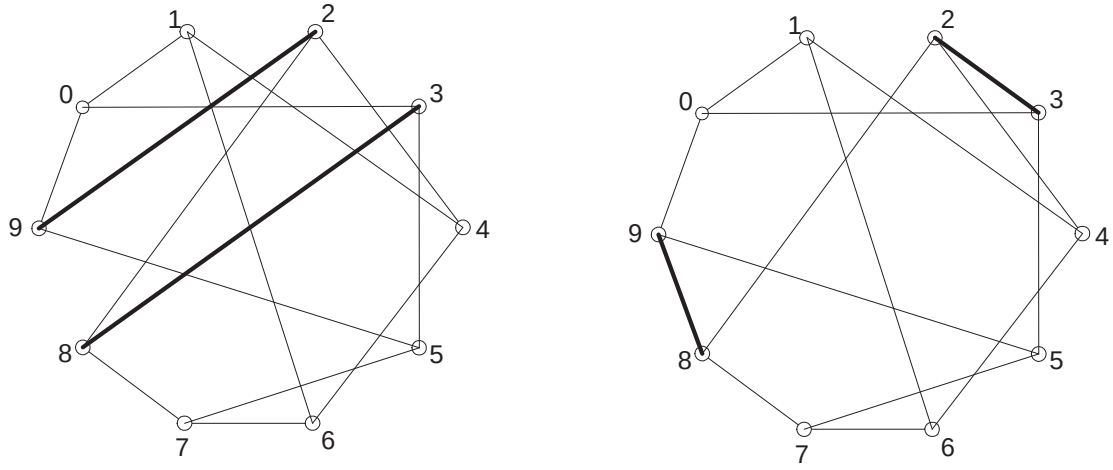

Figure 2.3: A sample branch exchange: lightpaths 2-9 and 3-8 are torn down and lightpaths 2-3 and 8-9 are set up for this branch exchange.

The **S-D pair** *corresponding to these two nodes is denoted by* $(i-j)$*. A unique ID* p *is defined for each S-D pair. Alternatively, we refer to an S-D pair using its ID,* (p)*.*

Definition 2.2 *A* **lightpath** *is set up between the end nodes of an S-D pair and forms a direct connection in the logical topology between the nodes of the corresponding S-D pair. A lightpath* L*, connecting nodes* i *and* j *is denoted by* $[i-j]$*. A unique ID,* p*, is defined for each lightpath, which is equal to the ID of the corresponding S-D pair. Alternatively, we refer to a lightpath using its ID,* $[p]$*. Each lightpath has a defined capacity denoted by* C*.*

If two nodes are not directly connected but there is a path that connects them, they are said to be indirectly connected. We also refer to fully connected nodes as the nodes that have used all their interfaces to set up lightpaths to other

S-D	S-D pair ID	S-D	S-D pair ID	S-D	S-D pair ID	S-D	S-D pair ID
(0-1)	0	(2-3)	17	(4-5)	30	(6-7)	39
...	...	...	...	...	...	(6-8)	40
(0-9)	8	(2-9)	23	(4-9)	34	(6-9)	41
(1-2)	9	(3-4)	24	(5-6)	35	(7-8)	42
...	...	...	...	...	...	(7-9)	43
(1-9)	16	(3-9)	29	(5-9)	38	(8-9)	44

Table 2.1: S-D pairs and their corresponding Lightpath IDs.

nodes. We also refer to fully connected network, that is a network in which all the nodes are fully connected. A partially connected network is a network with at least one partially connected node. We should also make a distinction between a fully connected network and a valid topology. A fully connected network is also a topology if all the nodes in the network are either directly or indirectly connected.

Lightpaths are assumed to be bidirectional, so there is no difference between lightpaths $[i - j]$ and $[j - i]$, but to have a unique representation for a lightpath we restrict the lightpath representation $[i - j]$ to have $i < j$. We use the same rule for the representation of S-D pairs. Each network with N nodes has $P = \binom{N}{2}$ S-D pairs. Table 2.1 shows S-D pair and S-D pair IDs for each S-D pair for the sample 10-node network of figure 2.1. Therefore, the range of lightpath IDs for each network is $0 \leq p \leq P - 1$.

As mentioned before, any set of lightpaths between the nodes of the network creates a logical topology but we do not consider all such logical topologies to be valid.

Definition 2.3 *Consider a network with N nodes, where each node has d available interfaces that are used to connect to other nodes of the network. A* **valid logical topology** *g for this network is defined to be a set of lightpaths, which meet the following criteria.*

1. *There should be at most one lightpath connecting a S-D pair, using one interface from each of the two end nodes.*
2. *There should be no node with any free interfaces.*
3. *There should be no disjoint subnetworks.*

Furthermore, we define set G_N^d to be the set of all valid logical topologies with N nodes and d interfaces per node.

Note that by the definitions of the S-D pair and lightpath, it is implied that there can be no lightpath between a node and itself.

Given the number of nodes, $N \geq 3$, and the number of available interfaces for each node, d ($2 \leq d \leq N-1$ and $N \cdot d$ even), there are $Q = N \cdot d/2$ lightpaths in each valid topology.

Definition 2.4 *Consider a network with N nodes and $P = \binom{N}{2}$ S-D pairs. For each valid logical topology for this network a unique* **Identification number** *or*

ID *is defined by a P-digit binary number, where the p-th most significant digit of this number is chosen to be 1 if the lightpath with ID of p is present in this logical topology. Otherwise, the p-th digit is set to 0.*

Definition 2.5 *A* **branch exchange** *is a triplet* (L_1, L_2, c)*, where* $L_1 = [i_1 - j_1]$ *and* $L_2 = [i_2 - j_2]$ *are lightpaths that will be torn down to form two new and different lightpaths* L_3 *and* L_4*, and* c *is a binary variable that differentiates between the two possible ways that* L_3 *and* L_4 *can be formed:* $c = 1$ *indicates that the resulting lightpaths are* $[i_1 - j_2]$ *and* $[i_2 - j_1]$*.* $c = 0$ *indicates that the resulting lightpaths are* $[i_1 - i_2]$ *and* $[j_1 - j_2]$*. Branch exchange* $b = (L_1, L_2, c)$ *is* **admissible** *for logical topology* $g \in G_N^d$ *if and only if*

1. $L_1, L_2 \in g$*, and*
2. $(g - \{L_1, L_2\}) \cup \{L_3, L_4\} \in G_N^d$.

In other words, the two lightpaths that are going to be removed should belong to the current topology g, and the resulting topology should be a valid one. Note that by the definition of a valid topology, and by item 2 above, it is implied that L_3 and L_4 should not belong to g. Because otherwise the resulting topology will have two lightpaths between a S-D pair.

Figure 2.4 displays the difference between the two types of branch exchanges that stem from the same two lightpaths. $c = 1$ corresponds to the type of branch exchange that looks like a cross.

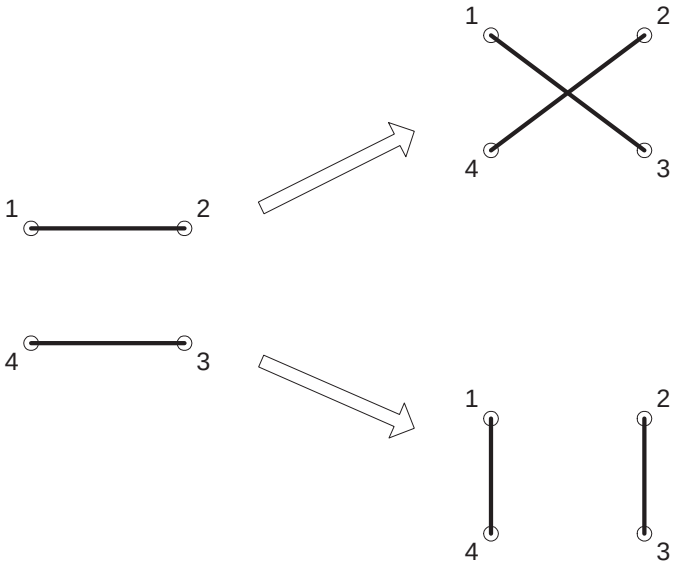

Figure 2.4: Two types of branch exchange resulting from tear down of a pair of lightpaths.

Definition 2.6 *Given a logical topology g, S-D pair $(p) = (j-k)$, a* **Label Switched Path** *or* **LSP** *φ_p represents a set of lightpaths that belong to the logical topology g and form a path between nodes j and k.*

It is possible to have multiple paths between an S-D pair. Assuming that there are N_p such paths, φ_p^i, $i = 0, \cdots, N_p - 1$ represents the ith LSP. A unique **identification number** for this LSP is defined by a P-digit binary number that has a 1 in the p-th position if $[p] \in \varphi_p^i$. Given a logical topology g and a S-D pair (p), index i of φ_p^i starts from 1 and is assigned to LSPs in the increasing order of their identification number.

Assumption 2.1 *The lightpaths forming an LSP must satisfy the following conditions.*

1. *There should be one and only one lightpath in φ_p^i that has j as one of its end nodes.*

2. *There should be one and only one lightpath in φ_p^i that has k as one of its end nodes.*

3. *Consider any node e_1 such that $[e_1 - e_2] \in \varphi_p^i$ or $[e_2 - e_1] \in \varphi_p^i$, if $e_1 \neq j$ and $e_1 \neq k$, then there should be one and only one other lightpath $[e_1 - e_3]$ or $[e_3 - e_1] \in \varphi_p^i$, $e_3 \neq e_2$.*

The conditions given above ensure each LSP to connect two nodes without creating a loop (i.e., visiting a node more than once along the way), or going through a lightpath more than once. Clearly, there is OEO conversion between the lightpaths along a multi-hop LSP. The smallest number of lightpaths in an LSP is 1, which is possible for S-D pairs that have direct connection to each other by a lightpath.

Property 2.1 *Given a logical topology g, there are $N \cdot d/2$ lightpaths. Since each LSP is a subset of lightpaths and the number of subsets of lightpaths are finite, there are a finite number of LSPs available for each S-D pair.*

Definition 2.7 *Each LSP, φ_p^i is associated with a number, ρ_p^i, called a* **reservation**. *This number signifies the amount of traffic that can be routed through that LSP.*

Note that the reservations are made for each LSP before the actual traffic assigned to the LSP is known. Any traffic that is attempted to be routed through an LSP in excess of the reservation is dropped.

Definition 2.8 *Consider a logical topology g and the corresponding LSPs. A* **traffic assignment** *is defined to be the set of reservations for every LSP for that logical topology with the following constraints.*

1. *The reservation for each LSP is a non-negative number.*

2. *The sum of reservations for the LSPs going through the same lightpath should not exceed the capacity of that lightpath.*

We follow the sample network given in Figure 2.1 throughout the study so the definitions are easier to follow. In this network $N = 10$ (number of nodes), and $d = 3$ (number of interfaces per node). The nodes are numbered from 0 to 9. There are 45 S-D pairs in this network that are numbered from 0 to 44. Out of the 45 S-D pairs, $\frac{N \cdot d}{2} = 15$ of them are connected directly by a lightpath. Table 2.1 defines the relation between the lightpath IDs and the corresponding S-D pairs. The LSPs shown in Figure 2.2 for S-D pair $(29) = (3-9)$ of the topology g_1 are referred to as $\varphi^1_{g_1,29} = \{2, 8\}$ with ID of 260 $(= 0 \cdots 0100000100)$, $\varphi^2_{g_1,29} = \{28, 22, 23\}$ with ID of 281018368, and $\varphi^3_{g_1,29} = \{25, 38\}$ with ID of 274911461376.

2.3 MMDP Model

Consider a two time scale MDP model as follows. There are two time scales associated with two levels of decision making. At the higher level (slow time scale), decisions about logical topology reconfiguration are made. The frequency of these decisions is on the order of once every few minutes. Each slow time scale step, denoted by n, consists of T fast time scale steps. At the fast time scale, reservations

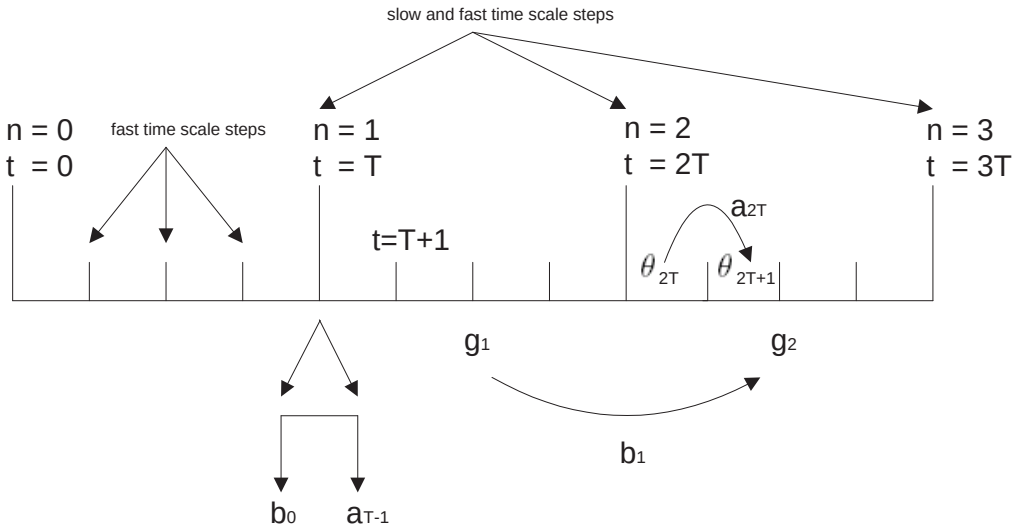

Figure 2.5: Graphical illustration of time evolution in the MMDP.

for the LSPs and/or the set of LSPs can be changed. Each fast time scale step is denoted by t. Figure 2.5 displays the time evolution process for the two time scale MDP model.

2.3.1 Slow Time Scale

The slow time scale MDP has a finite state space G, which is defined to be the set of all possible logical topologies (given the number of nodes, N, and the number of interfaces of each node, d). The action for the slow time scale is limited to either a BE or the null action, which means keeping the topology intact. The action space for the slow time scale is also finite and is denoted by $B(g)$, where $B(g)$ is the set with the null action (denoted by $\emptyset$) and all admissible BEs (as defined in Definition 2.5) given the current logical topology $g \in G$.

At the end of each slow time scale step $n \in \{0, 1, 2, \cdots\}$ an action (BE) $b_n \in$

$B(g_n)$ is taken and the state (topology) $g_n \in G$ transitions to $g_{n+1} \in G$ according to probability $P^s(g_{n+1}|g_n, b_n)$. This transition is always deterministic (transition probability is either 0 or 1), as we know how each BE changes the logical topology. The state and action taken at a slow time scale step n determine the fast time scale MDP for the duration of slow time scale step $n + 1$.

2.3.2 Fast Time Scale

In this thesis we adopt a fluid traffic model that assumes no constraint between the routing of traffic from one time step to the next. This means that routing of traffic at each time step can be done independently of how the traffic was routed in the previous step. Therefore the only dynamics in the fast time scale is in the traffic, which follows the underlying random process for the traffic and is not affected by the actions taken at the slow or fast time scales, i.e., the state of the fast time scale depends on the state of the traffic demand only. The traffic assignments are determined by the action at the fast time scale.

For an N node network, the state space for the fast time scale, Θ, is defined to be the set of all vectors of size $P = \binom{N}{2}$ with nonnegative elements. The element θ_p of vector $\theta \in \Theta$ corresponds to the traffic demand for S-D pair $(p) = (i - j)$.

Given a logical topology g, $L(g)$ is defined to be the set of all available LSPs for all S-D pairs. In this thesis we limit the number of LSPs that we use for each S-D pair. We allow a maximum of S LSPs that have the least number of hops. If the number of hops for a number of LSPs are equal, we let the LSP identification

number break the tie; the LSPs with lower IDs are selected over LSPs with higher IDs (this is just for uniqueness purposes). The resulting set of LSPs available for routing traffic is denoted by $L_S(g)$, which consists of $P \cdot S$ LSPs, S LSPs for each of the P S-D pairs.

Note that even though we allow for S LSPs, we may not be able to find S LSPs for every S-D pair. For example in a ring network, we can find a maximum of 2 LSPs for each S-D pair. Therefore, if we select $S = 3$ for a ring network the maximum number of LSPs we can find for each S-D pair is less than S. In general for networks with more than 2 interfaces per node, there is no such definitive number for the number of available LSPs per S-D pair. In a case where the number of LSPs available for each S-D pair is less than S, we define dummy LSPs to make the total number of LSPs for each S-D pair equal to S. This is only to facilitate the representation of the parameters we define later.

The action space at the fast time scale is defined to be A, the set of all $1 \times P \cdot S$ vectors. Element $a_{t,r}$ of vector $a_t \in A$ is defined as follows:

$$a_{t,r} = \rho_p^s(t+1)$$

where, $r = (p-1) * S + s$. Alternatively, $s = r \bmod S$ and $s = r \div S$.

In the above equation $\rho_p^s(t+1)$ represents the amount of reservation made for the sth LSP of the p-th S-D pair for the fast time scale step $t+1$. Note that the action at time t corresponds to the reservations for time $t+1$. The reservation for a dummy LSP is set to zero.

After the action is taken at the end of slow time scale step $n-1$, the fast time

scale MDP is set for the duration of $t = nT, \cdots, (n+1)T - 1$. For this duration the logical topology, g_n remains the same and therefore, the set of LSPs, $L_S(g_n)$, remains constant. Immediately after b_{n-1} is taken at the slow time scale, action a_{nT-1} is taken at the fast time scale with the knowledge of the action just taken at the slow time scale.

The state space only depends on the traffic vector and making a decision involves selecting the reservations for all LSPs given the current state of the slow time scale and the previous state of the fast time scale.

The transition probability for the state at the fast time scale step t (here assumed to be between slow time scale steps n and $n+1$) has the following form:

$$P^f(\theta_{t+1}|\theta_t, g_n, b_n, a_t) \quad = \quad P^f\Big(\theta_{t+1}|\theta_t\Big) \tag{2.1}$$

where g_n and b_n are the state and action of the slow time scale MDP at time n. Obviously, the traffic pattern at time $t+1$ does not depend on the state and action at the slow time scale and the action at the fast time scale.

The action a_t taken at time t most likely depends on θ_{t+1} or our estimate of it through fast time scale policy. This relationship is defined in the policy selected by the decision maker.

We will use the term "decision rule" when referring to infinite horizon and the term"policy" when referring to finite horizon. The problem of optimizing the fast time scale cost for the duration of one slow time scale is, by definition, a finite horizon problem with horizon of T. The slow time scale problem can be viewed as

either a finite horizon problem or an infinite horizon one depending on the type of traffic model.

When the slow time scale problem is infinite horizon, we define a sequence of T policies for the fast time scale. This sequence acts as a decision rule for the fast time scale in each slow time scale step. In other words, define a fast time scale *decision rule* $\omega^f = \{\pi_n^f\}$, $n = 0, 1, \cdots$ as a sequence of T-horizon *non-stationary policies* such that for all $n, \pi_n^f = \{\eta_{nT}^f, \cdots, \eta_{(n+1)T-1}^f\}$ is a sequence of functions where for all $k \geq 0, \eta_{t_k}^f : G \times B \times \Theta \rightarrow A$. We say that a fast time scale decision rule is *stationary* with respect to the slow time scale n if $\pi_n^f = \pi_{n'}^f$ for all n, n'. We will denote the set of all possible decision rules with respect to the slow time scale as Ω^f, and use Π^f as the set of all possible such T-horizon policies.

