Network Design and Availability Analysis for Large-Scale Mesh Networks

by

Wenjing Wang

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Engineering Management

Department of Mechanical Engineering
University of Alberta

ABSTRACT

Communication systems have cemented their position in many fields of our daily lives, such as governance, banking, correspondence, and traffic. Such systems often take the form of mesh networks in their topology. With ever-increasing data transmission rate in mesh networks and our growing reliance on continued network services, network availability has become increasingly important than ever before. Since survivable mesh networks are only designed to be 100% restorable under all single-failure scenarios, there is still chances for current networks to be failed, which usually leads to huge data rand revenue loss. In order to improve network availability while maintaining desirable investment cost of network design, the aim of this thesis is investigation on network design strategies and availability optimization algorithms based on the condition that the network has been designed to be fully restorable to all potential single failures.

The main contributions of this thesis comprise five parts. First, we provide thorough analysis of existing availability analysis methods for span-restorable networks. Based on this, we propose a more accurate method to evaluate network overall availability for span-restorable mesh networks. Moreover, comparisons between the existing and the new analysis methods are made. Second, we perform detailed investigation on traditional single-flow integer linear programming (ILP) and multi-flow ILP models of shared backup path protection mesh networks, and propose a new multi-flow ILP model. Experiments show that the new model solves 51% faster in terms of runtime than the traditional multi-flow ILP model. Meanwhile, we present an algorithm to analyze network overall availability for shared backup path protection networks. Results show that the new ILP

model works better in terms of overall availability in higher connected networks. Third, we present an algorithm to optimize availability for shared backup path protection networks. The core of this algorithm is an ILP model that is used to minimize the total lost flow caused by the second failure in a specified dual-failure scenario. The relationship between network overall availability and spare capacity is studied based on this optimization algorithm. Fourth, similar to the shared backup path protection networks, we also propose an algorithm to optimize network overall availability for path-restorable networks. At the meantime, the relationship between network overall availability and spare capacity is investigated. Fifth, we compare performance of the span-restorable, path-restorable, and shared backup path protection networks in terms of network overall availability. Results show that span-restorable networks have the highest overall network availability among the three above-mentioned types of networks, and that path-restorable networks have a slight advantage over shared backup path protection networks on average.

The theoretical analysis of this thesis provide insights in some degree in the understanding of mesh networks, and the algorithms proposed in this thesis is enlightening in the filed of network design and availability analysis. Implementation of the work in this thesis can help to design mesh networks faster with reasonable investment costs.

To my father, Zengfu Wang, my mother, Xixia Zong, my husband, Zhen Xu, my daughter, Arabella Xu, and my brother, Dongxiao Wang.

Acknowledgments

Seeing my name on the title page of this thesis, many memories from the past have been conjured, with lots of beautiful faces came into view. If it weren't for these remarkable people I have met, the work herein would not have been possible.

First of all, I would like to give my sincere thanks to *China Scholarship Council* (CSC) for their generous financial support. Without their help, I could not have had the opportunity to pursuit my Ph.D. degree in Canada.

The first person I would like to thank is my supervisor Dr. John Doucette. He is so knowledgeable, patient, and intelligent that he is always full of genius ideas. If it weren't for his guidance and encouragement, it would not have been possible for me to complete my Ph.D. program. I know even so many thanks still cannot repay what I owe him. I cherish our professor-student relationship as well as friendship forever.

The next person that I owe a lot is Dr. Yongsheng Ma. During the first two years of my PhD program, he had always believed in me and trusted my capability. Moreover, he assigned me important tasks with considerable payment. He is a nice, smart, resourceful, and broad-minded man full of energy, wisdom, and experience. I believe his guidance will be helpful in my future life and professional development.

I am also grateful to other doctors in my PhD candidacy exam and defense exam: Dr. Christopher Dennison, Dr. Donald Raboud, Dr. Hao Liang, Dr. Jacek Rak, Dr. Ming Zuo, Dr. Pedram Mousavi, and Dr. Zhigang Tian (alphabetical order). Thanks for their useful and valuable comments on my thesis work.

Next, I would like to give thanks to my family. More than anyone, it is my dad, Zengfu Wang, and my mum, Xixia Zong, to whom I owe the most and who I appreciate the most. It is their selfless love that gives me the courage, confidence, and power to face every challenge in my life. I will forever indebt to my husband, Zhen Xu, who has always stood by my side and gave me support whenever I felt exhausted and frustrated. My special thanks go to my daughter, Arabella Xu. Although she is just a baby girl right now, it is her that lights up my sky and gives my life more meaning and love. Meanwhile, I would like

to say thank you to my younger brother, Dongxiao Wang. Although one year younger than me, he always finds a way to encourage and support me like an older brother.

In the following, I would like to express my millions of thanks to my friends who have helped me in a variety of ways. First, I am thankful of Mustafa Babadagli for his help regarding Gurobi basics, because it is him that recommends me to switch from CPLEX to Gurobi (and the latter is much more powerful); I am thankful of Yali Wang, for her support in my topic switching, because without her patient introduction to my current research area, I would not have made up my mind to switch from my previous research group to my current one; and I am also very thankful of Guanghan Bai, for his help regarding reliability and availability basics. In addition, I would like to thank Cuiying Jian, Yanan Xie, Zhengrong Cheng, Jikai Liu, Lei Li, and Xihui Liang for helping me in proofreading of my publications, and Andres Castillo Lugo for his advice on my defense presentation. Meanwhile, I thank the tutors from the Center for Writers office for their help in grammar checkups of my thesis.

Last but not least, I would like to thank myself. Thank myself for choosing the right group even when some people thought it was late and crazy; thank myself for persisting on PhD program in the event of various difficulties; thank myself for working hard and never gave up; and thank myself for being optimistic all the time. I always believe that time will tell the truth and that time will let you know people around you and yourself. Always be an upright, optimistic, and obliging person. Only in this way, will you find peace, success, and happiness during your whole life!

Table of Contents

List of Tables

List of Figures

CHAPTER 1 INTRODUCTION

We now live in a progressively connected world where communication systems have cemented their position in almost every field of our daily lives, including agriculture, commerce, governance, healthcare, banking, industry, traffic, correspondence, etc. [1]-[3]. Although communication systems serve a wide variety of applications, at the core is a backbone network. *Wavelength division multiplexing* (WDM) is a technology that has proliferated across the globe in *backbone networks* [4]-[6]. At the physical level, such a network is composed of equipment (e.g., *add/drop multiplexers* (ADMs), *optical cross connects* (OXCs), amplifiers, switches, etc.) and transmission links (i.e., the physical fiber cables that are used to route the data traffic across different node equipment) [7]. At the logical level, node equipment and transmission links in the transport networks are abstracted respectively as nodes and spans within a network graph topology. Network design is typically established on this logical level. From an availability standpoint, network systems are susceptible to node and span failures. Nodes are more complicated than spans in terms of structures and functions; therefore, they get more physical protection and redundancy [8]. In addition, node failures are reducible to multiple span failures [9]. Since nodes are commonly assumed to be perfect, network designers are mainly concerned about span failures [9]. Determining the capacity on spans is a complex task and affects the service quality of millions of network users. Apart from that, outages may occur on spans. Potential causes for failures on spans can be categorized into three types: natural disasters (e.g., typhoon, flood, landslides, etc.), engineering activities (e.g., dig-ups resulting from repairs, maintenance, constructions, etc.), and willful deeds (e.g., deliberate sabotage, terrorism activities, etc.) [10]-[12].