The total expected cost achieved by the fast time scale T-horizon non-stationary policy π_n^f will act as a single-step reward for the slow time scale MDP (see Equation 2.7). We define a slow time scale decision rule ω^s as a sequence of functions $\pi_n^s : G \times \Theta \rightarrow B$. We say that a slow time scale decision rule is *stationary* with respect to n if $\pi_n^s = \pi_{n'}^s$ for all n, n'. However in our problem we cannot restrict ourselves to the stationary decision rules for the slow time scale because the decisions made in the slow time scale take into account the future predictions of the traffic, which are based on the past average behavior of the network traffic. Such predictions are time varying and depend on n. We use Ω^s as the set of all possible decision rules. When the slow time scale problem is finite horizon, $\omega^f = \{\pi_n^f\}$ and functions $\omega^s = \{\pi_n^s\}$ are called policies instead of decision rules.

2.3.3 Cost functions

The cost function at the fast time scale step t is the sum of two terms. The first term is the sum of squared utilization for each of the lightpaths for fast time step t. The second term is a linear function of the loss (or the amount of traffic that is dropped) at time step t. To summarize, we can write the fast time scale cost function as:

$$R^f(g_n, b_n, \theta_t, a_t) = R^f(g_n, \theta_t, a_t) = \beta \sum_q u_q^2(\theta_t, a_t, g_n) + \sum_p D_p(\theta_t, a_t, g_n) \qquad (2.2)$$

where utilization u_q denotes the sum of reservations made for all LSPs passing through lightpath $[q]$, and D_p represents the part of traffic for S-D pair (p) that is dropped if the sum of reservations made on LSPs of that S-D pair is less than the traffic demand. The coefficient β signifies the tradeoff between having high utilizations on the lightpaths and dropped traffic. We select β so low that one unit of dropped traffic is penalized more than any penalty for higher utilization (this selection will be described in Chapter 4.

In the presence of deterministic traffic, we can make the reservations for the exact value of traffic demand. If the network cannot accommodate all the traffic demand, part of the demand will be dropped. When the traffic is not deterministic we do not know the exact value of the demand at time t_1, when we select the action (at time t_0) for either slow or fast time scales. Therefore, the actions we take at the slow and fast time scales, factor in the estimate of the traffic demand at the corresponding time step and we calculate an estimate of the dropped traffic based

on the actions taken. The estimate of the cost function is calculated based on the reservations and the estimate of the dropped traffic. This calculation is given in Section 2.4.2. When we reach time t_1 we can assume that the traffic demand for time t_1 is known and use that to calculate the reservations and dropped traffic, but realistically the exact value of traffic demand for time step t_1 becomes known at the end of this step but we need the reservations to distribute the traffic during t_1 time step. Hence, to calculate the real value of the cost function at the end of time step t_1 (when the real value of demand is known), we compare the reservation to the demand. If the reservation for every S-D pair is larger than the demand, there is no dropped traffic and the utilization of each lightpath (used in the cost function) is the sum of reservations made on that lightpath. Otherwise, the dropped traffic is the demand in excess of the reservation.

Parameters u_q and D_p can be expressed in terms of the reservations for each LSP as follows:

$$D_p = (\theta_p - \rho_p)^+ = (\theta_p - (\sum_i \rho_p^i))^+ \tag{2.3}$$

where ρ_p is the sum of the reservations ρ_p^i for LSPs φ_p^i corresponding to S-D pair (p), and θ_p is the actual traffic demand for that S-D pair. In the above equation $(x)^+ := \max(0, x)$. Also,

$$u_q = \sum_{\forall i,p \ni [q] \in \varphi_p^i} \rho_p^i \tag{2.4}$$

where the sum is taken over all LSPs going through the lightpath q.

Note that the sum of all reservations made on a lightpath cannot exceed the capacity C, of the lightpath, defined in Definition 2.2, and hence we have the following constraint:

$$\sum_{\forall i,p \ni [q] \in \varphi_p^i} \rho_p^i \leq C, \forall q \tag{2.5}$$

The structure of the fast time scale cost function is such that when the network is not congested we can select the reservations high enough (with respect to the traffic demand) to make sure there are no packet drops, in which case the second term of the cost in Equation (2.2) becomes zero. In such a scenario the utilization part of the cost function will balance the traffic distribution among the lightpaths, and optimizing this cost function becomes a quadratic programming problem. When the network becomes congested the second part of the cost function becomes dominant due to the fact that β was selected to be very low. In this case the cost function has both quadratic and piecewise linear terms.

The single step cost function at the slow time scale step n is the sum of two terms. The first term comes from the sum of the costs for all fast time scale steps in that slow time scale step. The second term is a penalty assessed for each slow time scale action that is not null. Since each branch exchange causes disruption in the traffic, this term is designed to avoid unnecessary branch exchanges. Specifically,

$$\begin{aligned} R^s(\theta_\circ, g_n, b_n, \pi^f) &= E^{\theta_\circ}\left\{ \sum_{t=nT}^{(n+1)T-1} R^f\Big(\theta_t, \pi^f(\theta_{t-1}, g_n)\Big) \right\} + \alpha I(b_n \neq \emptyset) \\ &= \sum_{t=nT}^{(n+1)T-1} E^{\theta_\circ}\left\{ R^f\Big(\theta_t, \pi^f(\theta_{t-1}, g_n)\Big) \right\} + \alpha I(b_n \neq \emptyset) \end{aligned} \tag{2.6}$$

where $\theta_\circ$ represents the state of the fast time scale at the last fast time scale step in slow time scale step $n-1$, $E^{\theta_\circ}$ represents conditional expectation given $\theta_\circ$, π^f is the policy for the fast time scale, $I(\cdot)$ represents the indicator function, resulting in 1 when the argument holds true and 0 otherwise, and $\emptyset$ represents the null action. The expectation can move inside the sum because the sum is finite. The single step cost functions for the fast and slow time scales are used below to define the overall cost function for the MMDP model.

2.3.4 Description of the problem

In this section a receding horizon model (see [19]) is adopted to set up the overall objective function. This model is used to approximate the infinite horizon case with a finite horizon case for each step and then after solving the problem for one step, the horizon is extended to H steps again. The solution to the receding horizon problem is optimal if our optimality equation only considered H steps with no terminal cost. At step n of the slow time scale, the goal is to minimize the following objective function, use the action for the first step and then resolve the problem for the next step with the horizon extended to H again:

$$J_{n,H}^{\omega^s,\omega^f}(g_\circ,\theta_\circ) := E^{g_\circ,\theta_\circ}\left\{\sum_{k=0}^{H-1}\gamma_k R^s\Big(\theta_{(n+k)T}, g_{n+k}, \omega^s(\theta_{(n+k)T}, g_{n+k}), \omega^f\Big)\right\} \tag{2.7}$$

where γ_n is the discount factor for slow time scale step n, $g_\circ$ is the initial topology that we start from, and $\theta_\circ$ is the last known traffic pattern before the start of slow time scale step $n=0$.

Given the initial states $g_\circ$ and $\theta_\circ$ for the two levels, the receding horizon value function for this problem is given below; the objective is to obtain a pair of non-stationary policies $\{\tilde{\pi}^f_{n,k}, k = 0, \cdots, H-1\} = \tilde{\omega}^f_n \in \tilde{\Omega}^f$ and $\{\tilde{\pi}^s_{n,k}, k = 0, \cdots, H-1\} = \tilde{\omega}^s_n \in \tilde{\Omega}^s$ that achieves the minimum in Equation (2.8). $\tilde{\Omega}^f$ and $\tilde{\Omega}^s$ are the sets of H-horizon policies for the fast and slow time scales.

$$V^*_n(g_\circ, \theta_\circ) := \min_{\omega^s \in \Omega^s} \min_{\omega^f \in \Omega^f} E^{g_\circ, \theta_\circ} \left\{ \sum_{k=0}^{H-1} \gamma_k R^s(\theta_{(n+k)T}, g_{n+k}, \tilde{\pi}^s_{n,k}(\theta_{(n+k)T}, g_{n+k}), \tilde{\pi}^f_{n,k}) \right\} \tag{2.8}$$

Note that the receding horizon decision rules for the fast and slow time scales are respectively $\omega^f = \{\pi^f_n\} = \{\tilde{\pi}^f_{n,0}\}$ and $\omega^s = \{\pi^s_n\} = \{\tilde{\pi}^s_{n,0}\}$.

2.3.5 Initialization Function

As part of the definition of MMDPs (see [18]), there is an initialization function defined that gives the conditional probability of being in state θ' at fast time scale step nT, given the state of the fast time scale θ at fast time scale step $nT - 1$.

$$\delta^f[g_n, b_n, \theta](\theta') = \mathcal{P}\{\theta_{nT} = \theta' \mid \theta_{nT-1} = \theta\} \tag{2.9}$$

$$= \delta^f[\theta](\theta') \tag{2.10}$$

Given the fluid model used here, this conditional probability does not depend on state of the slow time scale and the action at either the slow or fast time scale.

Similarly, we define the initialization function for the slow time scale as follows:

$$\delta^s[g, \omega^s](g') = \mathcal{P}\{g_{n+1} = g' \mid g_n = g, \omega^s\} \tag{2.11}$$

where, ω^s implicitly may depend on the traffic pattern. If we select ω^s to not depend on traffic pattern or if we make the traffic deterministic, then the conditional probability above becomes an indicator function, i.e., if the branch exchange selected by ω^s transitions g to g' the conditional probability equals 1, and 0 otherwise.

2.3.6 Traffic Model

In this thesis our traffic model consists of two parts: a deterministic part that captures the time-of-day changes in the traffic demand for each S-D pair in the network and a random part that represents the daily variations in the traffic. Element θ_p of the traffic vector represents the amount of traffic demand for S-D pair (p). θ_p is defined as follows:

$$\theta_t^p = Y_t^p + Z_t^p$$

where Y_t^p is the deterministic part of traffic, and Z_t^p is the random part of the traffic for S-D pair (p). The elements of the traffic matrix change at the fast time scale. These elements represent the amount of traffic flow for each S-D pair. At each fast time scale step t_1, we know Z_{t_1-1} and $Y_t, \forall t \geq t_1 - 1$. Based on this information action a_{t_1} is taken. At the slow time scale step n_1 the decision is made based on Z_{n_1T-1} and $Y_t, \forall t \geq n_1T - 1$. Once the decision at the slow time scale is made, we know the logical topology and the LSPs. At the fast time scale, the decision

is made to determine corresponding reservation for each LSP. At the realization of each fast time scale step, the real traffic flow becomes known, at which point we can determine whether the reservations we have for each S-D pair are enough to accommodate the real traffic flow or not. If the reservation is less than the real flow, the excess traffic will be dropped. Otherwise, there is no dropped traffic. If the reservation is more than the traffic flow, the cost of the reservation is incurred even though we may not use all the reservation. This is justified by the argument in Section 2.3.3.

We consider two types of traffic in this thesis, constant and dynamic traffic. In the constant traffic case, the random part of the traffic is zero and the deterministic part is constant. In the dynamic traffic model we know the deterministic part of traffic and try to estimate the traffic using the statistical properties of our traffic demand.

2.4 Finding the solution

In this section we use properties of the traffic model and the decision processes at the slow and fast time scales to simplify the optimality equation and find the solution to it.

We now expand Equation 2.8:

$$V_n^*(g,\theta) = \min_{\tilde{\omega}^s \in \tilde{\Omega}^s} \min_{\tilde{\omega}^f \in \tilde{\Omega}^f} E^{g,\theta}\Big\{ \sum_{k=0}^{H-1} \gamma_k \{\alpha I(\tilde{\pi}^s_{n,k} \neq \emptyset)$$

$$+ \quad \sum_{t=(n+k)T}^{(n+k+1)T-1} R^f(\theta_t, \eta_t^f, g_{n+k})\}\Big\} \tag{2.12}$$

$$= \min_{\tilde{\omega}^s} \min_{\tilde{\omega}^f} E^{\theta}\Big\{\sum_{k=0}^{H-1} \gamma_k \sum_{g^k} \prod_{l=0}^{k-1} \delta^s[g^l, \tilde{\omega}^s](g^{l+1})\Big\{\alpha I(g^l \neq g^{l+1})$$

$$+ \quad \sum_{t=(n+k)T}^{(n+k+1)T-1} R^f(\theta_t, \eta_t^f, g^k)\Big\}\Big\} \tag{2.13}$$

where δ^s is as defined in Section 2.3.5.

We consider two types of traffic, constant traffic and dynamic traffic. For constant traffic the model becomes deterministic and the optimality equation above simplifies to an ordinary search. For the dynamic traffic case solving the above equation is very hard and is not feasible for a real-time decision maker. To make this optimization a feasible problem to solve during a real-time decision making process, we consider one more approximation. We use the certainty equivalence approximation (see [10]), where the expectation with respect to θ is removed, and each term depending on θ is replaced by its expected value. After applying certainty equivalence (CE), Equation (2.13) is simplified to Equation 2.14, which is applicable to both constant traffic case and dynamic traffic case with CE.

$$V_n^*(g, \theta) = \min_{\tilde{\omega}^s}\Big\{\sum_{k=0}^{H-1} \gamma_k \sum_{g^k} \prod_{l=0}^{k-1} \delta^s[g^l, \bar{\omega}^s](g^{l+1})\Big\{\alpha I(g^l \neq g^{l+1})$$

$$+ \quad \min_{\tilde{\omega}^f} \sum_{t=(n+k)T}^{(n+k+1)T-1} R^f(\bar{\theta}_t, \bar{\eta}_t^f, g^k)\Big\}\Big\} \tag{2.14}$$

where $\bar{\theta}_t$ is the expected value of traffic vector at time t, given the traffic vector θ at time n, and $\bar{\omega}^s$ and $\bar{\eta}_t^f$ are the policies for the slow and fast time scales, given the deterministic traffic vector $\bar{\theta}_t, \forall t > nT$.

By the argument given at the end of Section 2.3.5, the expression $\sum_{g^k} \prod_{l=0}^{k-1} \delta^s[g^l, \bar{\omega}^s](g^{l+1})$ is going to be 1 only for one sequence of BEs and zero for all other sequences of BEs. Therefore, solving Equation 2.14 is equivalent to searching for a sequence of BEs that results in the best cost given the initial conditions and the average behavior of the traffic.

By the fluid model assumption, the action η^f_{t-1} at the fast time scale step $t-1$ does not affect the state θ_t of the fast time scale step t or the state g_n of the slow time scale. Therefore, the minimization over policy ω^f can be broken into minimizations over individual actions η^f_t:

$$\begin{aligned} V^*_n(g,\theta) &= \min_{\tilde{\omega}^s}\Bigg\{\sum_{k=0}^{H-1}\gamma_k \sum_{g^k}\prod_{l=0}^{k-1}\delta^s[g^l,\bar{\omega}^s](g^{l+1})\Bigg\{\alpha I(g^l \neq g^{l+1}) \\ &+ \sum_{t=(n+k)T}^{(n+k+1)T-1} \min_{\bar{\eta}^f_t} R^f(\bar{\theta}_t, \bar{\eta}^f_t, g^k)\Bigg\}\Bigg\} \end{aligned} \tag{2.15}$$

The interpretation of Equation 2.15 is that the fast time scale T-horizon policy π^f_n is reduced to T individual optimizations for the fast time scale steps, i.e., $\pi^f_n = \{\eta^f_{nT}, \cdots, \eta^f_{(n+1)T-1}\}$.

Also, observe that there are many topologies that are reachable from the initial topology g using a sequence of H branch exchanges. For each topology at the end of such a sequence, we can calculate the corresponding expected cost at the time step we arrive at that topology with the estimate of traffic for that time step. Now the problem at the slow time scale becomes a search problem among all sequences reachable from the initial topology for a sequence that has the lowest cost.

In the next two sections we show how to calculate the expected value of the

fast time scale cost function, given the topology g.

2.4.1 Constant Traffic

In this section we assume that the traffic is deterministic and fixed. The objective here is to find a sequence of branch exchanges that leads us to the optimal topology for the given traffic. Constant traffic combined with the stationary policy for the fast time scale and the fact that the topology remains constant during a slow time scale step indicates that the cost for each fast time scale step in one slow time scale step is the same. Therefore, for the constant traffic case we select $T = 1$. Since the traffic demand is deterministic, all expected values in Equation 2.15 can be replaced by their deterministic values.

Given the action at the fast time scale, there is a constant cost associated with every topology. Therefore the topology optimization problem is reduced to a search in the space of all topologies to find the one with the best cost. This can be done by a brute force search if the topology space for the given size of the network is small enough that such a search is feasible. In Chapter 3 we will see how the MRAS method can be used to find close to optimal solutions, and in Sections 2.4.5.1 and 2.4.5.2 more algorithms are proposed for this search.

In the case of constant traffic we are only interested in the end result (the topology and cost associated with it) as opposed to a discounted cost structure that is used mainly for the dynamic traffic case. Therefore, we select the discount factor to be zero for every step of the slow time scale except for the last step (final cost).

Next we will show how to find the optimum traffic distribution at the fast time scale given the traffic and topology.

Consider an N node network with d interfaces per node. There are $Q = \frac{Nd}{2}$ lightpaths and $P = \binom{N}{2}$ S-D pairs in the network. Suppose for each S-D pair in the network we allow S LSPs. Then there are a maximum of $P \cdot S$ LSPs in the network. With the constant traffic assumption the traffic vector θ is a constant $1 \times P$ vector. The reservation made on the sth LSP of the p-th S-D pair is a fraction (λ_{ps}) of the traffic demand (θ_p) for that S-D pair. The traffic distribution among LSPs of all S-D pairs is summarized in $\Lambda_{P \times PS}$, which is a block diagonal matrix with $1 \times S$ vectors along its diagonal:

$$\Lambda = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \dots & \lambda_{1S} & & & & & & & \\ & & & & \lambda_{21} & \lambda_{22} & \dots & \lambda_{2S} & & & \\ & & & & & & & & & O & \\ & & & O & & & & & \ddots & & \\ & & & & & & & & & & \\ & & & & & & & & \lambda_{P1} & \lambda_{P2} & \dots & \lambda_{PS} \end{bmatrix} \tag{2.16}$$

We summarize the relationship between the LSPs in the network and the lightpaths that each LSP goes through, in a $P \cdot S \times Q$ matrix Ψ, where element ψ_{rq} is defined as

$$\psi_{rq} = \begin{cases} 1 & \text{if } [q] \in \varphi_p^s, r = (p-1)S + s; \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Now we can write the utilization vector that holds the utilization for all light-paths, in terms of traffic vector as follows:

$$\begin{aligned} U_{1\times Q} &= \theta_{1\times P} \cdot \Lambda_{P\times P\cdot S} \cdot \Psi_{P\cdot S\times Q} \\ &= \tilde{\Lambda} \cdot \tilde{\theta} \cdot \Psi \end{aligned} \tag{2.18}$$

where,

$$\tilde{\Lambda} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \dots & \lambda_{1S} & \lambda_{21} & \lambda_{22} & \dots & \lambda_{2S} & \dots & \lambda_{P1} & \lambda_{P2} & \dots & \lambda_{PS} \end{bmatrix} \tag{2.19}$$

and

$$\tilde{\theta} = \begin{bmatrix} \theta_1 & & & & & & \\ & \ddots & & & & & \\ & & \theta_1 & & & O & \\ & & & \theta_2 & & & \\ & & & & \ddots & & \\ & & & & & \theta_2 & \\ & & & & & & \ddots \\ & & O & & & & \\ & & & & & & & \theta_P & \\ & & & & & & & & \ddots \\ & & & & & & & & & \theta_P \end{bmatrix} \tag{2.20}$$

The block diagonal property of matrix Λ makes it easy to see how the first line of Equation (2.18) could be rewritten in the second line. Let $\tilde{\lambda}_r$ denote the rth element of vector $\tilde{\Lambda}$. We can see that λ_{ps}, the sth LSP of the p-th S-D pair, is equal to $\tilde{\lambda}_r$, where $r = (p-1)S + s$.