Due to the large amount of traffic carried by a single optical fiber in WDM networks, network outages can be quite disruptive, leading to significant impacts, both economically and socially [13]-[17]. Guaranteeing some level of service availability is a vital task for designers. Generally, designers adopt redundancy in capacity of spans (i.e., through addition of *spare capacity*) to improve network resilience against span failures. *Mesh network survivability* is a class of mechanism used to guarantee some level of network survivability under specified failure scenarios. With such a mechanism, working routes usually follow the shortest path and backup routes are allowed to make use of spare capacity throughout the entire network [18]. Spare capacity is generally allocated across the network to accommodate one or more survivability mechanisms, such as *span restoration* [19], *path restoration* [20], *demand-wise shared protection* (DSP) [21], *shared backup path protection* (SBPP) [22], *p-cycles* [23], etc. The application of survivability mechanisms is by no means a complete solution to network failures. Near-perfect availability is extremely costly and impractical to achieve, and any particular level of network availability cannot be strictly guaranteed. Properly evaluating a network's robustness under failures, therefore, becomes indispensable to a network operator. In general, network robustness against failures can be measured by network's *restorability* (which focuses on the lost working capacity on the spans) or *availability* (which focuses on the lost working units on the working routes) [24],[26]. More formally, restorability is "the average fraction of failed working span capacity that can be restored by a specified mechanism within the spare capacity that is provided in a network" [26]. Availability is the average fraction of failed working units on working routes that can be restored with the survivability mechanisms applied [24]. Because it more generally considers complex

interactions amongst links on various routes, availability is a more realistic reflection of a network's robustness. The requirement of network service availability is specified in *service level agreements* (SLA) between network service providers and customers [27], [28]. Typically, survivable mesh networks have been designed to be 100% single-failure restorable, i.e., survivable networks can withstand any single span failures [25], [29]-[30]. Given that single-failure restoration is generally guaranteed in networks designed using conventional survivable network design approaches (and assuming restoration time is negligible [8]), single failure scenarios will not contribute to network outages in such situations. Herein, we focus on dual-failure scenarios, which dominate network outages (and unavailability) in single-failure survivable networks [31]. Other higher order failures, including node failures, are also considered to contribute a negligible amount to unavailability, owing to their extremely small probability compared with dual failures [8]. Correspondingly, we only consider dual-failure scenarios in this thesis. We will therefore use the term "dual-failure availability" or "dual-failure unavailability" when referring to availability that arises from only dual-failure scenarios.

In network problem context, the flow demand between a pair of nodes is referred to as a *commodity* [32]. The *multi-commodity network flow* (MCNF) problem is a network flow problem where multiple commodities are to be routed between different pairs of nodes [32]. When each pair of nodes is associated with a commodity, the problem is referred to as a *full-mesh* MCNF problem. All the problems in this thesis are full-mesh MCNF problems. The major task of network design for MCNF problems in mesh-survivable networks is to allocate capacity on each link so as to satisfy demand requirements on each commodity. Capacity design approaches can be further divided into *spare capacity*

allocation (SCA) [33]-[43] and *joint capacity allocation* (JCA) [44]. SCA assumes that the working capacity of each span has been assigned ahead of the design already, whereas JCA allocates both working capacity and spare capacity for each span. Since JCA can be simply transformed into SCA by assigning fixed values to each span's working capacity before design, we mainly perform JCA in the present thesis.

1.1 MOTIVATION AND GOALS

Although extensive work has been done already regarding network design and availability analysis, there is still room for enhancement. For instance, most previous work has focused on relatively small sized and less sparse networks. Furthermore, few work has studied the relationship between network availability and increased spare capacity on spans. In this thesis, we will focus on three major network survivability mechanisms (i.e., span restoration, path restoration, and SBPP). The primary contribution of this thesis is to provide a deep insight into network availability analysis and a way for obtaining the maximum network availability. The second contribution is to present an alternative and time-efficient method for allocating both working and spare capacity throughout the entire network. We also aim to compare the availability performance of networks with different survivability mechanisms. More specifically, our goals can be divided into the following five aspects:

Goal 1: Availability optimization for span restoration networks

We will propose a method to obtain optimal network availability for span-restorable networks, and then compare the performance of the new method with the existing methods.

Goal 2: Design and availability analysis for shared backup path protection networks

We will propose a new design model for networks protected by shared backup path protection mechanism. The design results are then served as the input for availability analysis. We will develop an algorithm to analyze network availability for such networks.

Goal 3: Availability optimization for shared backup path protection networks

Based on Goal2, we will attempt to propose a method to obtain optimal network availability instead of just simply analyzing availability that arises from the previous design using the new design approach. In the following, we will investigate the impact of spare capacity on network availability.

Goal 4: Availability optimization for path restoration networks

Similarly to the previous goal, we will try to propose an algorithm to obtain optimal network availability for path-restorable networks. Again, we will investigate the impact of spare capacity on network availability.

Goal 5: Comparison among various types of networks

After completing the four goals mentioned above, we now have better design models and optimal availability analysis algorithms for span restoration, path restoration, and shared backup path restoration, respectively. As such, our last goal will be set to compare network performances in terms of optimal availability for those various networks.

1.2 THESIS OUTLINE

The reminder of this thesis includes a thorough introduction to relevant background, a detailed discussion of methodology, capacity design and availability analysis of various survivability mechanisms (i.e., span restoration, path restoration, and SBPP), and

availability performance comparison of these mechanisms. Generally, the thesis structure is illustrated in Figure 1.1.