Given a logical topology we can find the corresponding S LSPs for each S-D pair, which is equivalent to having matrix Ψ. With the constant traffic assumption in this section, the only parameters in Equation (2.18) that are not determined are elements $\tilde{\lambda}_r$ of vector $\tilde{\Lambda}$, which correspond to the reservations for each LSP in the network.

Now the first term in fast time scale cost function from Equation (2.2) becomes:

$$\sum_q u_q^2 = U \cdot U' \tag{2.21}$$

$$= \tilde{\Lambda} \cdot \tilde{\theta} \cdot \Psi \cdot \Psi' \tilde{\theta}' \cdot \tilde{\Lambda}' \tag{2.22}$$

$$= \tilde{\Lambda} \cdot \Upsilon \cdot \tilde{\Lambda}' \tag{2.23}$$

where

$$\Upsilon = \tilde{\theta} \cdot \Psi \cdot \Psi' \tilde{\theta}' \tag{2.24}$$

and superscript $'$ is the transpose operator.

The second part of the fast time scale cost function in Equation (2.2) is the sum of traffic losses for all S-D pairs. The lost traffic for S-D pair (p) is calculated as:

$$\theta_p (1 - \sum_s \lambda_{ps}) \tag{2.25}$$

Note that in the constant traffic case the reservations are made for the exact value of the traffic demand, since this value is known a priori. As a result

$$\sum_{s} \lambda_{ps} \leq 1, \forall p \tag{2.26}$$

Now we rewrite the second term on the right-hand side of Equation (2.2) as follows:

$$\begin{aligned} \sum_{p} D_p &= \sum_{p} \theta_p + \sum_{p} \sum_{s} \theta_p \lambda_{ps} && (2.27) \\ &= \theta \cdot e_P - \theta \cdot \Lambda \cdot e_{PS} && (2.28) \\ &= \theta \cdot e_P - \tilde{\Lambda} \cdot \tilde{\theta} \cdot e_{PS} && (2.29) \end{aligned}$$

where e_P and e_{PS} are respectively vectors of sizes $1 \times P$ and $1 \times PS$ with all 1 elements.

The optimization problem at the fast time scale for the constant traffic case and a given topology now looks like:

$$J_f^* = \min_{\tilde{\Lambda}} \{\tilde{\Lambda} \cdot \Upsilon \cdot \tilde{\Lambda}' - \tilde{\Lambda} \cdot \tilde{\theta} \cdot e_{PS} + \theta \cdot e_P\} \tag{2.30}$$

subject to

$$(\tilde{\Lambda}\tilde{\theta}\Psi)_q \leq C, \forall q$$

where $(\tilde{\Lambda}\tilde{\theta}\Psi)_q$ represents the qth element of vector $\tilde{\Lambda}\tilde{\theta}\Psi$, which is equivalent to u_q.

The above optimization is in the form of a quadratic programming problem and can be solved easily using the ILOG's CPLEX software (see [31]) with the condition that the objective function coefficient matrix (Υ) is positive semi-definite. By definition, $\Upsilon_{PS \times PS}$ is positive semi-definite if

$$\forall x \in \Re^{PS}, x'\Upsilon x \geq 0. \tag{2.31}$$

By Equation (2.24), Υ is positive semi-definite because

$$\begin{aligned} x\Upsilon x' &= x\tilde{\theta}\cdot\Psi\cdot\Psi'\tilde{\theta}'x' \\ &= (x\tilde{\theta}\Psi)\cdot(x\tilde{\theta}\Psi)' \geq 0 \end{aligned} \tag{2.32}$$

The solution to this optimization problem is $\tilde{\Lambda}^*$, which is the optimal distribution of traffic among the LSPs for each S-D pair, given the topology.

2.4.2 Dynamic Traffic

In this section we assume the random part of traffic for each S-D pair (p), $Z_p(t)$, is modeled by a Brownian motion (See [55]). We further assume that the Brownian motions for different S-D pairs are independent. This means that if we know the traffic at time t_0, i.e., we know the random part of traffic in addition to the deterministic part, then the estimate of the random part of traffic at a time $t > t_0$ is approximated by a Gaussian distribution, with a mean equal to the sum of observed random part at time t_0 and the deterministic part at time t, and a variance proportional to $t - t_0$:

$$m_t = \theta(t_0) - \theta_d(t_0) + \theta_d(t) \tag{2.33}$$

$$\sigma_t^2 = \tilde{\sigma}^2\cdot(t - t_0) \tag{2.34}$$

where m_t is the mean of traffic demand at time t, $\theta(t_0)$ is the observed traffic at time t_0, $\theta_d(t_0)$ is the deterministic part of traffic at time t_0, σ_t^2 is the variance of the random part of traffic at time t, and $\tilde{\sigma}^2$ is a variance factor that is a characteristic of the underlying Brownian motion.

Here we explain how to calculate the expected value of the fast time scale cost function at time t given the topology at time n_0 and traffic pattern at time t_0.

In this case, since we do not know the exact traffic demand, we may have to select reservations for the LSPs of a S-D pair higher than the estimate of the demand for that S-D pair to incur less cost due to loss in case the actual traffic demand was larger than our estimate. Therefore, Constraint (2.26) is not enforced and we use the reservations vector ($\rho(t) = [\rho_1 \cdots \rho_{PS}]$) in our cost function instead of the distribution vector multiplied by traffic demand ($\tilde{\Lambda}\tilde{\theta}$):

$$\begin{aligned} J_f(t) &= E\{\rho(t) \cdot \Psi \cdot \Psi' \rho(t)' + \sum_{p=1}^{P} \Big(\theta_t^p - \sum_{s=1}^{S} \rho_{ps}(t) \Big)^+ | g_t = g_0, \theta_{t_0} = \theta_0 \} \\ &= \rho(t) \cdot \Psi \cdot \Psi' \rho(t)' + E^{g_0,\theta_0} \{ \sum_{p=1}^{P} (\theta_t^p - \rho_p(t))^+ \} \\ &= \rho(t) \cdot \Psi \cdot \Psi' \rho(t)' + \sum_{p=1}^{P} E^{g_0,\theta_0} \{ (\theta_t^p - \rho_p(t))^+ \} \end{aligned} \tag{2.35}$$

where $\rho_p(t)$ represents the sum of all reservations on all S LSPs for the S-D pair (p), and Ψ is determined by the given topology g_0. The first term in Equation (2.35) takes into account the reservations made on every lightpath, which is the sum of the reservations made on the LSPs going through each lightpath. In other words, the reservations made on each LSP is multiplied by the number of lightpaths it goes through. The second term in this equation gives the expected value of the dropped traffic throughout the network.

Now we focus on the expectation in the second term of Equation (2.35), which is the expectation of the cost due to dropped traffic. We can expand it in an attempt to simplify the equation. To make it more readable, we omit the reference to time

t.

$$
\begin{aligned}
E^{g_0,\theta_0}\{\ (\theta^p-\rho_p)^+\} &= \int_{\rho_p}^{\infty}(\theta^p-\rho_p)\frac{e^{-(\frac{\theta^p-m_p}{\sigma_p})^2/2}}{\sigma_p\sqrt{2\pi}}\,d\theta^p && (2.36)\\
&= \int_{\frac{\rho_p-m}{\sigma_p}}^{\infty}(\sigma_p x+m_p-\rho_p)\frac{e^{-x^2/2}}{\sigma_p\sqrt{2\pi}}\sigma_p\,dx && (2.37)\\
&= \frac{\sigma_p}{\sqrt{2\pi}}\int_{\frac{\rho_p-m}{\sigma_p}}^{\infty}xe^{-x^2/2}\,dx+\frac{m_p-\rho_p}{\sqrt{2\pi}}\int_{\frac{\rho_p-m}{\sigma_p}}^{\infty}e^{-x^2/2}\,dx && (2.38)\\
&= \frac{\sigma_p}{\sqrt{2\pi}}\Big[-e^{-x^2/2}\Big]_{\frac{\rho_p-m}{\sigma_p}}^{\infty}+(m_p-\rho_p)Q(\frac{\rho_p-m_p}{\sigma_p}) && (2.39)\\
&= \frac{\sigma_p}{\sqrt{2\pi}}e^{\frac{-(\rho_p-m_p)^2}{2\sigma_p^2}}+(m_p-\rho_p)Q(\frac{\rho_p-m_p}{\sigma_p}) && (2.40)
\end{aligned}
$$

where, $x=\frac{\theta^p-m_p}{\sigma_p}$ and $Q(\cdot)$ represents the Q-function (complimentary cumulative distribution function) for the Gaussian random variables (See [39]).

Figure 2.6 shows a plot of the two terms in the right-hand side of Equation (2.40) and their sum. The first term is a Gaussian distribution curve and the second term is a Q-function multiplied by $(m_p-\rho_p)$. From the properties of the Gaussian distribution we know that the Gaussian bell curve is very close to zero when the variable is more than 2σ different from the mean and the Q-function is close to zero when the argument is more than 2 (corresponding to the reservations being much more than the average traffic) and close to 1 when the argument is less than -2 (i.e., average traffic being more than reservations). This translates into having a zero estimate for the cost of dropped traffic when the reservation is more than the estimate of the traffic demand by more than $2\sigma_p$ and having an estimate of $(m_p-\rho_p)$ when the reservation is less than the estimate of the traffic demand by more than $2\sigma_p$.

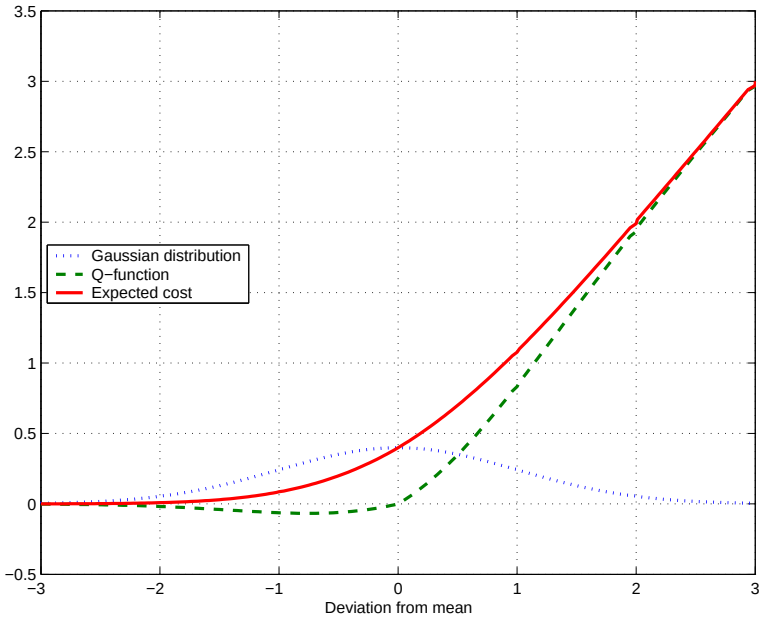

Figure 2.6: Sum of the first and second terms in Equation 2.36 plotted against $(m_p - \rho_p)$.

The right-hand side of Equation (2.40) could be approximated by a piecewise linear function, which is zero when the argument is less than -2 and is a line with slope of 1 when the argument is more than 2. Figure 2.7 displays the plot for the right-hand side of the Equation (2.40) and its approximation. This piecewise linear function can be used to approximate the expected cost of the dropped traffic for every S-D pair with reservation ρ_p and Gaussian traffic demand with mean m_p and standard deviation σ_p.

The overall cost function for the fast time scale could be formulated into a quadratic programming with piecewise linear terms, which is solvable using CPLEX software. Therefore, given the topology and the deterministic part of the traffic and the random part of traffic for time t_0, we can find the optimum set of reservations,

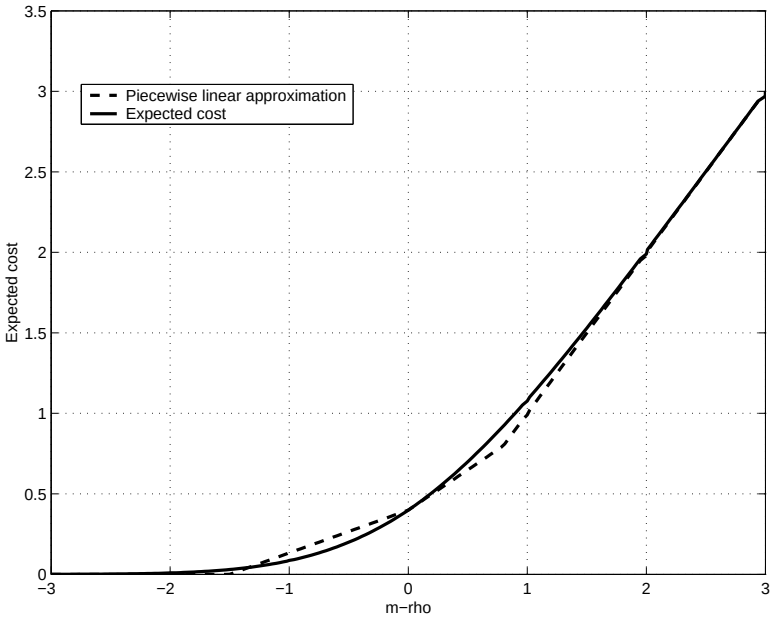

Figure 2.7: Approximation of the fast time scale cost by a piecewise linear function

i.e., the action at the fast time scale. This becomes the optimal policy for the fast time scale.

The slow time scale optimization is reduced to a search among all sequences of branch exchanges of length H, that starts from an initial topology g for the sequence that incurs the least cost. This search could be very time consuming for large networks and large H. Next we will discuss alternatives to this brute force search.

2.4.3 Alternative policies at fast and slow time scales

In the last section we showed how we can find the optimal set of reservations made at the fast time scale to optimize the cost for the fast time scale. The slow time scale problem becomes a search algorithm among all topologies reachable from the

initial state using branch exchanges to find the optimal path through the topology space that incurs the least cost.

The full search in the topology space and the optimization problem in the fast time scale, when combined together, are very time consuming. To be able to control the network in real time, we introduce two heuristic algorithms. One for the fast time scale as an alternative to the fast time scale optimal policy and one for the slow time scale.

2.4.4 Heuristic Algorithm

We describe a heuristic algorithm which is used to assign reservations to each LSP. The reservations made using this algorithm are for obvious reasons not as good as the optimal solution but the heuristic algorithm is useful when run time of the algorithm is critical. In this algorithm we first need a measure to sort the LSPs for each S-D pair and another measure to sort the S-D pairs in the order in which bandwidth is assigned to them. This determines the order in which the reservations are made for the LSPs of every S-D pair.

The rate of cost accumulation for each LSP is defined to be the increase in the total cost if we make reservation for one additional unit of traffic on that LSP. For example when we start fresh with no reservation made, this rate depends on the number of hops for each LSP. If we make a reservation for an LSP with 2 hops, the cost increases twice more than if we make the same reservation for an LSP with one hop, because in the former case the reservation is accounted for in two lightpaths,

whereas in the latter case it adds to the cost in only one lightpath. But as we make reservations using this heuristic, there will be LSPs that go through lightpaths with no more residual capacity. We cannot make further reservations on those LSPs. For such LSPs we count the deficit in the capacity towards the dropped traffic.

For example let us compare two LSPs for the same S-D pair. The first LSP goes through 2 lightpaths and the second LSP goes through 3 lightpaths. Therefore, when there is no congestion in the network, or more generally when none of the associated lightpaths are at full capacity, the rate of cost accumulation for each LSP is equal to the number of lightpaths they go through. Now, assume that the second LSP goes through a lightpath that is at full capacity. Then the rate of cost accumulation for that LSP becomes the rate at which we penalize the dropped traffic in our cost function, i.e. $\frac{1}{\beta}$, where β is the tradeoff coefficient in Equation (2.2). The LSPs for an S-D pair are sorted in increasing order of the rate of cost accumulation, i.e., we prefer to make reservations on the LSP that incurs less cost first.

We define a delta function ($\Delta(\cdot)$) over the set of S-D pairs, which returns the difference between the rate of cost accumulation for the first two LSPs for the S-D pair in its argument. The S-D pairs with higher $\Delta(\cdot)$ will get higher priority in receiving their reservations. The reason behind this is that for S-D pairs with low value of $\Delta(\cdot)$ it will make little difference to which LSP the traffic is assigned, therefore the assignment of traffic to LSPs of those S-D pairs is done later in the process. On the other hand, for S-D pairs with high $\Delta(\cdot)$ it is important to assign the traffic to the first LSP rather than second. Hence, such S-D pairs get higher priority to make sure their first LSPs will get the traffic assignment before the lightpaths

get close to full capacity.

The steps to this heuristic algorithm are listed in Figure 2.8. All time dependent variables are considered at a given fast time scale step t. This algorithm is run with a fixed traffic vector (m), in the case with dynamic traffic this is the mean of the traffic estimated through the past statistics of the network and σ_p is the standard deviation for pair p. In the case with constant traffic, m is the constant traffic and $\sigma_p = 0, \forall p$.

Note that in the case of constant traffic the reservations are made for the exact amount of traffic demand. But in the dynamic traffic case, since the exact demand is not known, we would like the reservations to be higher than the mean, m, by about 2σ. In this case we first make reservations for m and then if there is any more bandwidth left, additional reservations are made up to $m + 2\sigma$.

2.4.5 Slow time scale heuristic

Finding the optimal solution at the fast time scale is subject to a full search of the topologies in the state space of the slow time scale. This full search is feasible for small networks. In Chapter 3 we discuss how a full search can be done methodically to find the best topology. But as the size of the network grows larger (10 nodes or more) it becomes inefficient and impractical to do a full search of the state space.

We define two policies for the slow time scale. The first one is a (greedy) heuristic policy that looks into the immediate cost (one step ahead). The second policy uses the first policy and makes policy improvement using online simulations

Fast Time Scale Heuristic

Input: $g \in G$, $\varphi^s_{g_n,p}$, m_p, σ_p, $\forall p, s$

Assumptions: Number of S-D pairs: P, number of LSPs per S-D pair: S

Initialization: Create a table with P rows and S columns. Set the unassigned traffic $\delta_p = m_p, p = 0, \cdots, P$. Assign S-D pairs to the rows of the table and place the LSPs for each S-D pair in the columns of the corresponding row. Reset reservations $\rho^s_p \leftarrow 0, p = 0, \cdots, P, s = 0, \cdots, S$. Set $b \leftarrow 0$.

1- Sort available LSPs for each S-D pair. Set $a \leftarrow 0$

2- Sort S-D pairs based on their $\Delta(\cdot)$ measure. Set row pointer $c \leftarrow 1$.

3- For S-D pair p in row c, calculate the maximum reservation available on the LSP in the first column, ν, given the reservations already made on the lightpaths the LSP goes through. If $\nu = 0$ or $\delta_p = 0$ no reservation is possible, go to step 5.

4- Add to the reservation of LSP in the first column in row c: $\rho^s_p \leftarrow \rho^s_p + \min\{\nu, \delta_p\}$, where subscript p represents S-D pair (p) that is currently in row c, and superscript s represents LSP s that is currently in the first column. Set $\delta_p \leftarrow \delta_p - \min\{\nu, \delta_p\}$. Set $a \leftarrow 1$.

5- $c \leftarrow c + 1$. If $c \leq M$ go to step 3. Otherwise continue to step 6.

6- If $a = 0$ continue to step 7. Otherwise go back to step 1.