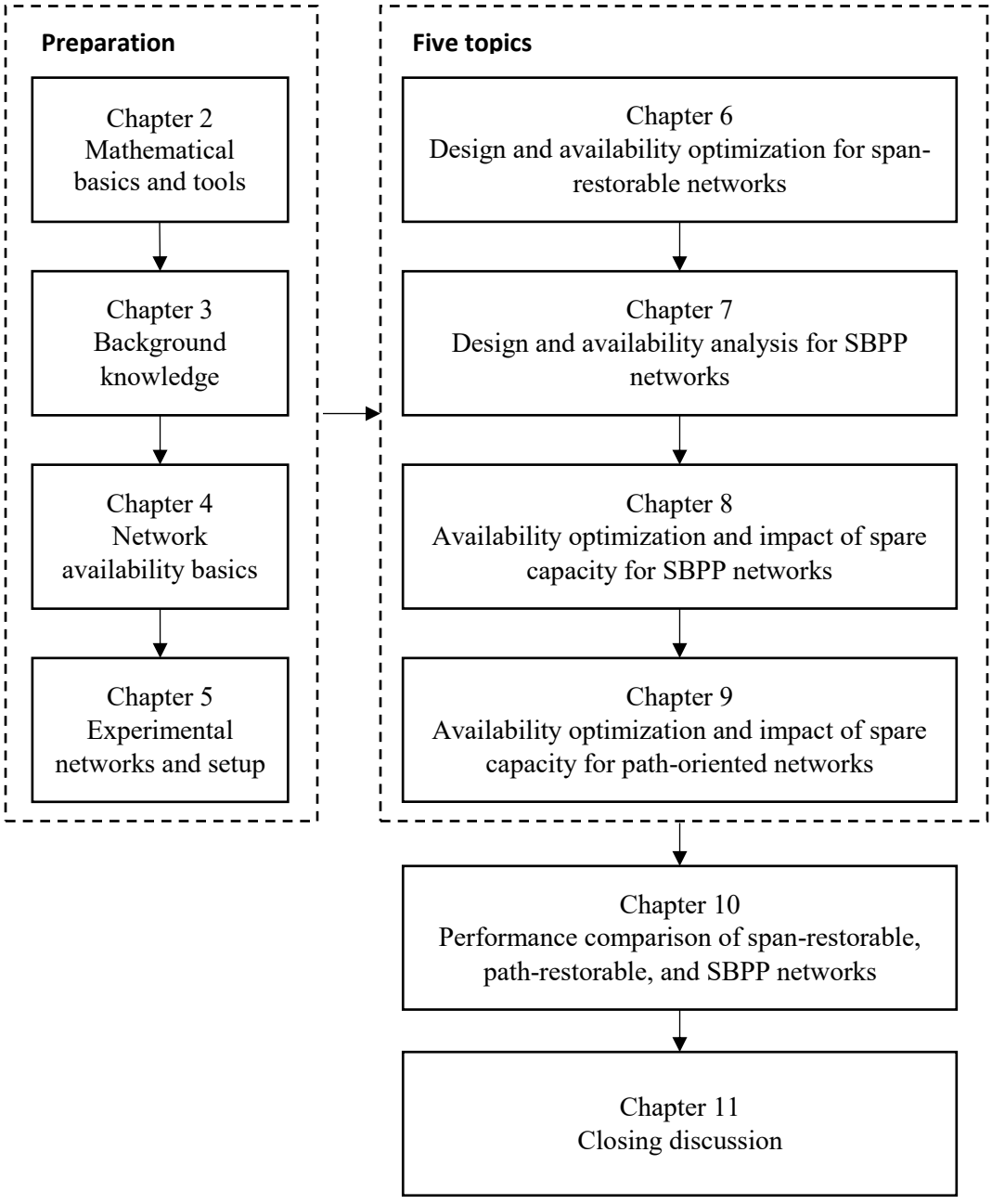


Figure 1.1 – Illustrations of thesis structure

As can be seen in Figure 1.1. In Chapter 2, we present the mathematical basics and tools used in the thesis work, including graph theory, searching algorithms, linear programming, and solving algorithms and tools. In Chapter 3, we introduce the concept of transport networks, the categories of mesh survivability mechanisms, and comparison between restoration and protection mechanisms. We also present the related research work in this chapter. In Chapter 4, we present the basics on network survivability and availability analysis. Additionally, we investigate the methods for analyzing the overall network availability for mesh networks that apply certain survivability mechanisms. In Chapter 5, we present the experimental networks, the assumptions, and the experimental setup used in the thesis work. In Chapter 6, we construct ILP design model for span-restorable mesh networks. Based on the design results, we look at the algorithm to optimize the overall network availability for such networks. In Chapter 7, we present a new multi-flow ILP design model for SBPP mesh networks and compare its performance with the traditional multi-flow ILP design model. Additionally, we investigate availability analysis method for SBPP mesh networks and propose an algorithm for evaluating the overall network availability for such networks. In Chapter 8, we seek to optimize network overall availability for SBPP mesh networks. The availability analysis algorithm proposed in Chapter 7 is used as the benchmark for validation of this new algorithm. We also look at the impact of increased spare capacity on network availability. In Chapter 9, we construct ILP design model for path-restorable mesh networks. Likewise, based on the design results, we propose an algorithm to optimize the overall network availability. In Chapter 10, we compare the performance of survivability mechanisms discussed in the previous chapters. In the last chapter, we summarize the PhD work and make conclusions for the thesis.

CHAPTER 2 MATHEMATICAL BASICS AND TOOLS

In order to suit the needs of readers with different levels of basic knowledge, we introduce the mathematic basics in this section, including graph theory, searching algorithms, linear programming, and solving algorithms and tools.

2.1 GRAPH THEORY

2.1.1 EDGES AND VERTICES

The two vertices that an edge connecting directly are known as its *end vertices* [84]. An edge is said to be *incident* on its end vertices, which are referred to as *adjacent vertices* [84]. If the end vertices of an edge have orders, then it is said to be *directed*, and *undirected* otherwise [84]. The vertex from which an edge starts is its *origin*, and the vertex at which an edge ends is its *destination* [84]. An edge can also be classified as *weighted* and *unweighted* [84]. If there is a value associated with an edge, then it is said to be weighted, and unweighted otherwise. The value associated with a weighted edge is referred to as its *weight* [84]. *Self-loop edges* (i.e., the two end vertices are the same) and *parallel edges* (i.e., a pair of edges with the same end vertices) are two special types of edges [84], [85].

Figure 2.1 shows examples of different edges. As shown in the figure, Figure 2.1 (a) is an undirected edge with end nodes of v_1 and v_2. Figure 2.1 (b) is a directed edge pointing from the origin v_3 to the destination v_4. Figure 2.1 (c) is a weighted edge with the length L as its weight. Figure 2.1 (d) describes a loop edge with the two end nodes both being v_7. Figure 2.1 (e) depicts a pair of parallel edges whose end nodes are both v_8 and v_9.

An edge can be labelled either by a single letter or by a pair of end vertices. For an undirected edge, the order of its end vertices is arbitrary, but for directed ones, the order is fixed. The number of edges incident on a vertex is referred to as its *degree* [84]. The average degree of all the vertices in a graph is termed as its *average degree* or *average nodal degree*.

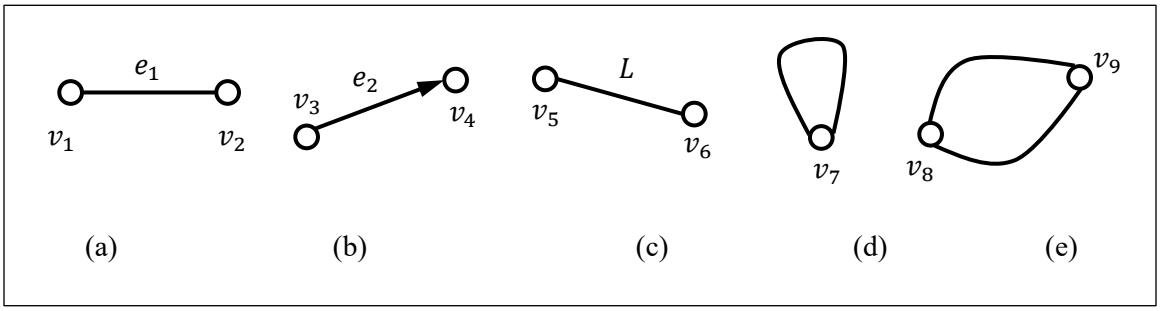


Figure 2.1 – Illustrations of edges

2.1.2 PATHS

A *path* is a sequence of connected distinct edges [84]. A path can be directed and undirected based on the property of its edges. The two end nodes of a directed path are named its origin and destination accordingly. The vertices of a path can be repetitive, whereas the edges should be different from each other. If all the edges and vertices are not repetitive except for its end vertices, the path is said to be a *simple path* [84]. Once the two end vertices of a path are the same, it is termed as a *cycle*. A *simple cycle* closes on itself only once [85]. A *Hamiltonian cycle* traverses all its nodes only once, whereas an *Eulerian cycle* crosses over all its edges only once [84]-[86]. The paths can be denoted either by a set of vertices or a set of edges.

Figure 2.2 shows examples of different paths. Figure 2.2 (a) is a general path with five vertices and six edges, among which there is one loop edge (i.e., e_2) and a pair of

parallel edges (i.e., e_4 and e_5). Figure 2.2 (b) is a directed path where its origin and destination are v_1 and v_5, respectively. Figure 2.2 (c) is a loop which closes itself twice. Figure 2.2 (d) is a simple loop. Figure 2.2 (e) is a Hamiltonian cycle (i.e., the blue arrowed lines), which can be denoted as $\{v_1, v_2, v_{5,} v_4, v_3, v_6, v_1\}$. The blue lines with arrows in Figure 2.2 (f) form an Eulerian cycle, which can be represented as $\{(v_{5,} v_2)$, (v_2, v_1), (v_1, v_5), $(v_{5,} v_3)$, (v_3, v_4), $(v_4, v_5)\}$.

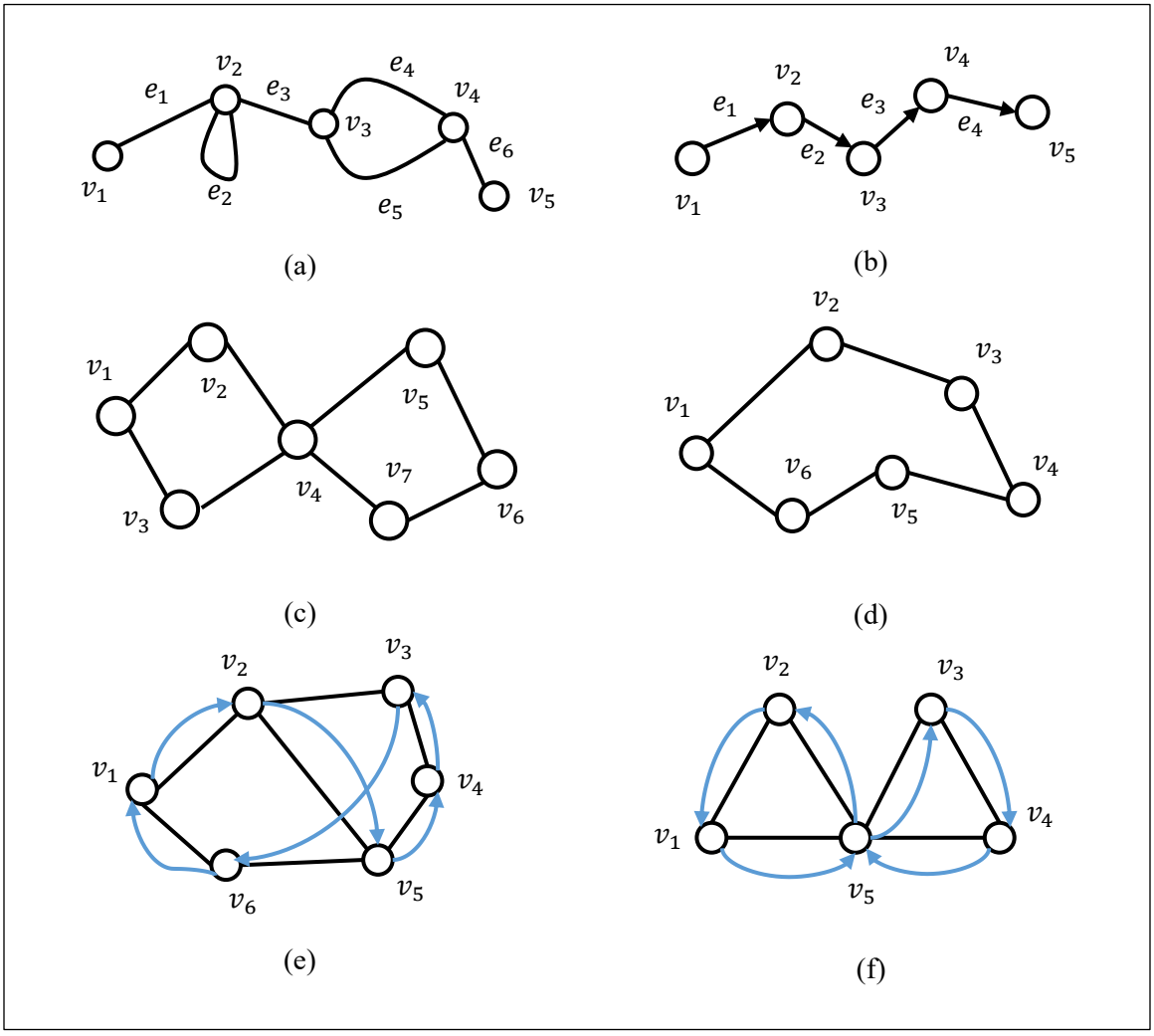


Figure 2.2 – Illustrations of paths

2.1.3 GRAPHS

A *graph* is a mathematical entity composed of vertices and edges connecting these vertices [84]-[85]. Mathematically, a graph is represented by a set of vertices (denoted by $V = \{v_1, v_2, \dots, v_m\}$ where m is the number of vertices and v_i $(i = 1,2, \dots, m)$ is the individual vertex), and a set of edges (denoted by $E = \{e_1, e_2, \dots, e_n\}$ where n is the number of edges and e_j $(j = 1,2, \dots, n)$ is the individual edge). The *order* of a graph is the number of vertices in the graph [84].