7- If $b = 1$ stop. Otherwise, set $b \leftarrow 1$, and set $\delta_p \leftarrow \delta_p + 2\sigma_p, \forall p$, and go to step 1.

Output: $\rho^s_p, p = 0, \cdots, P, s = 0, \cdots S$.

Figure 2.8: Fast time scale heuristic algorithm assigns reservations to every LSP of every S-D pair based on the traffic demand.

to determine which one of the possible BEs is more likely to yield a lower cost during the next few slow time scale steps. This is called a rollout policy.

2.4.5.1 Heuristic policy

First we provide a sketch of the algorithm. We consider every action in the set of admissible BEs (including the null action). For each of these BEs we calculate an estimate of the traffic distribution among LSPs resulting from the new topology and our estimate of the traffic matrix. Based on this estimate we then find the rate of cost accumulation during the next slow time scale step and use that figure to compare different BEs, and the BE that results in the lowest rate of cost accumulation is selected. Note that the heuristic algorithm for the slow time scale can use either of the two policies available for the fast time scale. The details of this algorithm are shown in Figure 2.9.

2.4.5.2 Rollout policy

The rollout policy applies a policy improvement method (called rollout) to the above heuristic policy in order to obtain a better policy [10]. At each decision time n (slow time scale), we make a decision analysis of all alternative actions, b_n^k. Each action corresponds to one of the admissible BEs. We are trying to minimize the cost, $J_n = E\{R_n^s(b_n) + J_{n+1}\}$, where $R_n^s(b_n)$ is the step cost for slow time scale step n given action b_n for that step. Since we do not know the exact value of the future costs J_{n+1}, we estimate it using online simulations.

Slow Time Scale Heuristic

Input: $g_n \in G$, estimate of the traffic demand $\tilde{\theta}_p(t), p = 0, \cdots, P, \forall t \geq nT$, list admissible branch exchanges $B(g_n)$.

Assumption: Number of LSPs per S-D pair: S

Loop until Stopping Rule is satisfied:

- For each b_m in the set $B(g_n)$ perform:
  1. Find the resulting topology, g^m, after application of b_m.
  2. Find S LSPs with least number of hops for each S-D pair.
  3. Run fast time scale policy to find the reservations for all LSPs at every fast time scale step within slow time scale step $(n+1)$. The reservations aim to cover the estimate of the traffic demand $\tilde{\theta}_p(t)$. This will result in $\rho_p^s(t), s = 0, \cdots, S, p = 0, \cdots, P, t = (n+1)T, \cdots, (n+2)T-1$.
  4. Given the traffic reservation for all S-D pairs, calculate the fast time scale costs, $R^f(t), t = (n+1)T, \cdots, (n+2)T-1$, using Equation 2.2, then calculate the slow time scale cost for $(n+1)$, using Equation 2.7.
- Find the BE that results in the lowest $E\{R^s(n+1)\}$.

Output: b^*.

Figure 2.9: Slow time scale heuristic algorithm finds the best next step BE based looking only one step ahead

Thus, assuming that a given action is taken we let the system proceed from the state where it is after that action and use the heuristic policy to make all the subsequent decisions for h steps, where h is defined to be the horizon. We should point out here that rollout algorithm as defined here is a single step rollout. But horizon helps determine how good a cost we can achieve (over a longer period of time than one step), if we pick each one of the branch exchanges. Our progression through one topology may not be immediately obvious by looking at the cost of that topology. It is the consequent topologies that are reachable through one topology that defines its value. We calculate the estimated discounted cost during these h steps. This cost is used as a measure of goodness for the original BE. Rollout policy picks the BE from the set of admissible BEs which results in the best multi-step cost.

2.4.5.3 Reducing the size of the action space in the slow time scale

In this section we describe a method that is used to reduce the size of the set of admissible branch exchanges. This is an attempt to give a direction to the search and eliminate the branch exchanges that may seem undesirable choices as the first step towards a local optimum.

By eliminating the unreasonable BEs we reduce the search time for our algorithm. If such BEs are indeed undesirable the result of our search would still find the same solution we would find with the full search of the available BEs. Figure 2.10 shows the algorithm. Here we define $B(g)$ to be the set of all admissible BEs, and

$B_1(g)$ is the resulting reduced set of admissible BEs.

There are two thresholds defined and used in this algorithm. These two thresholds, γ_l and γ_h, are positive real numbers less than 1 (with γ_l significantly less than γ_h).

The idea behind this algorithm is to include BEs that result in lightpaths that would help create direct (single hop) paths for S-D pairs that are contributing to the congestion in at least one lightpath. A lightpath is considered congested in this algorithm if the sum of reservations for that lightpath is more than $\gamma_h U_{max}$. With this definition, at least one lightpath (the one with maximum utilization) is considered congested with any given distribution of traffic.

A caveat here is that we might eliminate some BEs from the admissible set that would have led to topologies with worse costs but would have provided a path to another topology with a better cost. In the Chapter 4 we discuss this issue further.

Reducing the Size of Action Space

Input: $g_n \in G$, reservations $\rho_p((n+1)T-1), p = 0, \cdots, P$, list admissible branch exchanges $B(g_n)$, γ_l, γ_h.

1. Find the lightpath with maximum sum of reservations on it. Denote the sum of reservations for this lightpath by U_{max}.
2. Form set H such that it includes all S-D pairs that have a considerable bandwidth (more than $\gamma_l U_{max}$) on one of the congested lightpaths. We consider a lightpath congested if the sum of reservations on that lightpath is more than $\gamma_h U_{max}$.
3. From $B(g)$ add the null BE to $B_1(g)$.
4. Move to the next BE in $B(g)$.
5. If one of the two lightpaths resulting from this BE corresponds to a pair in H add this BE to $B_1(g)$ otherwise discard this BE.
6. Go to step 4.

Output: $B_1(g)$.

Figure 2.10: Algorithm to reduce the size of the admissible BE set

Chapter 3

Finding a Reference Solution

In the previous chapters we have described a model and policies that find suboptimal solutions. But, so far we have no reference as to how good a solution each policy provides. In the presence of fixed traffic it is easy to compare topologies and determine which topology is better than another one based on our cost function. Finding an optimal solution for the fixed traffic problem involves a search among all topologies. This optimal solution provides the reference solution that we are looking for. However the number of possible topologies grows exponentially as the number of nodes in the network grows.

The purpose of this chapter is to provide a reference solution using the Model Reference Adaptive Search (MRAS) algorithm [29]. The MRAS algorithm finds solutions to the static optimization problems that are often very close to optimal. The MRAS algorithm starts with an initial probability distribution that is used to generate a sample space. The cost associated with each member of the sample space is used to bias the probability distribution towards generating more samples with better costs. In each iteration of MRAS a new probability distribution is found that favors samples with better costs. MRAS culminates in a degenerate probability distribution that generates that same topology with probability 1. The corresponding topology is the solution of the MRAS algorithm.

In order to implement the MRAS algorithm for our problem, we need an algorithm to generate topologies based on a probability distribution. A topology as defined in Definition 2.3, is equivalent to a connected labeled graph as defined in graph theory. If we further assume that all nodes have the same number, d, of interfaces, the equivalent graph becomes a d-regular connected labeled graph. Generating random topologies is equivalent to generating random connected labeled graphs. In the first section of this chapter we give some definitions related to the graph theory and we give an algorithm that counts the number of labeled graphs with a given degree sequence (or the number of topologies in our problem). This counting algorithm gives us some perspective as to how big the topology space is and how fast the number grows with the number of nodes. In addition, it provides a basis for the random graph generation (topology generation) algorithm that we will use in the MRAS method.

The examples we give throughout this chapter are regular graphs (all nodes having the same number of interfaces) because this makes the exposition much easier to follow. But the results are valid for non-regular graphs as well.

3.1 Counting the topologies

The set G_N^d defined in Definition 2.3, can be described in graph-theoretic terminology as the set of d-regular connected labeled graphs of N vertices, which is the set of all labeled graphs with N vertices, each having a degree of d. For example, G_N^2 (set of 2-regular connected labeled graphs of N nodes) is the set of

all N-node rings. The lowest number of nodes for a ring is 3, since we do not allow more than one lightpath set up between two nodes. The number of possible N-node rings grows exponentially with N (to be exact, it is $(N-1)!/2$).

In this section we adopt a counting algorithm that is identical to the counting algorithm given in [52] with a minor difference. The algorithm given in [52] counts the number of labeled graphs (connected or not). Our minor modification to this algorithm allows us to count the number of connected labeled graphs. This algorithm is used to count the number of possible topologies (connected labeled graphs) given the number of nodes and each node's number of interfaces (degree). We start with a number of definitions that set up the stage for the counting algorithm.

Definition 3.1 *The* **degree sequence** *of an N-node network (or graph) is defined to be a sequence of N numbers, where each number in the sequence represents the degree of one of the nodes of the network. It is required for the digits to be sorted in ascending order from left to right.*

Note that sorting the digits assures each degree sequence has a unique representation.

Definition 3.2 *A* **partition** *for a network is defined to be the ordered set of non-negative integers $m_1, \cdots, m_{N-1}$, where m_i is the number of nodes having degree i. The following notation is used for such partition:*

$$(1^{m_1}2^{m_2}3^{m_3}\cdots)$$

Definition 3.3 *A degree sequence or the corresponding partition is said to be* **graphical** *if there exists at least one graph that matches the degree sequence.*

Definition 3.4 *An* **empty node** *is one with no connections to other nodes.*

Definition 3.5 *An* **empty graph** *is one with all empty nodes.*

Definition 3.6 *A node with d interfaces is said to be* **partially connected** *if it is connected to $d_1 < d$ other nodes.*

Definition 3.7 *The* **remaining degree** *of a partially connected node is $d - d_1$.*

Definition 3.8 *A* **partially connected network** *or* **partially connected graph** *is one with at least one node that is partially connected.*

Definition 3.9 *The* **residual degree sequence** *of a partially connected network is the sequence of the partial degrees of its nodes sorted in ascending order. The partition corresponding to the residual degree sequence is called the* **residual partition**.

For a residual degree sequence we may drop the leading zeros. This will not cause any confusion as long as the number of nodes in the network is known. For example the degree sequence 000002 can be written as 2. For any size network, the following statements are true. Residual degree sequence 2 is not graphical because one node cannot connect to itself. Residual degree sequence 22 is not a graphical sequence because two nodes cannot connect to each other twice. But the residual degree sequence 112 is graphical because we can connect the node with degree 2 to each of the nodes with degree 1, and the result is a connected graph.

Note that an all-zero residual sequence corresponds to a fully connected network. We should also note that by fully connected network we do not mean a mesh network, where all nodes have direct connections to all the other nodes in the network.

Definition 3.10 *A* **subnetwork** *(or* **subgraph***) g_1 of an N_2-node network (or graph) g_2 has N_1 nodes, $N_1 \leq N_2$, where the nodes of g_1 form a subset of the nodes of g_2 and the edges of g_1 are a subset of the edges of g_2.*

Definition 3.11 *The residual partition p_2 of network g_2 is called a* **subpartition** *of the residual partition p_1 of network g_1 if there exists a set of vertices in g_2 that if removed, the residual partition p_2 becomes equal to p_1. Equivalently, we say p_2 is reachable from p_1.*

Given the above definitions, we can easily see that an all zero residual partition is reachable only from a graphical partition. Equivalently, the all-zero partition is a subpartition of only the graphical partitions.

The objective of the counting algorithm is to find the number of ways a set of nodes can connect to each other, which results in a connected graph matching the given degree sequence. The starting point of the algorithm is a degree sequence for N nodes in the network and N disconnected nodes, each having degree matching one of the numbers in the given degree sequence. In each step of this algorithm we select a node, called a **pivot**, and connect all of its remaining interfaces to other nodes with non-zero residual degree and eliminate that node from the residual degree sequence, as its residual degree becomes zero. As a result, we are left with

a subpartition of the residual partition of the previous step. We repeat this process for each of the nodes having a non-zero remaining degree until either all nodes have been eliminated (resulting in a graph) or we reach a degree sequence that is not graphical.

For example, 333333 is a degree sequence representing all networks that have 6 nodes and 3 interfaces available at each node. The corresponding partition is 3^6. We can eliminate the first node by connecting it to three other nodes, resulting in a residual degree sequence of 022233. We may or may not display the leading zeros. The corresponding partition is $2^3 3^2$.

The purpose of defining partitions and sub-partitions is that, given a partially connected network, we are interested in counting the ways the rest of the free interfaces in the network could be connected to each other to form a fully connected network.

Definition 3.12 *The* **value** *of a given degree sequence or the corresponding partition is defined to be the number of distinct graphs with that degree sequence or partition.*

By convention we give a value of 1 to an all-zero partition, and a value of zero to all non-graphical partitions. Function $V(\cdot)$ is defined on the set of all partitions and gives the value of its argument.

Definition 3.13 *A* **node elimination step** *is the process of selecting the nodes to which each and every free interface of the pivot node should connect.*

At the end of a node elimination step the pivot digit becomes zero and joins the fully connected part of the network. At this point the pivot node can be eliminated from the residual degree sequence. If there are any more nodes that as a result, have a zero residual degree, those nodes also join the set of fully connected nodes.

The number of ways we can select the points that connect to the free interfaces of the pivot node is very similar to the formula given in [52]. Let the residual partition of the graph be $\alpha = (1^{m_1}2^{m_2}\cdots\Delta^{m_\Delta})$, where Δ denotes the maximum degree of the remaining sub-partition. Let our pivot point, P, be of degree δ. Now let $k_i, i = 1, 2, \cdots, \Delta$ be the number of points of degree i to which P is connected. By eliminating the pivot node we reach a graph with the following partition:

$$\beta = (1^{m_1-k_1}2^{m_2-k_2}\cdots\delta^{m_\delta-k_\delta}\cdots\Delta^{m_\Delta-k_\Delta}).$$

There are

$$M(\alpha; k_1, k_2, \cdots) = \Big[\prod_{i=1, i\neq\delta}^{\Delta}\binom{m_i}{k_i}\Big]\cdot\binom{m_\delta - 1}{k_\delta} \tag{3.1}$$

ways of choosing which specific nodes will connect to the free interfaces of P. Note that $m_\delta - 1$ appears instead of m_δ because P itself is not allowed to connect to itself.

To avoid counting of fully connected networks that are not considered valid topologies, we define an invalid partition.

Definition 3.14 *Define an* **invalid sub-partition** *to be one that is composed of a non-zero number of fully connected nodes and a non-zero number of empty nodes and no partially connected nodes.*

As an example of an invalid sub-partition, consider the sub-partition 00003333 for an 8-node 3-regular graph. This sub-partition fits the definition of an invalid sub-partition. Such a sub-partition is illegal because there are two sets of nodes in invalid sub-partitions: nodes that are already connected to each other and do not have any other free interfaces, and empty nodes. This means that at best we could make two disjoint connected graphs from this kind of sub-partition, which is not allowed (see Definition 2.3).

Definition 3.15 *A* **partition tree** *is defined to be a rooted tree that has a network partition or a degree sequence as its root, and the children of each node are sub-partitions that are reachable from that node by node elimination steps. The leaves of a the partition tree are either all-zero subpartition or a subpartition with value of zero.*

In order to build a partition tree, we start from a network partition and recursively find the children for each subpartition by performing a node elimination step on it. When all the leaves of the tree have a value of either 0 or 1, the partition tree is complete. In the next section we use this partition tree to generate random graphs. It is crucial for the random graph generation algorithm to save the partition tree in memory. Since the tree grows exponentially with the number of nodes in the graph, it would be hard to keep the tree in its original form. However, the nodes in a partition tree tend to repeat several times. This property is crucial in reducing the amount of memory needed to save a partition tree. We build a **compact partition tree** the same way as the partition tree, except when we calculate the children of a

partition p (say $p1, p2, \cdots$), we check whether each sub-partition p_i already exists in the tree or not. If a partition p_i does not already exist in the tree, we create a new node with that sub-partition. Otherwise, we merely add a link from partition p to the existing partition p_i in the tree. With this method we significantly reduce the amount of memory needed to save a partition tree in memory. We give statistics about the amount of memory needed to save a partition tree in Chapter 4. With application of this compacting procedure the resulting partition tree is no longer an actual tree. However, we still call it a partition tree because it has the same information. Figure 3.4 show an example of such compact partition tree.

Theorem 3.1 *Given any starting partition, if the pivot node is selected to be a partially connected node (except for the initial step), and if the all-zero subpartition is reached by traversing the partition tree without going through an invalid subpartition, the resulting topology is valid.*

Proof: The mathematical induction is used to prove this theorem. The initial partition is the sub-partition right after the first node elimination step. This sub-partition consists of a zero, representing the previous pivot node that has no more free interfaces, and a set of nodes that have at least one free interface remaining. Otherwise, an invalid sub-partition is reached. The pivot node is selected from the set of partially connected nodes. This is depicted in Figure 3.1. This initial sub-partition has one fully connected node. The following notable properties hold for the initial sub-partition:

1. The nodes with free interfaces have direct connections only to the fully connected nodes.

2. The pivot node has connections only to the fully connected nodes (no connections to the nodes with free interfaces).

3. Every node in the set of connected nodes (initially consisting only one node) has a direct or indirect connection to every other node in this set, i.e. there is no disjoint subnetwork in it.

4. There are no two nodes in the set of connected nodes that are connected with more than one lightpath.

The set of 4 properties above are posed as the induction hypothesis for the nth step of the algorithm. We prove that if the hypothesis holds for step n, it is also going to hold for step $n+1$.

Note that for the case of a 3-regular graph, any pivot in the intermediate steps cannot be a 3 because by definition a pivot node has all zeros to its left and is less than or equal to any digits to its right. In other words, if a pivot node is 3 all digits to its left are zeros and all digits to its left are 3's, which would make the sub-partition an invalid one.

Now we start from a sub-partition with the 4 properties above and show that a node elimination step does not change these properties.

Property 1 holds because we always set up connections between the pivot node and the set of nodes with free connections, and the pivot node joins the set of fully connected nodes at the end of each node elimination step.

Property 2 holds because the pivot node is always selected from the set of nodes with free interfaces.

Property 3 holds because if the sub-partition is a valid one the pivot node is a partially connected node, i.e., it has connections to the fully connected network. Any other node that joins the fully connected network in the same step as the pivot node has connections to the pivot node for that elimination step.

Property 4 holds because we never assign two connections between the pivot node and any other node in the set of nodes with free connections.

Hence, if the series of node elimination steps ends in an all-zero sub-partition properties 3 and 4 hold for that network and therefore, by definition of valid topolo-

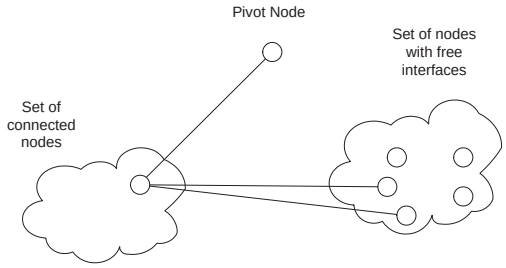

Figure 3.1: Initial condition for the induction

gies the resulting topology is valid □

Figure 3.2 describes the algorithm that finds the value of a partition. To add clarity to this algorithm, define a partition tree in which the root is the starting partition, the nodes of the tree are sub-partitions reachable from the root by node elimination steps, and the leaves of the tree are all-zero, non-graphical or invalid sub-partitions.

Note that when we start the algorithm the tree is not complete, and in the intermediate steps the leaves of the tree are just sub-partitions reachable from the root. But as we complete the algorithm the resulting tree has leaves that are either all-zero, non-graphical or invalid. The value calculated in this algorithm by definition represents the number of possible topologies for that network partition given the constraints that were used to build the tree.