According to the properties of vertices and edges, graphs fall into a variety of categories as follows [84]-[86]. A *general graph* has at least one self-loop. A *simple graph* has no self-loops or parallel edges. A *multigraph* has at least one pair of parallel edges. If each distinct pair of vertices forms an edge, the graph is said to be *complete*. If there is a number associated with each edge, the graph is *weighted* and unweighted otherwise. If the weight represent capacity, the graph is also called a *capacitated graph*. Graphs can be split into *directed graphs* (with at least one directed edge) and *undirected graphs* (with no directed edges) as well. For a directed graph, if a vertex is the origin of all the edges incident on it, then it is called the *source* of that graph. Similarly, if a vertex is the destination of all the edges incident on it, then it is referred to as the *sink* of that graph.

In terms of connecting characteristics, graphs are broken down into the following categories. A *connected graph* is one with no unreachable vertices, i.e., there is always a path from any point to any other point in the graph. A graph is *disconnected* if it is not a connected graph. If a graph has at least two vertex-disjoint paths between every pair of vertices, it is called a *bi-connected* or *two-vertex connected* graph. If a graph processes at

least two edge-disjoint paths between every pair of vertices, it is dubbed a *two-connected* or *two-edge connected* graph.

Additionally, there are two special types of graph. A *tree* is a connected graph with no cycles; in comparison, a *forest* is a disconnected graph with no cycles [84].

In the field of transport networking, a network is typically represented and described as a simple graph. As mentioned in Chapter 1, the vertices and edges in the transport networks are called nodes and spans, respectively. Nodes are often expressed with $N = \{N_1, N_2, \dots, N_n\}$ where n is the number of node, and spans are typically denoted by $S = \{S_1, S_2, \dots, S_m\}$ where m is the number of nodes.

2.2 SEARCHING ALGORITHMS

2.2.1 DEPTH FIRST SEARCH

Depth first search (DFS) is an algorithm that searches for a specified node or traverses all the nodes in a graph starting with a *root node* and exploring as deep as possible before backtracking [87]. Since DFS is capable of exploring all the paths in the graph, we can obtain the *k*th shortest paths by sorting all the paths. Nevertheless, DFS is not well scalable for large-scale graphs. The general procedure for DFS to traverse all the nodes is described in the following three steps.

Step 1: Start from the root (pick one if it is not specified), and determine the traverse direction (either from left to right or from right to left).

Step 2: Explore from current node as deep as possible until you reach the bottom before backtracking to the upper layer.

Step 3: Repeat Step 2 until all the nodes are traversed.

An example of the DFS algorithm is illustrated in Figure 2.3 where the graph is searched from left to right. Figure 2.3 (a) is the original graph and Figure 2.3 (b) marks the detailed searching procedure explicitly.

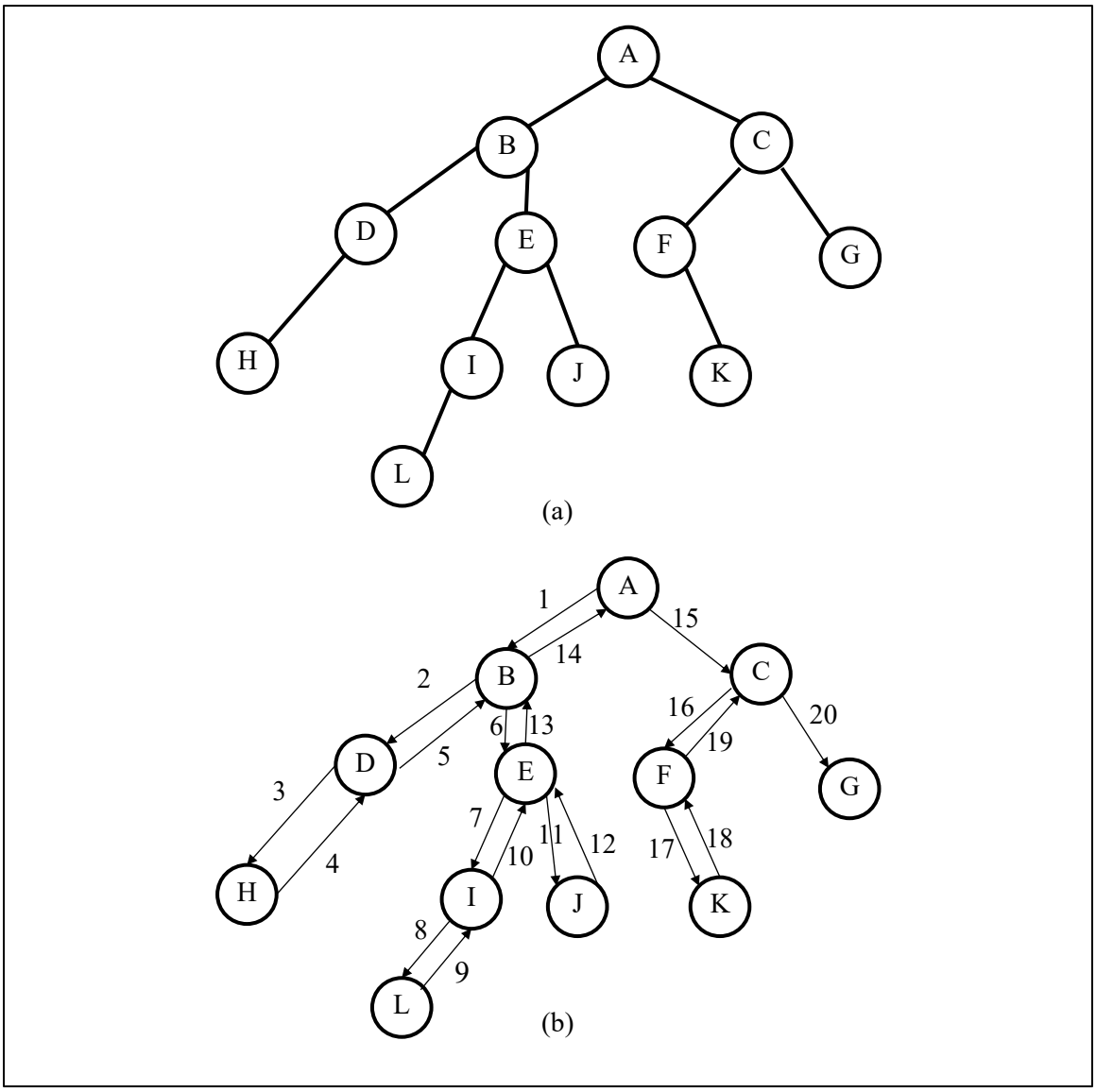


Figure 2.3 – An example of the DFS algorithm

2.2.2 DIJKSTRA'S ALGORITHM

Dijkstra's algorithm, also known as *the shortest path algorithm*, is an algorithm searching a given graph for the shortest path between node pairs [87]-[88]. The general procedure of Dijkstra's algorithm is described as follows.

Step 1: Starting at the source, *scan* all its neighbors (i.e., the adjacent nodes), and assign a *temporary label* to them with the format of {T, total distance to the source, predecessor/previous nodes}. Here, the letter "T" means temporary.

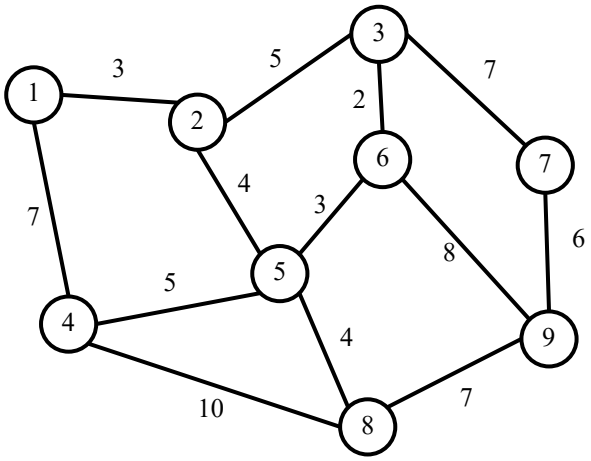
Step 2: Pick the node with the smallest total distance from the temporarily labeled nodes, and switch its label from temporary to permanent with the format of {P, total distance to the source, predecessor/previous nodes}. Likewise, the letter "P" represents permanent. When selecting the temporary labeled node with the smallest total distance, if there is a tie, it means more than one shortest path exists. As long as we can find one shortest path, it suffices, so we can pick one randomly. If a permanently labeled node has no neighbors, simply skip it and go to the next permanently labeled node.

Step 3: Scan the neighbors of the newly permanently labeled node, skipping the nodes that have permanent labels (the permanently labeled nodes have already been selected to appear on the path, so we skip them to avoid repetitiveness), assign a temporary label to the nodes that are not labeled. Then update the temporary label of the nodes that are labeled already if the new total distance is smaller.

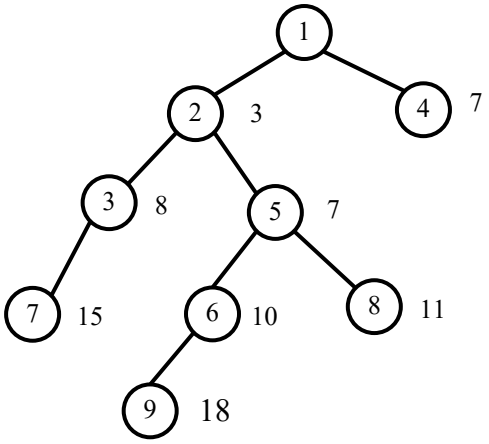
Step 4: Repeat Step 2 and Step 3 until the sink is permanently labeled. By tracing back the predecessors among the permanently labeled nodes starting from the sink, we can obtain the shortest path from the source to the sink in a reverse order.

After the sink is permanently labeled, if we continue to scan until all the nodes in the graph are permanently labeled, we will end up with a shortest path tree with each node associated with a permanent label. The root of the tree is the specified source, and there is only one path from the source to each of the other nodes. This unique path is the shortest path and the total distance associated with it is the corresponding shortest distance.

Figure 2.4 displays an example of Dijkstra's algorithm. Figure 2.4 (a) depicts a graph with the length of each span on the edge, Figure 2.4 (b) explains each scan process in the above described procedures, and Figure 2.4 (c) portrays the complete shortest path tree. The value close to each node in the shortest path tree is the total distance between the source and that node. For example, the shortest path from Node 1 to Node 5 is $\{1 \rightarrow 2 \rightarrow 5\}$ with the shortest distance of 7.



(a)



(c)

Scans	Nodes	Temporary label	Permanent label
1	Node 1	Node 2 {T, 3, Node 1}	Node 2 {P, 3, Node 1}
		Node 4 {T, 7, Node 1}	Node 4 {P, 7, Node 1}
2	Node 2	Node 3 {T, 8, Node 2}	Node 3 {P, 8, Node 2}
		Node 5 {T, 7, Node 2}	Node 5 {P, 7, Node 2}
3	Node 4	Node 8 {T, 17, Node 4}	
4	Node 5	Node 6 {T, 10, Node 5}	Node 6 {P, 10, Node 5}
		Node 8 {T, 11, Node 5}	Node 8 {P, 11, Node 5}
5	Node 3	Node 7 {T, 15, Node 3}	Node 7 {P, 15, Node 3}
6	Node 6	Node 9 {T, 18, Node 6}	Node 9 {P, 18, Node 6}
7	Node 8		
8	Node 7		
9	Node 9	Meeting stop criteria	

(b)

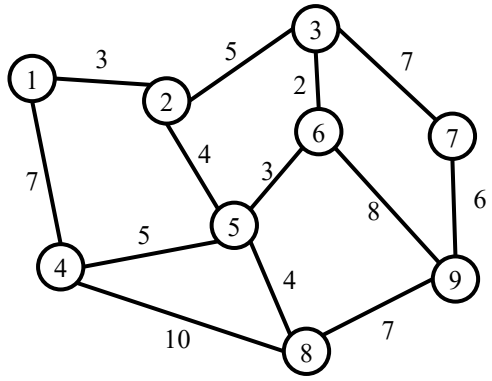
Figure 2.4 – An example of Dijkstra's algorithm

2.2.3 K SHORTEST PATH

K shortest path (KSP), an extension of the shortest path algorithm, is an algorithm to seek the first k*th* disjoint shortest paths between node pairs in a graph [89], [90]. This algorithm has important application because on some occasions, more than one path is required between a pair of nodes. The KSP algorithm is implemented though iterative

operations of the shortest path algorithm. More specifically, it finds the first shortest path with the shortest path algorithm, then removes the first shortest path from the graph before it continues to find the next shortest path. This process is repeated until all the *k*th shortest paths are found.