Figure 3.3 shows an example with $N = 4$ and $d = 2$ (a 4 node ring). We start from network partition 2222, showing there are 4 nodes each having 2 free interfaces. Now we count the number of ways we can connect the first node, with two free interfaces, to the remaining nodes (a node elimination step), which is $\binom{3}{2} = 3$. This number is displayed over the arrow pointing to the resulting network sub-

Find the value of a network partition

Input: Network partition p_r.

1- create the partition tree with p_r as its root.

2- On each arrow in the tree connecting a parent, p, to its ith child, c_i, note the number of ways, μ_i, the corresponding transition could take place using Equation 3.1.

3- To each all-zero sub-partition assign a value of 1 and to each invalid or non-graphical sub-partition assign a value of zero. A value of 1 represents one valid topology and 0 represents no valid topology.

4- Calculate the value $V(p)$ of each node p, with l children, from the value $V(c_i), i = 1, \cdots, l$ of its children, as follows: $V(p) = \sum_i \mu_i.V(c_i)$, $i = 1, \ \cdots, \ l$.

5- Stop when the value of the root of the tree is calculated.

Output: $V(p_r)$

Figure 3.2: Algorithm to calculate the value of a network partition

partition. As indicated in Definition 3.11, we always sort the digits so that all the zeros are on the left and all d's are on the right. One consequence of this sorting is that we connect the nodes with fewer free interfaces first. It turns out that for ring networks the resulting tree has one main branch and all side branches result in invalid states. As a result, the formula for ring networks simplifies to $(N-1)!/2$.

The example depicted in Figure 3.4 shows the details of this algorithm for a network with $N = 6$ and $d = 3$. We see that for this network there are some valid branches in the tree. We can also see that, as we go through different branches, some sub-partitions repeat. In such cases we can calculate the value for one of them and use the same figure for the others. For simplicity, we have combined such nodes into one. For example, state 000123 has no possible combinations that result in a valid topology. Thus, we can assign value of 0 to this state without having to do the calculations every time. Non-graphical sub-partitions that could arise in our calculations for networks of up to 10 nodes are all sub-partitions ending in 02, 013, 022, 033, 0123, 0233, 01133, 01333. Note that the list above is for networks with the constraint of 3 interfaces per node. As the size of the network gets bigger some invalid sub-partitions also arise in the corresponding trees. Examples of such invalid sub-partitions are 0000333333, and 0000003333 for a 10-node network. Keeping track of such invalid sub-partitions is very easy for d-regular graphs and is automatic when we sort the degree sequence in ascending order, because by taking the lowest degree as pivot we guarantee that if the sub-partition is not invalid, the pivot is not an empty node. In [52] the authors take the pivot node to be one with the highest degree; they do not care about the connectedness of the graph.

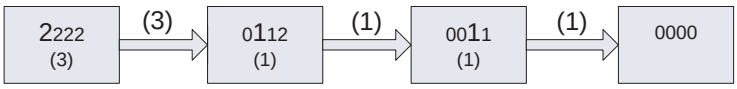

Figure 3.3: Calculation of number of possible topologies in a 4 node network with 2 interfaces available at each node.

For non-regular graphs, we keep track of the non-empty, non-zero nodes by placing a $*$ next to each one and we have to select the pivot from these nodes. With this minor modification, we are able to count the connected graphs with any degree sequence.

As an example of how we calculated the number of ways we could go from one sub-partition to its children in the tree consider sub-partition 022233 in Figure 3.4. We expect to connect the node represented by the leftmost nonzero digit in the sub-partition that has two interfaces to other nodes to its right. The first option is to connect both of the interfaces to nodes that have two free interfaces remaining. In this case there is only one possible way, because there are only two such nodes available and we have two free interfaces. The second option is to connect both interfaces to nodes that have three free interfaces. In this case also there is only one possible way, because there are two interfaces to be connected and there are two nodes remaining with three free interfaces. The third option is to connect one of the interfaces to a node with two free interfaces and the other interface to a node with three free interfaces. In this case there are $\binom{2}{1}.\binom{2}{1} = 4$ possibilities.

Using this algorithm we can calculate the number of possible topologies for 6-, 8-, and 10-node networks. In Table 3.1 we compare this number to that of ring

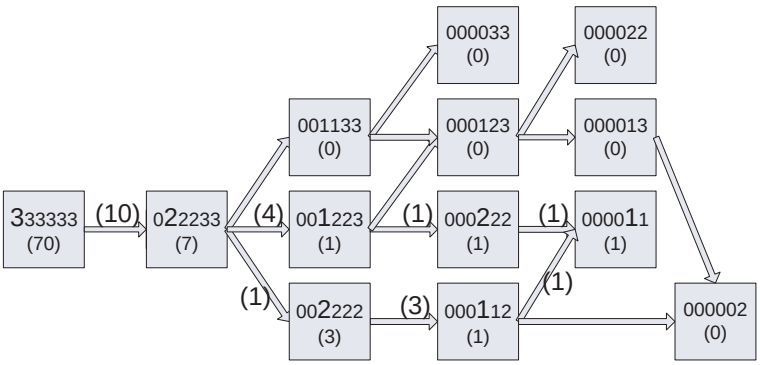

Figure 3.4: A compact partition tree and calculation of number of possible topologies in a 6 node network with 3 interfaces available at each node.

Number of nodes	2 interfaces per node	3 interfaces per node
4	3	1
6	60	70
8	2520	19320
10	181440	11166120

Table 3.1: Number of possible topologies, 2- vs. 3-interface constraint for each node

networks. The values of the partitions found in calculation of the smaller networks could be used recursively in calculating the values of the partitions for the larger networks. By saving these values in a reference table the computation time for the larger networks will be reduced.

3.2 Random graphs with uniform distribution

In this section we define an algorithm that can generate random topologies with uniform distribution with a given degree sequence. First we build the compact

partition tree for the given degree sequence. Then for generating a random graph, we start from the root of the partition tree and randomly pick a path from the root of the tree to an all-zero child. Each such path corresponds to a graph with the given degree sequence.

Definition 3.16 *The* **connect subset** *of a node is the subset of nodes that have a lightpath to this node. It is represented by the corresponding d node numbers sorted in ascending order from left to right.*

For example if node 0 is connected to nodes 3, 5 and 9, the corresponding connect subset for it would be $\{3, 5, 9\}$ or in short 359. In each iteration of the algorithm described in Figure 3.5, the connect subset of one of the partially connected nodes is selected and as a result, at least one node is added to the set of fully connected nodes.

Theorem 3.2 *The algorithm in Figure 3.5 has the following properties:*

A. *It chooses each possible graph with the given degree sequence uniformly at random.*

B. *It generate a valid graph with every run with probability 1.*

C. *It has a runtime on the order of* $\mathcal{O}(Nd\log(d))$ *for d-regular graphs.*

Proof:

A. This part of this theorem is straightforward, since at every decision point the algorithm picks a branch with a probability proportional to the number of possible graphs that could stem from that branch.

Uniform generation of random topologies

Input: $P_0 = P$, network partition

Initialization: Create a compact partition tree with the network partition as its root. Set $j \leftarrow 0$. Set the current partition p to be the root of the partition tree.

1- Randomly pick a node m from the pivot candidates, i.e., nodes with the lowest number of free interfaces.

2- Pick branch i stemming from p (leading to child c_i of p), with probability proportional to $\mu_i V(c_i)$.

3- Out of the μ_i ways child c_i can be reached, pick one at random and find the corresponding connect subset for the pivot node.

4- If c_i is the all-zero partition stop, otherwise set $p \leftarrow c_i$ and go to Step 2.

Output: Random graph picked uniformly at random from the set of all possible graphs with the given degree sequence.

Figure 3.5: Random topology generation algorithm (uniform distribution)

B. Since the branches of the partition tree that end in invalid or non-graphical partitions have a value of zero, those branches are picked with a probability of zero.

C. Each iteration of the algorithm includes Steps 2, and 3. Step 2 is a weighted selection of one of the branches. For child j of the current node (which has k children), the cumulative weight $\frac{\sum_{i=0}^{j} \mu_i V(c_i)}{\sum_{i=0}^{k} \mu_i V(c_i)}$ is calculated when the partition tree is built. A random number $0 \leq r \leq 1$ is generated. A binary search is performed to find the smallest j for which the cumulative weight is larger than r. This binary search has a runtime of $\mathcal{O}(\log(k))$. To find the worst case k, consider that the pivot node is at most d, and there are at most d digits in the partition for the current node. Assuming that the power of each digit in the partition corresponding to the current node is larger than d, the worst case scenario results in d^d branches. Therefore, the runtime of this step is at most $d\log(d)$. In Step 3, at most d interfaces of the pivot will be connected to d nodes matching the selection made in Step 2. There are at most $N-1$ iterations. Therefore, the worst case complexity of this algorithm is $\mathcal{O}(Nd\log(d))\square$

Note that worst case scenario for Step 2 only happens when N is in the order of d^2. Therefore, for small N a runtime of $\mathcal{O}(Nd)$ is expected. In Chapter 4, numerical results for the runtime of this algorithm are provided.

It should be noted that the initialization for the algorithm has a runtime that grows as a polynomial in the number of nodes in the graph. But once the partition

tree is complete, the speed of the algorithm is faster than any other algorithm in the literature (see Section 1.5). In Chapter 4 we give the runtime of the simulations we ran for this algorithm.

The very fast runtime of this algorithm is due to property B. The main reason for a higher runtime for pairing algorithms is that they cannot guarantee that at the end of each attempt they have a valid graph. The best algorithm available in the literature (see [8]) is based on the pairing model and has a runtime of $\mathcal{O}(md\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ to obtain a distribution within $1+\epsilon$ of the uniform distribution and failure probability of less than δ; i.e., such algorithms have a large runtime to generate close to uniform distributions and low failure probability.

Our algorithm has the following advantages compared to the algorithm given in [8]:

- After the initialization, our algorithm is extremely fast and generates random topologies on the order of $\mathcal{O}(Nd\log(d))$, where N is the number of nodes in the network and d is the average degree of each node.

- There is no trial and error, and the graph reached at the end of the algorithm is valid and connected with certainty.

- The distribution generated by our algorithm is exactly uniform.

- Our algorithm can be used to generate general random graphs with any degree sequence. There are no restrictions on the type or degree sequence of the graph.

The only drawback of our algorithm is that there is a precalculation necessary for it to set up the database for each possible degree sequence that could appear in the corresponding partition tree. This database gets quite large for networks with a large number of nodes and especially larger if the maximum degree of the nodes in the network is large. In Chapter 4 we give some numerical results as to how large a database we need as the number of nodes in the network grows. It turns out that using this algorithm it is practical to generate random graphs with up to 200 nodes for 3-regular graphs, 50 nodes for 6-regular graphs, and 30 nodes for 12-regular graphs, and so on.

3.3 A variation of the random topology generation algorithm

In this section we introduce a variation of the random topology generation algorithm that uses a matrix P to create bias towards topologies with better cost. We assume we have an N-node network and each node has d interfaces. Define a probability matrix, P, with N rows and $\binom{N}{d}$ columns. Each column represents one possible connect subset for each node. Going back to the 10-node network example if $d = 3$, and $N = 10$, there are 120 columns that start with connect subset 012, and end with 789.

Each row P^m of matrix P gives a probability distribution for selecting a connect subset for node m. Matrix P is updated after each iteration of the algorithm to show the valid connect subsets for each node with positive probability. Therefore, P must meet the following consistency checks:

- Every row of P must sum up to 1.
- The elements of the matrix P where the row number is one of the members of the connect subset are set to zero.
- For a partially connected topology, each nonzero element in row m of the matrix P must correspond to a connect subset that includes nodes that node m is already connected to.

It should be noted that there is a degenerate probability matrix corresponding to every given topology. To find this degenerate matrix, we start with a zero matrix with the size of P. Then we go through each row, m, of this matrix and place a 1 in the column which corresponds to the node combination that is connected to the node m in the topology. As we will explain in the next section, the purpose of the MRAS iteration is to reach one such degenerate matrix that corresponds to the solution topology.

Now we define an algorithm that starts from a probability distribution matrix P, and generates a random topology. The algorithm in Figure 3.6 is used to generate such a random topology. During each iteration the probability matrix P is modified. The subsequent matrix is represented by P_j, where j is the iteration step of the algorithm.

Now we examine the topology generation algorithm more closely. In step 5 we eliminate all entries in the probability matrix that result in an invalid topology. First, in step (5.A) we make sure that for the nodes that have been already connected, the corresponding row has all zeros except a 1 that represents the connect

Random topology generation with bias

Input: $P_0 = P$, network partition

Initialization: Create a tree with the network partition as its root. Set $j \leftarrow 0$.

1- Randomly pick a node m from the pivot candidates, i.e., nodes with the lowest number of free interfaces.

2- Use the probability mass function in row m to pick the connect subset (abc) to node m.

3- Connect node m to all nodes in the connect subset picked in step 2 and determine the resulting sub-partition.

4- If the new sub-partition is all-zero, Stop. Otherwise continue to step 5.

5- Update P_j to find P_{j+1} using the following steps:

[A-] In row m set entry corresponding to connect subset abc to 1 and set every other entry to zero.

[B-] In rows a, b, and c, set every entry corresponding to connect subsets that do not include m to zero.

[C-] In rows other than a, b, c, and m, set every entry corresponding to a connect subset that includes m to zero.

[D-] In every row, including a, b, and c set every entry corresponding to a connect subset that results in an invalid sub-partition to zero.

[E-] Normalize the rows of the resulting matrix.

6- Go back to step 1.

Output: A random topology consistent with the network partition and biased by matrix P.

Figure 3.6: Random topology generation with bias

subset connected to that node. Then in step (5.B) we make sure that the connect subsets that have a chance to be selected in the future include node m that is already connected to a, b, and c. In step (5.C) we make sure that no other node is going to connect to m, since m does not have any remaining interfaces. Finally, in step (5.D) we make sure that no invalid sub-partition has a chance to be selected in the future steps. The resulting P matrix ensures that we generate a valid topology at every attempt.

Property 3.1 *The distribution of the connect subsets for each node in sample topologies generated using this algorithm does not generally match the rows of matrix P.*

The reason for this phenomenon is that the conditional probability of picking a connect subset for a node depends on the order in which we pick our pivot nodes. If we fixed our first pivot node to be node m, the distribution of the connect subsets for node m would match row m of matric P. However this same statement does not hold for nodes other than m, because of how we update rows of matrix P in step 5 of the algorithm.

One special case (exception to above property) is when we have a symmetric degree sequence and matrix P has uniform distribution in every row. In this special case due to the symmetric nature of the problem biases given to each topology due to the picking order averages out over different picking orders.

Despite Property 3.1, the algorithm above is very useful in implementing the MRAS method for our problem. We randomize the picking order to reduce the

amount of bias given to topologies due to the picking order. Therefore, the resulting sample of topologies generated using this algorithm is in some neighborhood of the intended distribution. As we progress through stages of the MRAS method and the distributions in matrix P become more concentrated, the effects of Property 3.1 are reduced and eventually disappear.

3.4 Model Reference Adaptive Search Algorithm

The MRAS algorithm(see [29]) is a generic model based approach to combinatorial and continuous optimization problems, where a solution space is created using a probability model that is updated in every step of the algorithm to improve the probability of getting a sample solution with better cost. In this method a sequence of parameterized probability distributions is defined, and in each iteration of the algorithm the parameters of the probability distributions are updated such that the sample solutions generated by using the probability distributions are more concentrated on solutions with better cost function. The MRAS method is similar to another algorithm called the Cross Entropy (CE) method ([22], [60]) that uses a similar sequence of parameterized probability distributions. However the MRAS method guarantees convergence to a solution with very mild assumptions.

Both the MRAS and CE methods involve an iterative procedure where each iteration can be broken down into two phases:

- Generate random sample solutions based on a parameterized probability distribution.

- Update the parameters of the probability distribution based on the costs associated with the samples generated in the first phase, in order to bias the future generation of the samples towards solutions with better cost.

To be able to use this algorithm, we first need to find a way to generate valid topologies based on a probability distribution function. This is done using the random topology generation algorithm explained in the previous section. Then in each iteration we update the probability distribution matrix that generates the topologies in such a way that the topologies that return lower costs are more likely to be generated in the subsequent iterations. The iteration will end in a degenerate probability matrix that would generate the same topology every time. That topology is the solution found by MRAS.

In [20] the theory behind MRAS algorithm is given for a maximization problem. Here we apply this approach for a minimization problem. Consider the optimization problem below:

$$x^* = \arg\min_{x \in \chi} J(x) \tag{3.2}$$

where χ is a non-empty set and $J : \chi \rightarrow \Re$ is a deterministic function that is bounded from above, i.e., $\exists \mathbf{M} < \infty$ such that $J(x) < \mathbf{M}, \forall x \in \chi$. We assume that Equation 3.2 has a unique global optimal solution.

We define a parameterized family of distributions $\{f(\cdot, \theta), \theta \in \Theta\}$ on the solution space, where Θ is the parameter space. At the kth iteration of the algorithm we use the distribution function $f(\cdot, \theta_k)$ to generate candidate solutions. The per-

formance of these sample solutions is used to find the parameter $\theta_{k+1} \in \Theta$ for the next iteration according to a parameter updating rule. If this rule is selected appropriately, future iterations of the method generate distribution functions that are more concentrated on better solutions. We continue the iterations until we reach a stopping criterion.

The parameter updating in MRAS uses another sequence of distributions $\{g_k(.)\}$, called the **reference** distributions. At each iteration of the MRAS method the new parameters θ_{k+1} are calculated by minimizing the Kullback-Leibler (KL) divergence $D(g_k, f(\cdot, \theta_k))$ of the reference distribution and the parameterized distribution:

$$D(g_k, f(\cdot, \theta_k)) := E_{g_k}\left[\ln \frac{g_k(X)}{f(X, \theta)}\right] = \sum_{x \in \chi} \ln \frac{g_k(x)}{f(x, \theta)} g_k(x), \tag{3.3}$$

where $E_{g_k}[\cdot]$ is the expectation with respect to $g_k(\cdot)$. The parameterized distribution $f(\cdot, \theta_{k+1})$ can be viewed as an approximation of the reference distribution $g_k(\cdot)$. Our selection of the reference distribution is mainly responsible for the performance of the algorithm.

The reference distribution selected in MRAS is constructed using the following scheme. An initial probability mass function (p.m.f.) $g_0(x) > 0, \forall x \in \chi$ is selected on the solution space χ. At each iteration $k \geq 1$, a new p.m.f. is calculated by tilting the old p.m.f. $g_{k-1}(x)$ with the cost function $J(x)$.

$$g_k(x) = \frac{H(J(x)) g_{k-1}(x)}{\sum_{x \in \chi} H(J(x)) g_{k-1}(x)}, \forall x \in \chi, \tag{3.4}$$

where $H : \Re \rightarrow \Re^{+}$ is a strictly decreasing function. Equation 3.4 assigns more weight to the solutions with lower cost. As a result, each iteration of this updating scheme improves the expected performance. i.e.,

$$E_{g_k}[H(J(X))] = \frac{E_{g_{k-1}}[H(J(X))^2]}{E_{g_{k-1}}[H(J(X))]} \geq E_{g_{k-1}}[H(J(X))] \tag{3.5}$$

or

$$E_{g_k}[J(X)] \leq E_{g_{k-1}}[J(X)] \tag{3.6}$$

It is also possible to show that $\{g_k(\cdot), k = 0, 1, \cdots\}$ converges to a distribution that concentrates only on the optimal solution, for an arbitrary initial reference distribution, $g_0(\cdot)$. Therefore, $\lim_{k\rightarrow\infty} E_{g_k}[H(J(X))] = H(J(x^*))$.