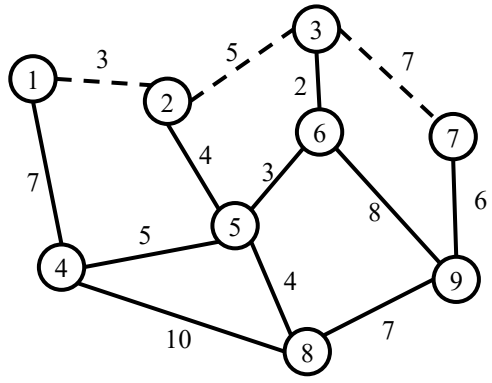
Figure 2.5 gives an example of the KSP algorithm. The objective is to find the first two shortest paths from Node 1 to Node 7. Figure 2.5 (b) finds the shortest path 1→2→3→7 with Dijkstra's algorithm, and it is then removed from the graph, as shown in Figure 2.5 (c). Figure 2.5 (d) shows the process for searching for the second shortest path 1→4→5→ 6 →9→7.



(a)

Scans	Nodes	Temporary label	Permanent label
1	Node 1	Node 2 {T, 3, Node 1}	Node 2 {P, 3, Node 1}
		Node 4 {T, 7, Node 1}	Node 4 {P, 7, Node 1}
2	Node 2	Node 3 {T, 8, Node 2}	Node 3 {P, 8, Node 2}
		Node 5 {T, 7, Node 2}	Node 5 {P, 7, Node 2}
3	Node 4	Node 8 {T, 17, Node 4}	
4	Node 5	Node 6 {T, 10, Node 5}	Node 6 {P, 10, Node 5}
		Node 8 {T, 11, Node 5}	Node 8 {P, 11, Node 5}
5	Node 3	Node 7 {T, 15, Node 3}	Node 7 {P, 15, Node 3}
6	Node 6	Node 9 {T, 18, Node 6}	
7	Node 8		
8	Node 7	Meeting stop criteria	

(b)



(c)

Scans	Nodes	Temporary label	Permanent label
1	Node 1	Node 4 {T, 7, Node 1}	Node 4 {P, 7, Node 1}
2	Node 4	Node 5 {T, 12, Node 4}	Node 5 {P, 12, Node 4}
		Node 8 {T, 17, Node 4}	
3	Node 5	Node 2 {T, 16, Node 5}	Node 2 {P, 16, Node 5}
		Node 6 {T, 15, Node 5}	Node 6 {P, 15, Node 5}
		Node 8 {T, 16, Node 5}	Node 8 {P, 16, Node 5}
4	Node 6	Node 3 {T, 17, Node 6}	Node 3 {P, 17, Node 6}
		Node 9 {T, 23, Node 6}	Node 9 {P, 23, Node 6}
5	Node 2		
6	Node 3		
7	Node 8	Node 9 {T, 23, Node 8}	
8	Node 9	Node 7 {T, 29, Node 9}	Node 7 {P, 29, Node 9}
9	Node 7	Meeting stop criteria	

(d)

Figure 2.5 – An example of the KSP algorithm

2.3 LINEAR PROGRAMMING

2.3.1 MATHEMATICAL TERMINOLOGY

The *operations research* (OR), also known as *management science*, is a scientific decision making approach that aims to seek for the best way to design and operate the system [91]. The term *optimization* represents the process in search of the best solutions for OR problems. This process begins with simplifying the real problems by a *mathematical model*, and then designs a *mathematical programming model* based on it. A mathematical programming model is a mathematical decision model that optimizes (i.e., maximize or minimize) the *objective function* while satisfying a series of *constraints* by choosing the values of *decision variables* [91], [92]. The terms regarding a mathematical programming model are defined as follows.

The *decision variables* of a mathematical programming model are the quantity whose values are varied (i.e., can be controlled). For simplicity, all the decision variables in a model are denoted collectively by the column vector of $\boldsymbol{x} = [x_1\ x_2\ \dots x_m]^{\mathrm{T}}$. The objective function of a mathematical programming model, denoted by $f(\boldsymbol{x})$, is a function represented by the decision variables. The constraints of a mathematical programming model are the conditions that are required to be followed in search of solutions. They are usually expressed as equalities or inequalities, such as $f_1(\boldsymbol{x}) = b_1$ and $f_2(\boldsymbol{x}) \leq b_2$.

If the objective function and the constraints are all linear combinations of the decision variables, the mathematical programming is called *linear programming* (LP) [92], [93]. Any combination of the values for all the decision variables in the decision variable vector is dubbed a *solution*. If a solution satisfies all the constraints, it is referred to as a *feasible*

solution. If a feasible solution optimizes the value of the objective function, it is named an *optimal solution*. We use "an optimal solution" here because it is possible that more than one optimal solution exists for an LP problem. All the constraints form a *constraint set*, and all the feasible solutions form a *set of feasible solutions*. Thus, the objective of an LP can be interrelated as finding out the best solution from the set of feasible solutions. For two-dimensional LPs, the set of feasible solutions is also known as the *feasible region* on the two-dimensional coordinate systems.

A given LP is in the standard format if it complies with the following rules [92]:

(1) The objective function is a minimization.

(2) All the variables are non-negative.

(3) All the constraints except the non-negativity constraints are equalities.

(4) All the values of the right hand side of the constraints are non-negative.

The algebraic form of the standard LP is expressed as follows:

$$\text{Minimize} \quad z = \sum_{j=1}^{n} c_j x_j \tag{2-1}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \qquad \forall i = 1,2,\dots,m \tag{2-2}$$

$$x_j \geq 0 \qquad \forall j = 1,2,\dots,n \tag{2-3}$$

where c_j $(j = 1,2,\dots,n)$ is the coefficient of the decision variable x_j in the objective function, which is also referred to as the *cost coefficient*, and a_{ij} $(i = 1,2,\dots,m, j = 1,2,\dots,n)$ is the coefficient of the decision variable x_j in the ith constraint.

The *simplex method* [92], developed by George Dantzig, is an algorithm for solving non-integer LP problems. This algorithm involves a series of matrix operations on the standard form of the LP and arrives at a better solution after each operation.

2.3.2 INTEGER LINEAR PROGRAMMING

Integer linear programming (ILP) is an LP in which at least one of its decision variables is restricted to be an integer [96]. ILP has a standard form similar to that of LP, except that all decision variables are restricted to be integers.