In Figure 3.7, we give the MRAS0 algorithm, also known as idealized version of MRAS, as stated in [20]. P_{θ_k} is the probability taken with respect to distribution $f(\cdot, \theta_k)$, i.e.,

$$P_{\theta_k}(J(X) \geq q) = \sum_{\chi} I\{J(X) \geq q\} f(x, \theta_k). \tag{3.10}$$

This version of the algorithm assumes that objective function $J(\cdot)$ is deterministic and the expectations on the solution space can be evaluated exactly.

It can be shown ([20]) that the solution found by MRAS0, is in fact the optimal solution. However the idealized version of MRAS is not very useful as it can be applied only to problems with small solution spaces, where it is more efficient to find the optimal solution directly.

In the case of our topology space this is feasible for small networks, where we can calculate the expectations for all possible topologies with the given number of

MRAS0 algorithm

Input: $\rho \in (0,1]$, $\epsilon \geq 0$, strictly decreasing function $H : \Re \to \Re^+$, family of distributions $\{f(\cdot,\theta)\}$, with θ_0 such that $f(x,\theta_0) > 0 \forall x \in \chi$.

Initialization: Set iteration count $k = 0$.

Loop until the stopping criterion is satisfied:

1- Calculate the $(1-\rho)$-quantile:

$$q_k = \sup_l \{l : P_{\theta_k}(J(X) \leq l) \geq \rho\}. \tag{3.7}$$

2- Update elite threshold:

$$\bar{q}_k = \begin{cases} q_k & \text{if } k = 0 \text{ or } q_k \geq \bar{q}_{k-1} + \epsilon; \\ \bar{q}_{k-1} & \text{otherwise.} \end{cases} \tag{3.8}$$

3- Update parameter vector:

$$\theta_{k+1} = \arg\min_{\theta \in \Theta} E_{\theta_k} \left[\frac{[H(J(X))]^k}{f(X,\theta_k)} I\{J(X) \leq \bar{q}_k\} \ln f(X,\theta_k) \right] \tag{3.9}$$

4- $k \leftarrow k+1$

Output: θ_k

Figure 3.7: Algorithm MRAS0

nodes and degree sequence. For example we can apply the MRAS0 algorithm, also known as idealized MRAS, to a network of 6 nodes, each having 3 free interfaces. But in such small networks we can directly find the best topology by just comparing the costs for each topology (there are 70 combinations for a 6 node network).

Figure 3.8 shows the MRAS1 algorithm as given in [20], also known as the Monte Carlo version of MRAS. This version of the algorithm is the stochastic counterpart of MRAS0, in which only a limited number of samples are used in each iteration and expected values in the algorithm are replaced by their corresponding sample averages. For instance, the parameter updating rule is replaced with

$$\tilde{\theta}_{k+1} = \arg\max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \frac{[H(J(X_i))]^k}{f(X_i,\tilde{\theta}_k)} I\{H(J(X_i)) \le \tilde{q}_{k+1}\} \ln f(X_i,\theta), \qquad (3.11)$$

where X_is are i.i.d. random samples generated using $f(x,\tilde{\theta}_k)$, $\tilde{\theta}_k$ is the estimated parameter vector computed at the previous iteration, and $\tilde{q}_{k+1}$ is a threshold determined by the sample $(1-\rho)$-quantile of $H(J(X_1)),\cdots,H(J(X_N))$.

The theoretical convergence of the algorithm is no longer guaranteed, and neither is the convergence to the optimal point. For example, the set $\{x : J(x) < \tilde{q}_{k+1}\}$ turns out to be empty if the sample solutions generated at the current iteration are much worse than the samples of the last iteration. Or if the optimal solution to the problem is never generated in any of the random sample sets, it may never be discovered. The former problem is resolved by adaptively increasing the sample size N, and adaptively decreasing parameter ρ. The latter problem may not be resolved completely, so the MRAS1 algorithm may terminate in a suboptimal solution. But in practice the solutions found by this algorithm are very close to, if not the optimal

solution.

We now turn our attention to the specific problem of static topology optimization, where we want to find the best topology given a constant traffic pattern. As defined before, χ is the set of all topologies and cost function $J(\cdot)$ is defined on this set and the constant traffic pattern is embedded in the cost function. The objective is to find the topology that minimizes the cost function.

$$\min_{x \in \chi} J(x) \tag{3.14}$$

In order to implement the MRAS algorithm for our topology optimization problem, we need to be able to generate random topologies according to a parameterized probability distribution, and then be able to evaluate the sample topologies and update the parameters of the distribution. We start by a probability matrix P constructed in Section 3.3 and we show how to generate random topologies in a way that the updating formula is easily determined. For the sake of easier notation, we assume that the random topology we are trying to build has N nodes and each node has exactly d interfaces (regular connected labeled graph)., matrix P has an $N \times N_c$ dimensions, where $N_c = \binom{N}{d}$. The TSP problem becomes a special case of this problem with $d = 2$. Even though we have a more general form of TSP, our optimization problem is not the same as a generalized TSP as explained in [44], and [63].

We first assume that our connections are unidirectional and in order to create a bidirectional connection between nodes i and j, we need to set up two connections, one from i to j and one from j to i. We select the connections from node $i = 0, \cdots, N$

MRAS1 algorithm

Input: $\rho \in (0,1]$, and initial sample size $N_0 > 1$, $\epsilon \geq 0$, $\alpha > 1$, a mixing coefficient $\lambda \in (0,1]$, strictly decreasing function $H : \Re \to \Re^+$, family of distributions $\{f(\cdot,\theta)\}$, with θ_0 such that $f(x,\theta_0) > 0 \forall x \in \chi$.

Initialization: Set iteration count $k = 0$ and $\tilde{\theta}_0 = \theta_0$.

Loop until the stopping criterion is satisfied:

1- Generate N_k i.i.d. samples $X_1^k, \cdots, X_{N_k}^k$ according to $\tilde{f}(\cdot,\tilde{\theta}) := (1-\lambda) f(\cdot,\tilde{\theta}_k) + \lambda f(\cdot,\theta_0)$.

2- Calculate the $(1-\rho)$-quantile:

$$\tilde{q}_{k+1}(\rho_k, N_k) := H_{(\lceil (1-\rho_k) N_k \rceil)}, \tag{3.12}$$

where $\lceil a \rceil$ is the smallest integer greater than a and $H_{(i)}$ is the ith order statistic of the sequence $\{H(J(X_i^k)), i = 1, \cdots, N_k\}$.

3- if $k = 0$ or $\tilde{q}_{k+1}(\rho_k, N_k) \geq \tilde{q}_k + \frac{\epsilon}{2}$, then

[3a.] Set $\tilde{q}_{k+1} \leftarrow \tilde{q}_{k+1}(\rho_k, N_k), \rho_{k+1} \leftarrow \rho_k, N_{k+1} \leftarrow N_k$.

else, find the largest $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{q}_{k+1}(\bar{\rho}, N_k) \geq \tilde{q}_k + \frac{\epsilon}{2}$.

[3b.] If such a $\bar{\rho}$ exists, then set $\tilde{q}_{k+1} \leftarrow \tilde{q}_{k+1}(\bar{\rho}, N_k), \rho_{k+1} \leftarrow \bar{\rho}, N_{k+1} \leftarrow N_k$.

[3c.] else (if no such $\bar{\rho}$ exists), set $\tilde{q}_{k+1} \leftarrow \tilde{q}_k, \rho_{k+1} \leftarrow \rho_k, N_{k+1} \leftarrow \lceil \alpha N_k \rceil$.

endif

4- Update parameter vector:

$$\tilde{\theta}_{k+1} = \arg\max_{\theta \in \Theta} \frac{1}{N_k} \sum_{i=1}^{N_k} \frac{[H(J(X_i^k))]^k}{\tilde{f}(X_i^k, \tilde{\theta}_k)} I\{H(J(X_i^k)) \geq \tilde{q}_{k+1}\} \ln f(X_i^k, \theta_k) \tag{3.13}$$

4- $k \leftarrow k+1$

Figure 3.8: Algorithm MRAS1

to other nodes based on the probability distribution represented by row P_i. We denote the set of all graphs that can be constructed in this fashion $\tilde{\chi}$. Note that if for every connection $i \to j$, we happen to have a connection $j \to i$, the resulting graph meets our definition of a topology. We can construct all possible topologies using this method, i.e. $\chi \subset \tilde{\chi}$.

We define a new cost function $\tilde{J}(\cdot)$ on $\tilde{\chi}$ as follows:

$$\bar{J}(\tilde{x}) = \begin{cases} J(x) & \text{if } x \in \chi; \\ \infty & \text{otherwise.} \end{cases} \tag{3.15}$$

Then obviously Equation 3.14 is equivalent to

$$\min_{x \in \tilde{\chi}} \tilde{J}(x). \tag{3.16}$$

Now we define a probability distribution function $f(\cdot, P)$, parameterized by the matrix P:

$$f(\cdot, x) = \prod_{i=1}^{N} \sum_{j=1}^{N_c} p_{ij} I\{x \in \tilde{\chi}_{ij}\}, \tag{3.17}$$

where $\tilde{\chi}_{ij}$ represents all graphs in which node i is connected to the connect subset corresponding to column j of the matrix P. The updating rules for this modified problem follow from applying Lagrange multipliers to Equation 3.11. Parameter θ is replaced with probability matrix P, and imposing the constraint that each row of matrix P sums up to 1. P_{k+1} is the solution to the following optimization problem.

$$\max_{P} \min_{u} \Big[\frac{1}{N_k} \sum_{n=1}^{N_k} \frac{[H(J(X_n))]^k}{f(X_n, P_k)} I\{H(J(X_n)) \le \tilde{q}_{k+1}\} \ln f(X_n, P) + \sum_{i=1}^{N} u_i \Big(\sum_{j=1}^{N_c} p_{ij} - 1 \Big) \Big]$$

where, N_k is the number of sample topologies generated using P_k in stage k of MRAS1, and $u = (u1, \cdots, u_N)$ is the vector of Lagrange multipliers. Differentiating the expression within square brackets above with respect to p_{ij}, yields, for all $j = 1, \cdots, N_c$,

$$\frac{\frac{1}{N_k}\sum_{n=1}^{N_k}\frac{[H(J(X_n))]^k}{f(X_n,P_k)}I\{H(J(X_n)) \leq \tilde{q}_{k+1}\}I\{X_n \in \tilde{\chi}_{ij}\}}{p_{ij}} + u_i = 0. \tag{3.18}$$

Summing over all $j = 1, \cdots, N_c$,

$$\frac{1}{N_k}\sum_{n=1}^{N_k}\frac{[H(J(X_n))]^k}{f(X_n,P_k)}I\{H(J(X_n)) \leq \tilde{q}_{k+1}\} = -u_i \tag{3.19}$$

because $\sum_j I\{X_n \in \tilde{\chi}_{ij}\} = 1$, since node i has to be connected to one and only one of the possible connect subsets. Therefore, combining Equations 3.18 and 3.19, gives

$$p_{ij} = \frac{\sum_{n=1}^{N_k}\frac{[H(J(X_n))]^k}{f(X_n,P_k)}I\{H(J(X_n)) \leq \tilde{q}_{k+1}\}I\{X_n \in \tilde{\chi}_{ij}\}}{\sum_{n=1}^{N_k}\frac{[H(J(X_n))]^k}{f(X_n,P_k)}I\{H(J(X_n)) \leq \tilde{q}_{k+1}\}} \tag{3.20}$$

Theoretically, we can carry out the sample generation and parameter updating this way. However, we will not do so, since the majority of graphs generated using the above method have a cost of ∞ and are not valid topologies. Instead, we use the topology generation algorithm of Section 3.3, to speed up the process of topology generation. Since we now generate only valid topologies, we can replace $\tilde{\chi}$ with χ in Equation 3.20.

Figure 3.9 shows the details of the MRAS algorithm for the problem of finding the best topology for a given traffic matrix. The solutions found by this algorithm are compared to our MMDP solutions in the next chapter.

MRAS1 algorithm for topology optimization

Input: Traffic matrix, $\rho \in (0,1]$, and initial sample size $N_0 > 1$, $\epsilon \geq 0$, $\alpha > 1$, a mixing coefficient $\lambda \in (0,1]$, strictly decreasing function $H : \Re \to \Re^+$, family of distributions $\{f(\cdot, P)\}$, with P_0 being the uniform distribution.

Initialization: Set iteration count $k = 0$ and $\tilde{P}_0 = P_0$.

Loop until the stopping criterion is satisfied:

1- Generate N_k i.i.d. samples $X_1^k, \cdots, X_{N_k}^k$ using the topology generation algorithm according to $\tilde{P} := (1-\lambda)P_k + \lambda P_0$.

2- Using the given traffic matrix calculate the cost for each of the sample topologies.

2- Calculate the $(1-\rho)$-quantile:

$$\tilde{q}_{k+1}(\rho_k, N_k) := H_{(\lceil (1-\rho_k)N_k \rceil)}, \tag{3.21}$$

where $\lceil a \rceil$ is the smallest integer greater than a and $H_{(i)}$ is the ith order statistic of the sequence $\{H(J(X_i^k)), i = 1, \cdots, N_k\}$.

3- if $k = 0$ or $\tilde{q}_{k+1}(\rho_k, N_k) \geq \tilde{q}_k + \frac{\epsilon}{2}$, then

[3a.] Set $\tilde{q}_{k+1} \leftarrow \tilde{q}_{k+1}(\rho_k, N_k), \rho_{k+1} \leftarrow \rho_k, N_{k+1} \leftarrow N_k$.

else, find the largest $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{q}_{k+1}(\bar{\rho}, N_k) \geq \tilde{q}_k + \frac{\epsilon}{2}$.

[3b.] If such a $\bar{\rho}$ exists, then set $\tilde{q}_{k+1} \leftarrow \tilde{q}_{k+1}(\bar{\rho}, N_k), \rho_{k+1} \leftarrow \bar{\rho}, N_{k+1} \leftarrow N_k$.

[3c.] else (if no such $\bar{\rho}$ exists), set $\tilde{q}_{k+1} \leftarrow \tilde{q}_k, \rho_{k+1} \leftarrow \rho_k, N_{k+1} \leftarrow \lceil \alpha N_k \rceil$.

endif

4- Update parameter vector:

$$p_{ij} = \frac{\sum_{n=1}^{N_k} \frac{[H(J(X_n))]^k}{f(X_n, P_k)} I\{H(J(X_n)) \leq \tilde{q}_{k+1}\} I\{X_n \in \chi_{ij}\}}{\sum_{n=1}^{N_k} \frac{[H(J(X_n))]^k}{f(X_n, P_k)} I\{H(J(X_n)) \leq \tilde{q}_{k+1}\}} \tag{3.22}$$

4- $k \leftarrow k+1$

Figure 3.9: Algorithm MRAS1 for the topology optimization problem

Chapter 4

Numerical results

This chapter is divided into two sections. In section 4.1 the performance of the random graph generation algorithm is demonstrated for networks with various sizes. In section 4.2 the performance of the logical topology optimization algorithms are evaluated for both the static and dynamic traffic cases.

4.1 Random Graph Generator

The random graph generation algorithm was implemented and the runtime of the algorithm for graphs of various sizes was measured. The simulations were limited to regular graphs, mainly because it is much easier to summarize the results for regular graphs than for graphs with irregular degree sequences. Figure 4.1 shows the runtime of the uniform graph generator algorithm, generating 100000 graphs in every instance, for graphs with up to 50 nodes. The runtime is calculated for different regularity of the graph (d).

To explain the behavior seen in Figure 4.1, we need to refer back to the algorithm in Figure 3.5. As was explained in the proof of Theorem 3.2, there are N iterations of the algorithm and Steps 2 and 3 of the algorithm have complexity of $d\log(d)$ and d respectively. However, $d\log(d)$ becomes visible only when N gets large. For small N, the number of branches at each node of the partition tree is

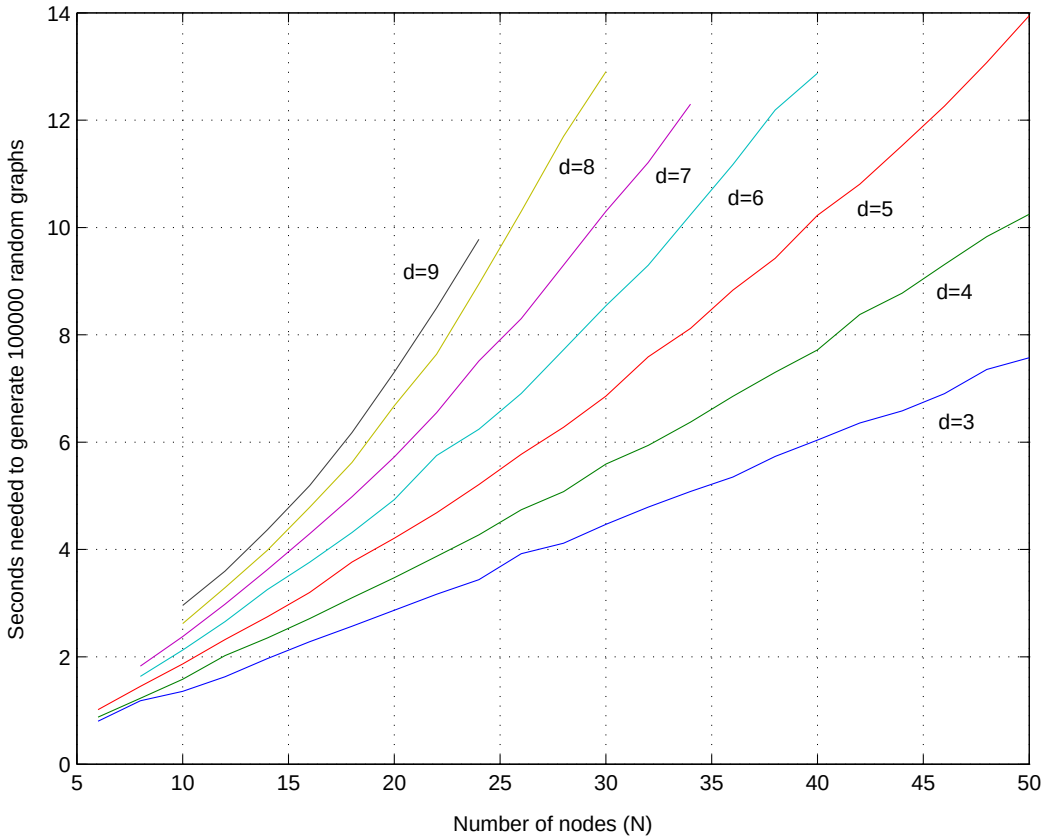

Figure 4.1: Runtime for generating 100000 random graphs.

very small and the runtime of Step 2 becomes negligible. Therefore, for small N, we expect a runtime of $\mathcal{O}(Nd)$, and as N gets larger we expect the runtime to be $\mathcal{O}(Nd)$. This explains why the runtime of the algorithm does not increase linearly with N, when $d >= 5$. For $d = 3, 4$ the term $\log(d)$ is very small and therefore, the runtime of the Steps 2 and 3 become very close to each other. As a result for small d the runtime grows almost linearly.

Figure 4.2 shows the runtime of the algorithm versus the regularity of the graphs (d). Here also, the nonlinear growth of $Nd\log(d)$ is more pronounced for larger N.

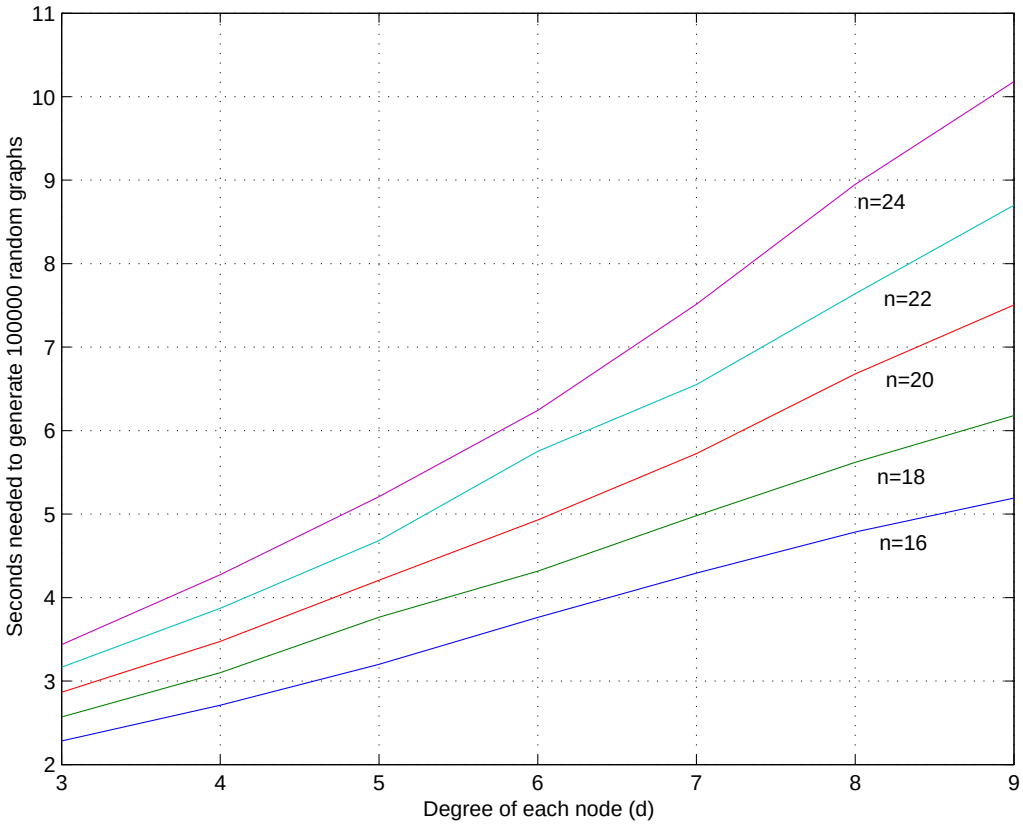

Figure 4.2: Runtime for generating 100000 random graphs.

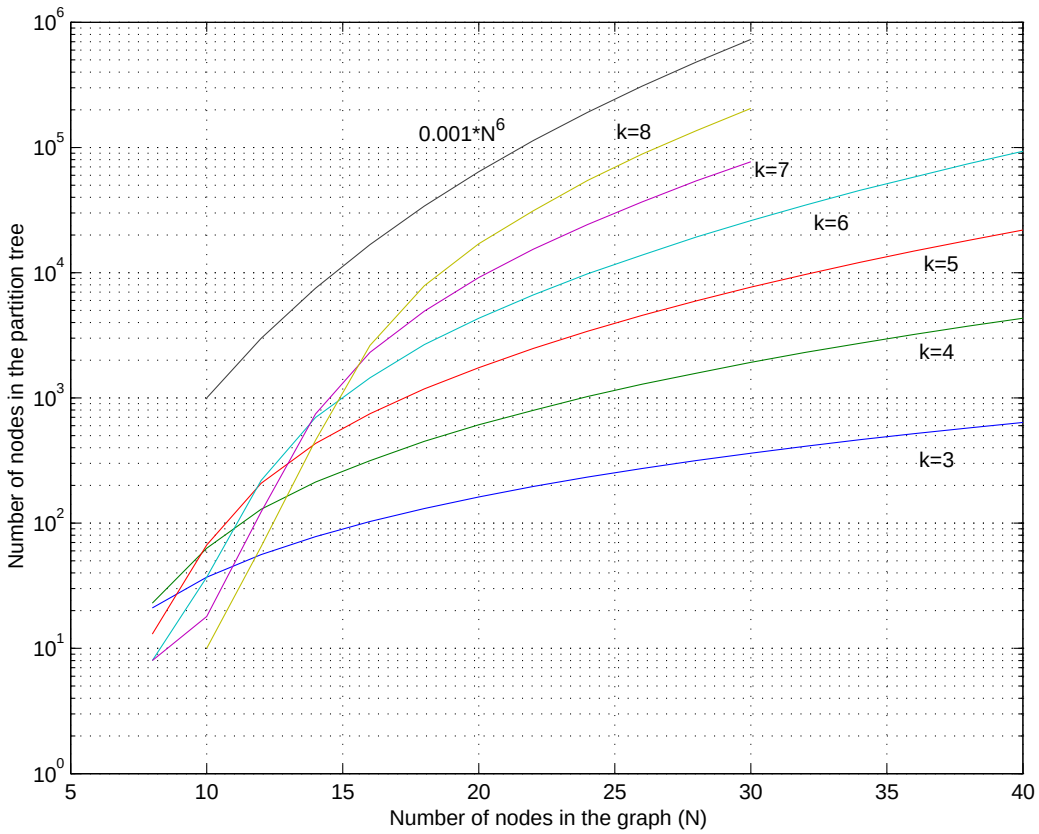

Figure 4.3: Number of database entries needed for the partition tree.

Figure 4.3 shows the number of nodes in the partition tree on a log scale, versus the number of nodes in the graph. This is directly related to the amount of memory needed to save the partition tree. The graph shows the partition tree seems to have a polynomial growth, and the degree of the polynomial increases as the regularity (d) of the graph increases.

4.2 Performance of the topology optimization algorithms

In this section we examine different algorithms and the effect of design parameters on the performance of the network using simulation results. We divide the simulation results into two subsections. Section 4.2.1 goes into detail about the simulations with fixed traffic. Section 4.2.2 examines important design problems when there is dynamic traffic involved.

The following parameters have been used for all the simulations throughout this section; unless stated otherwise, the network is selected to have 10 nodes and each node has 3 interfaces. Each interface is assumed to have a capacity of 36000. In the fast time scale cost function (Equation 2.2) β is selected to be 1×10^{-5}.

In the dynamic case we consider two types of traffic. The first type is generated randomly and the second type is real data taken from the Abilene network (see [1]) traffic during a 5-week period as explained in Section 4.2.3. The dynamic traffic patterns are also scaled to create periods of congestion in the network.

4.2.1 Fixed Traffic

Fixed traffic patterns are generated by first picking a real number between 0 and 1 for each S-D pair, where this number represents the ratio of the traffic demand of each S-D pair to a reference demand. Then by selecting the reference demand we can scale the traffic pattern to one with various congestion levels in the network. There are three traffic levels that are considered here. In the light traffic case there is no topology change required to be able to accommodate all the traffic. In this

case the changes made to the topology seek to improve the utilization of the the network. In the moderate traffic case, we scale the network traffic such that if we take no actions during the simulation, we might lose some traffic. However, we can easily get to a topology that incurs no loss of traffic. In the heavy traffic case we have a very congested network, and without a few topology changes we are going to incur a lot of traffic loss. 100 sample traffic patterns were generated for each of the light, moderate, and heavy traffic cases. These samples were used throughout this section as input to our simulations.

4.2.1.1 Fast time scale policy evaluation using MRAS results

The fast time scale policy is evaluated in this section. There are three policies that are used.

- The optimal solution to the fast time scale problem, that is found by solving the quadratic programming problem of routing traffic through different LSPs available for each S-D pair;
- The heuristic of Figure 2.8;
- The single LSP case, where each S-D pair is restricted to a single LSP.

The single LSP case, in effect, eliminates the fast time scale, since there is no decision to be made about how to distribute the traffic among LSPs of each S-D pair. This is equivalent to the shortest path routing of each S-D pair, since in this case the LSP with the least number of hops is selected for each S-D pair.

Traffic type	Optimal	Heuristic	Ratio to Opt	Single LSP	Ratio to Opt
Heavy	55565.8	58258.9	1.048	60570.0	1.090
Moderate	39227.6	41513.4	1.058	42601.9	1.086
Light	25960.7	27461.1	1.058	28298.7	1.090

Table 4.1: Comparing the fast time scale heuristic to the optimal and single LSP solutions.

The MRAS method was used to search for the best topology for each of the sample traffic patterns. The solutions in this section are going to be used as reference solutions for the following sections where slow time scale policies are evaluated. Table 4.1 compares the fast time scale heuristic to the best and worst case scenarios, i.e., the optimal case and the single LSP case. The tabulated results show that the single LSP case is about 8.8% worse than the optimal solution and the fast time scale heuristic is about 5.5% worse than the optimal solution.

Obviously, the single LSP case does not involve any calculations. Quadratic programming problem takes about 3 times longer than the heuristic algorithm. If there is some computing power to run the heuristic algorithm, we can do some load balancing to improve the congestion in the network. If there is more time to solve a quadratic programming problem at the fast time scale, we take full advantage of the load balancing at the fast time scale.

Samples per iteration	Average Deviation from optimal cost	Average ranking	Matched the optimal solution
20	5.15%	28.8	0
50	1.59%	4.7	30%

Table 4.2: Comparing the MRAS solution to the optimal solution.

4.2.1.2 Evaluating the MRAS solution

In this section the performance of the MRAS solution is evaluated compared to the optimal topology. It is not tractable to do this comparison for networks of 10 or more nodes, as the number of possible topologies becomes extremely large. Therefore, in this section we consider a network of 8 nodes only. For an 8 node network with each node having a degree of 3, there are 19320 possible topologies. A full search of these topologies results in the optimal topology. The number of samples generated in each iteration of the MRAS is a parameter that affects the accuracy of the final solution. Obviously, the larger the sample size at each iteration, the better the result is going to be. Simulations were performed using two different values of 20 and 50 samples per iteration for 50 different traffic patterns. Table 4.2 shows the average ranking of the MRAS solution. Note that the ranking in this case is a number between 1 and 19320.

For the 10 node network we use the number of samples to be 1000.

4.2.1.3 Comparing heuristic and rollout algorithms

In this section the performance of the heuristic and rollout algorithms are compared to each other and to the MRAS solutions. The traffic samples used in this experiment are the same samples referenced at the beginning of Section 4.2.1 and the initial topologies are selected at random. The greedy heuristic algorithm for the slow time scale performed extremely well compared to both rollout and MRAS. This can be seen from the summary of results in Table 4.3. The elements of this table are determined as follows: For each instance of the traffic pattern the best and worst topologies were found using the MRAS method. The average cost of those topologies over all 300 instances of the traffic pattern makes up the average min and max. The heuristic and rollout algorithms were executed for the same instances and the corresponding cost of the solutions found by the each policy was averaged to make the remaining elements of the average row. Then we normalized the average row such that the min and max are set to 0 and 100 respectively. We can see that the heuristic and rollout policies generate solutions that are only 0.55% and 0.12% off of the minimum cost. It should be noted that rollout policy, in some cases, returned solutions that had a better cost than the MRAS solutions. In those cases the rollout solution replaced the MRAS solution such that the min value given in the table 4.3 is the actual minimum found by either of the methods. This was mainly done to avoid negative numbers in the normalized row.

Figure 4.4 is composed of 10 different graphs corresponding to 10 sample traffic patterns. Each graph for the heuristic and rollout shows the progression of the

	Min	Max	Heuristic	Rollout
Average	42224.7	151179.5	42826.0	42358.9
Normalized average	0	100	0.5585	0.1197

Table 4.3: Comparing performance of heuristic and rollout.

heuristic and rollout algorithms from the cost of the initial topology until they reach a local minimum. We have allowed 10 steps for each run of the heuristic or rollout to find the corresponding end point. For each of those simulations the initial topology is the same for heuristic and rollout. For that reason the heuristic and rollout graphs start from the same random starting point (at step 0,10, 20 ,...). No action policy is one in which no action is taken. As a result for the no action policy the initial topology does not change and neither does the corresponding cost (for constant traffic case). The graph for no action is also provided for reference.

The excellent performance of the heuristic algorithm prompted further investigation of the situation. Our simulations showed that given any initial topology for a 10 node network, there are approximately 60 admissible branch exchanges. Consider the initial topology to be the worst topology for the given traffic pattern. In that case all 60 BEs result in a better topology. As the heuristic algorithm makes the decision for each step and selects a new topology, a smaller fraction of the 60 BEs for the new topology results in a better topology. The heuristic algorithm stops when it reaches a local minimum, which is equivalent to a topology for which all admissible branch exchanges result in a topology with worse cost.

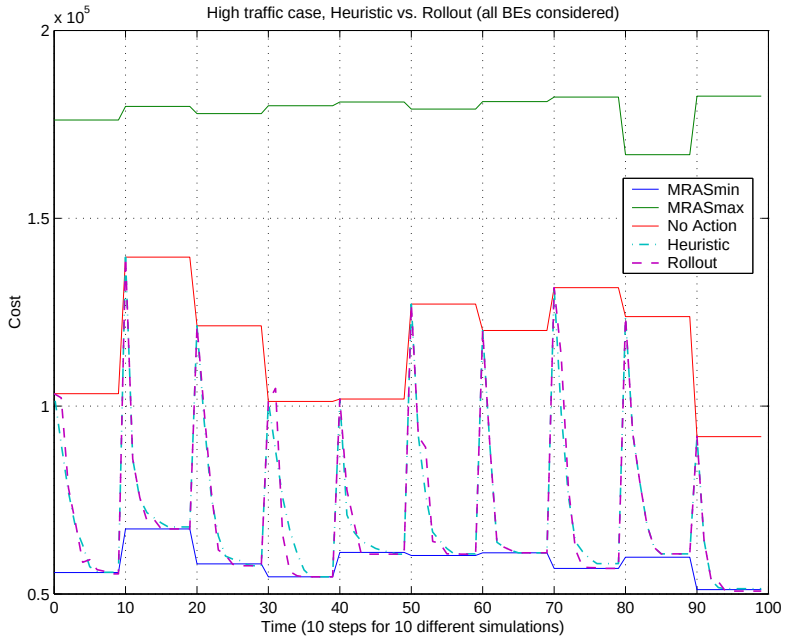

Figure 4.4: Graphical comparison of the performance of heuristic and rollout policies.

In the next experiment we use the method described in Section 2.4.5.3 with parameters $\gamma_l = 0.05$ and $\gamma_h = 0.80$, to reduce the number of admissible branch exchanges for any given topology. As a result, the direction of the search for the heuristic and rollout algorithms becomes more limited and there is a bigger chance that we run out of BEs that lead to topologies with better cost. This is illustrated in Figure 4.5, for the same traffic patterns of Figure 4.4. Clearly, the heuristic algorithm for the more restrictive set of BEs is more improved by the rollout than in the previous case. This is an indication that in cases where external restrictions are placed on the set of admissible branch exchanges (such as wavelength assignment constraints or optical impairments pointed out in Section 1.2), the rollout policy is more useful than in the case with no restrictions on BEs.

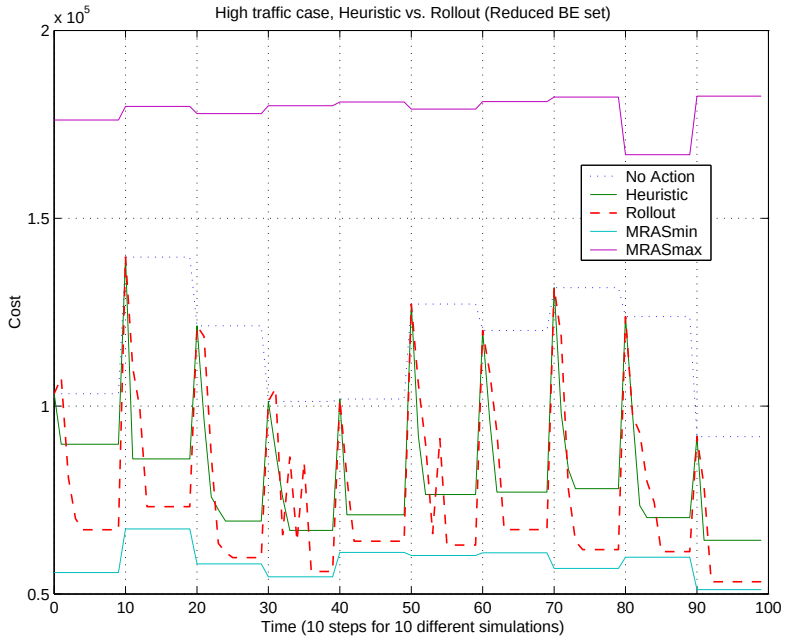

Figure 4.5: Graphical comparison of the performance of heuristic and rollout policies for restricted admissible action set.

4.2.2 Dynamic Traffic

This section summarizes the results for the case of dynamic traffic. In the presence of dynamic traffic the notion of an optimum solution becomes more elusive. This is mainly because the solution that is composed of the optimum solutions for each step cannot be the optimum solution for the overall problem, since with the limitation of one Branch Exchange per step we cannot reach from one topology to another topology if they are more than one BE away from each other and the transitional costs of going from one topology to another become important. In the absence of the MRAS solutions, the performance of the heuristic and rollout policies can only be compared with each other.

4.2.3 Real traffic data

In part of this study we use real data collected from the Abilene network. we have collected 5 weeks of network traffic from this network. We averaged 4 weeks of data to represent the weekly behavior of the network and used the 5th week as the input to our simulation. The 4-week average is used to represent the deterministic part of the traffic, and the random part of traffic is the difference between the value of the 5th week of data and the value of the 4-week average for the same time of week. The sum of the two results in the amount of traffic in the 5th week of data.

Figure 4.6 compares the performance of the heuristic and rollout algorithms for the case in which all BEs are considered. As expected, the performance of the two policies are so close that the two graphs are not distinguishable. The no action policy shows the cost if the topology of the network was fixed at the initial topology. Figure 4.7 shows the same graph for the case of a reduced action set. In the latter case the improvement of the rollout over the heuristic is much more obvious than in the first case.

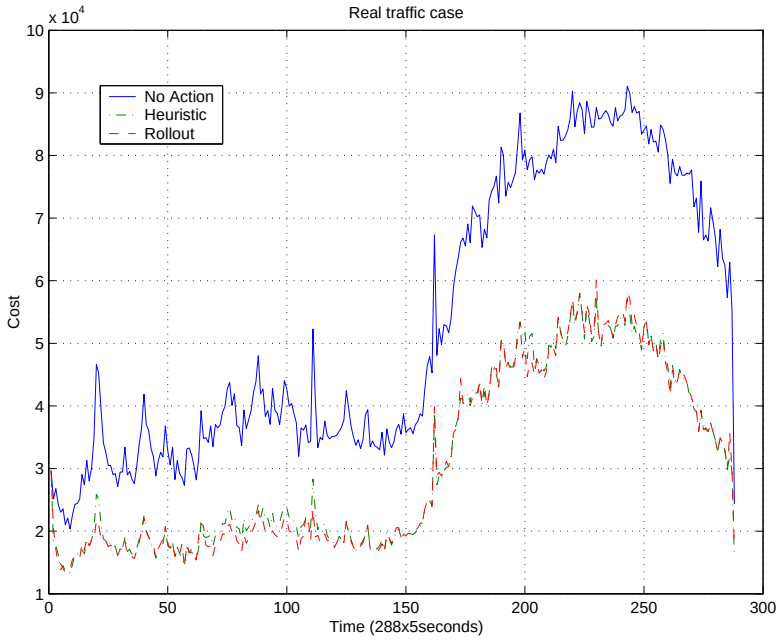

Figure 4.6: Graphical comparison of the performance of heuristic and rollout policies for the case with real data.

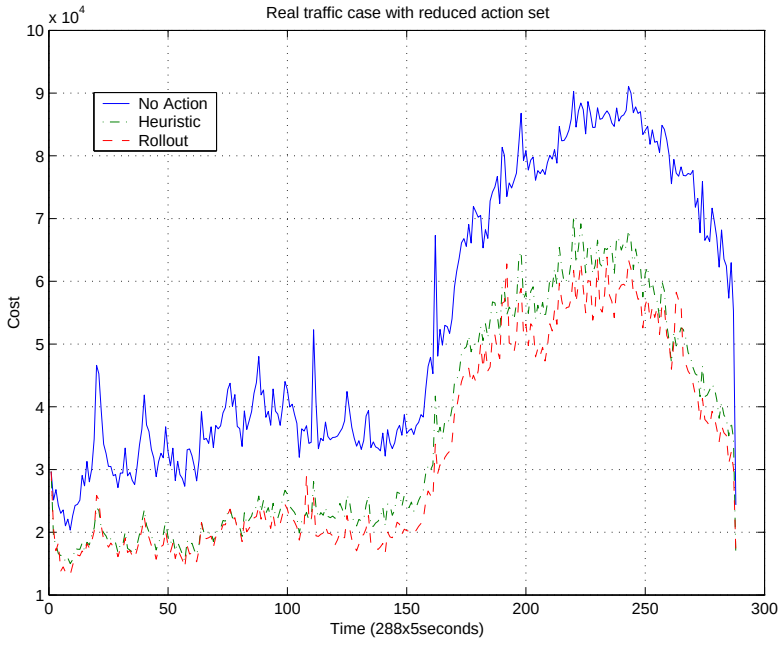

Figure 4.7: Graphical comparison of the performance of heuristic and rollout policies for the case with real data and restricted admissible action set.

Chapter 5

Concluding remarks

The goal of this thesis has been to develop a framework that will consider multi-path routing and logical topology reconfiguration simultaneously and will find a network configuration (topology and routing) that is suitable for the traffic behavior (static or dynamic) of the network. There are several parameters that shape the network design problem and determine whether the approach in this thesis is effective or not. The important parameters include but are not limited to the following list:

1. Traffic demand
2. Size of the network
3. Type of action available for network reconfiguration (e.g., BE)
4. Number of topologies reachable from the current topology using the admissible action set
5. Amount of available resources
6. Type of the cost function

The most important factor in determining how to approach the network design problem is the type of traffic. When the changes in the traffic are infrequent and

predictable using the past behavior, a predictive approach such as the MMDP model is useful. On the other hand, if the traffic is frequently changing and the changes are unpredictable, a more reactionary model such as a heuristic is recommended. If the traffic is static, then the calculations for optimizing the network configuration can be performed offline and there is no need for allowing dynamic changes to the network.

The size of the network plays a major role in determining the strategy for reconfiguring the network. For very large networks, most probably a very localized approach is more useful than trying to optimize the entire network at once. However, if the very large network under discussion has a traffic demand that has minor changes compared to the size of the traffic for the entire network, then a conceivable approach is to define a core network that has a constant configuration, and allow a small portion of the nodes or their interfaces to be dynamically configurable. With this approach, the core of the network is responsible for serving the traffic demand that is always there and the dynamic part of the network will be adjusted to fit the minor changes in the traffic.

Another very important factor is the type of actions available to the decision maker. In this thesis the actions were limited to branch exchanges. It is possible to allow single deletion or setup of a lightpath as the primary actions. It is also possible to allow more than one BE at a time, which increases the search space dramatically. However, decreasing the slow time scale step will have similar effect.

The size of the search space at every slow time scale step determines whether it is practical to use a rollout algorithm or not. Use of rollout becomes computationally

intractable if the size of the search space is too large. We also saw that when the the number of available BEs is large, rollout is not as useful as in the case when there is a limited number of BEs available.

The next item in the list is the type of cost function, which determines whether an exact solution is possible or an approximate solution is the best that can be expected. In this thesis we adopted a quadratic function and were able to find an exact solution for the fast time scale problem. A cost function with more nonlinearity will make it harder to find an exact solution.

Finally, one of the key assumptions here is that the capacity of the network infrastructure is in the same order of magnitude as the peak traffic demand. If the capacity of the network is orders of magnitude higher than the traffic demand, then the type of cost function defined in this thesis is not the right cost function and the assumption that all interfaces of a node are used may not make sense.

The main objective of the thesis was to find a solution to the multi-path routing and topology optimization problem for WDM networks. This was accomplished by applying the MMDP model to this problem. Heuristics were defined for the fast and slow time scale problem, an exact solution was found for the fast time scale problem, and for the slow time scale problem rollout was used to improve the heuristic.

For the static problem the MRAS method was used to find a reference solution that is very close to optimal. The implementation of the MRAS algorithm for this problem is a contribution to the field of static optimization, and the random topology generation algorithms presented in this thesis are contributions to the graph theory literature.

Two random topology generation algorithms were introduced. One generates a uniform distribution and the other allows a probability matrix to bias the distribution. The uniform random topology generator has a precalculation step to build a partition tree that helps speed up the random topology generation. A compact partition tree is introduced that uses the repetition of the same degree sequences in the tree to reduce the amount of memory needed to save this tree. The compact partition tree allows building and saving such a tree for networks (graphs) of moderate size (200 nodes for 3-regular graphs, 50 nodes for 6-regular graphs, 30 nodes for 10-regular graphs, and so on). Once this precalculation is complete, fast generation of random graphs with the specified degree sequence is possible. The runtime of this algorithm is of $\mathcal{O}(Nd\log(d))$, and every run of this algorithm generates a valid graph with probability 1. Compared to pairing model methods in the literature, this is a marked improvement.

The topology search using the slow time scale heuristic and rollout leads to solutions very close to MRAS solutions. Our results show that the greedy heuristic is a very effective tool for finding topologies very close to optimal in the static traffic case. In the case where all the branch exchanges are available, even though rollout improves the heuristic it does so with a very small margin. But with the restricted action space, the heuristic performs much worse and rollout improves the heuristic by a large margin and makes the result close to optimal.

5.1 Future work

There are a few directions for future work to be pursued.

1. The size of the database needed for the partition tree limits the use of the uniform random graph generator to graphs with up to 200 nodes (for 3-regular graphs, but much less for graphs with larger regularity or irregular graphs). It is conceivable that our algorithm can be combined with one of the existing algorithms (pairing model) to extend the use of this algorithm to larger graph sizes.

2. The biased random graph generator given in Section 3.3 does not generate a sample that matches the distributions in matrix P. Generating a sample that matches the rows of matrix P remains a challenge.

3. Extending the MMDP solutions derived in this thesis to other practical network models is another interesting topic for future work.

Bibliography

[1] Abilene Backbone network, http://abilene.internet2.edu/

[2] A. Arapostathis, V. S. Borkar, E. Fernandez-Gaucherand, M. K. Ghosh, and S. I. Marcus, "Discrete-time controlled Markov processes with average cost criterion: a survey," SIAM J. Control and Optim., Vol. 31, 1993, pp. 282344.

[3] C. Assi, Y. Ye, A. Shami, S. Dixit, I. Habib, and M.A.Ali, "On The Merit Of IP/MPLS Protection/Restoration in IP over WDM networks," Globecom, 2001

[4] I. Baldine and G.N. Rouskas, "On the Design of Dynamic Reconfiguration Policies for Broadcast WDM Networks," Proceedings of SPIE '98 Conference on All-Optical Networking: Architecture, Control, and Management Issues, Vol. 3531, pp. 146–157.

[5] I. Baldine and G.N. Rouskas, "Reconfiguration and Dynamic Load Balancing in Broadcast WDM Networks." Photonic Network Communications, Vol. 1, No. 1, Jun. 1999, pp. 49–64.

[6] D. Banerjee and B. Mukherjee, "Wavelength-Routed Optical Networks: Linear Formulation, Resource Budgeting Tradeoffs, and a Recofiguration Study," IEEE/ACM Trans. on Networking, Vol. 8, No. 5, Oct. 2000, pp. 598–607.

[7] T. Başar and G. J. Olsder, "Dynamic Noncooperative Game Theory," Academic Press, London/New York, 1995.

[8] M. Bayati, J. H. Kim, and A. Saberi, "A Sequential Algorithm for Generating Random Graphs," 2007),priliminary version.

[9] E.A. Bender and E.R. Canfield, "The asymptotic number of labeled graphs with given degree sequence," J. Combinatorial Theory Ser., 1978, A 24, 3, pp. 296–307.

[10] D. Bertsekas, "Dynamic Programming and Optimal Control," 2nd edition, Athena Scientific, 2000.

[11] D. Bertsekas and R. Gallager, "Data Networks," Prentice Hall, 1992.

[12] D. Bertsekas and D.A. Castañon, "Rollout Algorithms for Stochastic Scheduling Problems," Proceedings of the 37th IEEE CDC.

[13] D. Bienstock and O. Gunluk, "Computational experience with a difficult mixed-integer multicommodity flow problem," Mathematical Programming, Vol. 68, pp. 213–237, 1995.

[14] B. Bollobás, "A probabilistic proof of an asymptotic formula for the number of labelled regular graphs," European Journal of Combinatorics, 1980, Vol. 1, No. 4, pp. 311–316.

[15] B. Bollobás, "Graph Theory, an introductory course," Springer-Verlag,1979

[16] B. Bollobás, "Random Graphs," 2nd edition, Cambridge university press,2001

[17] E. Bouillet, J. Labourdette, R. Ramamurthy, and S. Chaudhuru, "Lightpath Re-optimization in Mesh Optical Networks," NOC'02, Darsmstadt Germany, Jun. 2002.

[18] H. S. Chang, P. Fard, S. I. Marcus and M. Shayman,"Multitime scale Markov decision processes," IEEE Transactions on Automatic Control, Vol. 48, Jun. 2003, 976–987.

[19] H.S. Chang and S.I. Marcus, "Approximate Receding Horizon Approach for Markov Decision Processes: Average Reward Case," Journal of Mathematical Analysis and Applications, Vol. 286, October 2003, 636–651.

[20] H.S. Chang, M. Fu, J. Hu, and S.I. Marcus, "Simulation-based Algorithms for Markov Decision Processes," Springer, 2007

[21] B. Chen, G.N. Rouskas, and R. Dutta, "A Framework for Hieracrchical Traffic Grooming in WDM Networks of General Topology," BROADNETS 2005: Boston, MA, pp. 167–176.

[22] P.T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A Tutorial On the Cross-Entropy Method," Annals of Operation Research, Vol. 134, pp. 19–67.

[23] M. Dorigo and L. M. Gambardella 1997. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Trans. on Evolutionary Computation, Vol. 1, pp. 53–66.

[24] P. Erdős and A. Rènyi, "On Random Graphs I," Publicationes Mathematicae Debrecen, Vol. 6, 1959, pp. 290–297

[25] P. Fard, R. J. La, K. Lee, S. I. Marcus and M. A. Shayman, "Reconfiguration of MPLS/WDM Networks Using Simulation-based Markov Decision Processes," Conference on Information Sciences and Systems, Johns Hopkins University, 2005.

[26] A. Gencata and B. Mukherjee, "Virtual-Topology Adaptation for WDM Mesh Networks Under Dynamic Traffic," Proc. of IEEE INFOCOM, Jun. 2002.

[27] Harald Gropp, "Enumeration of regular graphs 100 years ago," Discrete Math., Vol. 101, 1992, pp. 73–85.

[28] F. Harary and E. Palmer, "Graphical Enumeration," Academic Press, New York, 1973.

[29] J. Hu, M. Fu, and S. Marcus, "A Model Reference Adaptive Search Method for Global Optimization," Operations Research, Vol. 55, 2007, pp. 549–568

[30] P. Ho and H.T. Mouftah, "Network Planning Algorithms for the Optical Internet based on the Generalized MPLS Architecture," Globecom, 2001.

[31] ILOG's CPLEX mathematical programming optimizer: http://www.ilog.com/products/cplex/.

[32] E. ISHIKAWA, S. XU, and Y. TANAKA "A Heuristic Wavelength Assignment Optimization for Optical Packet Switching Networks," IEEE TENCON, 2006.

[33] M. Jacobson, N. Shimkin, and A. Shwartz, "Piecewise stationary Markov Decision Processes, I: constant gain," Submitted to Mathematics of Operations Research.

[34] M. R. Jerrum and A.J. Sinclair, "Fast Uniform Genration of regular graphs," Theoretical Computer Science, Vol. 73, 1990, pp. 91–100.

[35] J. H. Kim and V. H. Vu, "Generating random regular graphs," Symposium on Theory of Computing, 2003, pp. 213–222.

[36] J. H. Kim and V. H. Vu, "Sandwiching Random Graphs," STOC, 2003, pp. 213–222.

[37] J. Labourdette and A. Acampora, "Logically rearrangeable multihop lightwave networks," IEEE Transactions on Communications, Vol. 39, Aug. 1991.

[38] J. Labourdette, G. Hart, and A. Acampora, "Branch-Exchange Sequences for Reconfiguration ofLightwave Networks," IEEE Transaction on Communications, Vol. 42 No. 10, pp. 2822–2832, Oct 1994.

[39] A. Leon-Garcia, "Probability and random Processes for Electrical Engineering," 2nd edition, Addison Wesley, 1994

[40] K.H. Liu, C. Liu, J.L. Pastor, A. Roy, and J.Y. Wei, "Performance and testbed study of topology reconfiguration in IP over optical networks ," IEEE Transactions on Communications, Vol. 50, Oct 2002, pp. 1662–1679.

[41] P. Marbach and J. Tsitsiklis,"Simulation-Based Optimization of Markov Reward Processes," IEEE TRANSACTIONS ON AUTOMATIC CONTROL, 46, NO. 2, Feb. 2001, pp. 191–209

[42] D. Mitra and K.G. Ramakrishnan, "Techniques for Traffic Engineering of Multiservice, Multipriority Networks," Bell Labs Tech. Journal, Vol. 6, No. 1, Jan. 2001, pp. 139–151.

[43] A. Narula-Tam and and E. Modiano, "Dynamic Load Balancing in WDM Packet Networks With and Without Wavelength Constraints," IEEE Journal on slelcted areas in Communications, Vol. 18, No. 10, Oct. 2000, pp. 1972–1979.

[44] C. Noon, "The generalized traveling salesman problem," PhD Dissertation, The University of Michigan, Ann Arbor: 1988.

[45] M. L. Puterman, "Markov Decision Processes: Discrete Stochastic Dynamic Programming," Wiley, New York, 1994.

[46] B. Ramamurthy and A. Ramakrishnan, "Virtual topology reconfiguration of wavelength routed optical WDM networks," in Proc. GLOBECOM, Nov. 2000, pp. 1269-1275.

[47] R. Ramaswami and K. Sivarajan, "Design of Logical Topologies for Wavelength-Routed Optical Networks," IEEE Journal on Selected Areas in Communications, Vol. 14, No. 5, Jun 1996, pp. 840–851.

[48] R.C. Read, "Some enumeration problems in graph theory," Doctoral Thesis, University of London, 1958.

[49] R.C. Read, "The enumeration of locally restricted graphs (I)," J. London Math. Soc., Vol. 34, 1959, pp. 417–436.

[50] R.C. Read, "The enumeration of locally restricted graphs (II)," J. London Math. Soc., Vol. 35, 1960, pp. 344–351.

[51] R.C. Read, "Some unusual enumeration problems," Ann. New York Acad. Sci., Vol. 175, 1970, pp. 314–326.

[52] R.C. Read and N.C. Wormald, "Counting the ten-point graphs by partition," J. Graph Theory, 1981, Vol. 5, pp. 183–196.

[53] R.C. Read and N.C. Wormald, "Number of labeled 4-regular graphs," J. Graph Theory, Vol. 4, 1980, pp. 203–212.

[54] F. Ricciato, S. Salsano, A. Belmonte, and M. Listanti, "Offline Configuration of a MPLS over WDM Network under Time-Varying Offered Traffic," INFOCOM 2001.

[55] S. Ross, "Stochastic Processes," 2nd edition, John Wiley & Sons Inc., 1996

[56] R. Y. Rubinstein 1997. "Optimization of Computer Simulation Models with Rare Events," European Journal of Operations Research, Vol. 99, pp. 89–112.

[57] R. Y. Rubinstein 1999. "The Cross-Entropy Method for Combinatorial and Continuous Optimization," Methodology and Computing in Applied Probability, Vol. 2, pp. 127–190.

[58] R. Y. Rubinstein 2001. "Combinatorial Optimization, Ants and Rare Events." In S. Uryasev and P. M. Pardalos, editors, Stochastic Optimization: Algorithms and Applications," pp. 304–358 Kluwer. R.

[59] R. Y. Rubinstein and A. Shapiro,"Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method," John Wiley & Sons, 1993.

[60] R. Y. Rubinstein and D. P. Kroese, "The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation, and Machine Learning," Springer, New York, 2004

[61] N. Sreenath, B. Gurucharan, G. Mohan, and C. S. R. Murthy, "A two-stage approach for virtual topology reconfiguration of WDM optical networks," Optical Networks Magazine, Vol. 2, No. 3, May/Jun. 2001, pp. 58–71.

[62] A. Steger and N.C. Wormald, "Generating random regular graphs quickly," Combinatorics, Probability and Computing, Vol. 8, 1999, pp. 377–396.

[63] H. Tang and E. Miller-Hooks, "Solving a generalized travelling salesperson problem with stochastic customers," Computers & Operations Research, Vol. 34, 2007, pp. 1963–1987.

[64] F. Tu and K.R. Pattipati, "Rollout Strategies for Sequential Fault Diagnosis," IEEE Trans. on Systems, man, and Sybernetics-Part A: Systems and Humans, Vol. 33, No. 1, Jan. 2003.

[65] X. Wang, H. Morikawa, and T.Aoyama, "Priority-based wavelength assignment algorithm for burst switched WDM optical networks," IEICE Transactions on Communications, Vol. E86-B, No.5, May 2003, pp.1508–1514.

[66] John Y. Wei, Chang-Dong Liu, and Sung-Yong Park, "Network Control and management for the next generation Internet," IEICE Transactions on Communications, Vol. E83-B, No. 10, pp. 2191–2209, Oct 2000.

[67] N.C. Wormald, "Some problems in the enumeration of labelled graphs," Doctoral Thesis, University of Newcastle, Callaghan, NSW, Australia, 1978.

[68] N.C. Wormald, "Enumeration of labelled graphs II : Cubic graphs with a given connectivity," J. London Math. Soc. (2), Vol. 20, 1979, 1–7.

[69] N.C. Wormald, "Enumeration of Labelled Graphs I: 3-connected graphs," J. London Math. Soc. (2), 1979, Vol. 19, pp. 7–12.

[70] N.C. Wormald, "Generating random regular graphs," Journal of Algorithms, 1984, Vol. 5, pp. 247–280.

[71] N.C. Wormald, "Models of Random Regular Graphs," Surveys in combinatorics (canterbury) London Math. Soc. Lecture Notes Ser., 265, 1999, Cambridge Univ. Press, Cambridge, pp. 239–298.

[72] G. Wu, E.K.P. Chong, and R. Givan, "Congestion Control Using Policy Rollout," Proceedings of the 42nd IEEE CDC

[73] H. Zang, J.P. Jue, and B. Mukherjee, "A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Optical WDM Network," Optical Networks Magazine, Vol. 1, No. 1, Jan. 2000, pp. 47–60.