Due to the variables being integers, the continuous variables-based Simplex algorithm is not applicable for ILP problems. However, Simplex algorithm still can enlighten us in solving ILP problems, because if we allow the integer constraints to be violated, then we can use Simplex algorithm to solve the problem. We can later add integer constraints back to obtain integer solutions. Generally, the ILP problems are relaxed first, i.e., the integer variables are allowed to take a real value. The relaxed version of the original ILP problem is known as its LP relaxation. It is obvious that in solving an LP relaxation, we can obtain either of the two results: (1) all the variables happen to be integers, and (2) some of the variables that are integers in the original ILP problems take on non-integer values. If it is the first case, we are lucky because the solution of the LP relaxation is also the solution of the original ILP problems. That being said, we cannot guarantee to have integer solutions for an LP relaxation. The *branch and bound algorithm* [97]-[99] was originally designed for solving discrete optimization problems. It enumerates over the candidate solutions based on a systematic rule. The candidate solutions take the form of a rooted tree. The root node denotes the full set of the candidate solutions, and the branches

of the tree (the nodes other than the root node) represent the subsets of the candidate solution set. To improve solving efficiency, the branch and bound algorithm enumerates the set of candidate solutions by applying terminating conditions instead of enumerating all the candidate solutions in order to searching for the optimal solution.

2.4 PROGRAMMING AND SOLVING TOOLS

In the thesis work, we use two languages to model the proposed problems. One is *a mathematical programming language* (AMPL) [100], developed by Robert Fourer, David M. Gay, and Brian W. Kernighan in around 1985. The other one is *Python*, developed by Guido van Rossum and first released in 1991 [102]-[103].

Gurobi, [101], developed by Zonghau Gu, Ed Rothberg, and Bob Bixby in 2008, is a mathematical programming solver for solving a variety of optimization problems, including LP, *mixed integer linear programming* (MILP), *quadratic programming* (QP), *mixed integer quadratic programming* (MIQP), *quadratic constrained programming* (QCP), and *mixed integer quadratic constrained programming* (MIQCP). More importantly, Gurobi optimizer offers interactive interface with Python. This allows us to insert the optimization process into Python by calling Gurobi repetitively wherever applicable in some of the availability optimization topics.

CHAPTER 3 BACKGROUND

In order to prepare the readers in a different field with adequate background knowledge, we provide an overview of transport networks and associated survivability mechanisms in this section.

3.1 NETWORK CLASSIFICATIONS

A telecommunication network is a collection of terminals that are connected by links where terminals can communicate through those links [45]. Fundamentally, a telecommunication network falls into two categories: *private networks* and *public networks* [50]. The private networks are owned and operated by private corporations for internal use [50]. These networks can be partitioned into three tiers according to their geographical boundaries: *local area networks* (LAN), *metropolitan area networks* (MAN), and *wide area networks* (WAN) [50]. LAN usually covers a small area such as a building and a residence; MAN covers a wider area, which is usually a few blocks of a city; and WAN usually spans over hundreds to thousands of kilometers in area [8]. The corporations owning the private networks do not necessarily own the land that their networks cross over, especially in the case of WAN. In fact, many companies of private networks lease transmission capacity from the owners of the public networks [50]. The owners of the public networks are also known as *carriers* or *service providers*, and the owners of the private networks are referred to as *private users* or *clients* [8].

As explained in the above paragraph, public networks offer services to private networks. Beside that, public networks are capable of providing much higher transmission

capacity than private networks [8]. The architecture of a public network is illustrated in Figure 3.1 [50]. As shown in the figure, a public network consists of the *metropolitan access network* (or *access network*), the *metropolitan inter-office network* (or *inter-office network*), and the *inter-exchange network* (or *long-haul network*) [50]. Among them, the access network and the inter-office network are referred to as *metropolitan* (or *metro*) *networks* collectively. The access network usually spans over a few kilometers and it connects the clients to nearby *central offices* (COs). The inter-office network typically ranges from up to tens of kilometers, and it connects multiple COs within the same city or region. The long-haul network's reach is usually hundreds to thousands of kilometers, and it connects a group of different cities or regions. The traffic going out of the metropolitan area is finally routed into the long-haul network through the big hubs of the metropolitan inter-office network. Apart from the covering area differences, access networks tend to be very sparse in terms of topology, and is not necessarily survivable [18]. Metro networks connect regional COs together and their cost is dominated by nodal equipment costs [18].

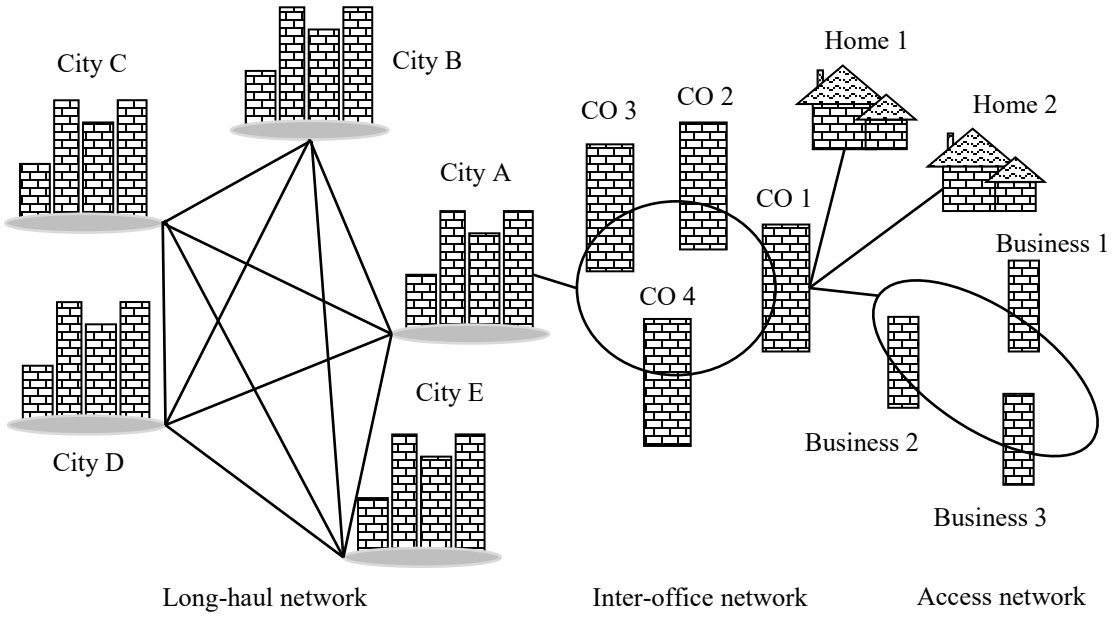


Figure 3.1 – The graphical structure of public networks [50]

3.2 TRANSPORT NETWORKS

As explained in Section 3.1, the public networks offer services to the private networks, i.e., they are in a service provider and client relation. Based on such a relation, public networks are referred to as *transport networks* and private networks are referred to as *client networks* [8]. Formally, a transport network, which is also known as a *transportation network* or a *backbone network*, is the core part of a telecommunication network that provides bulk carriage for various communication services such as vehicular movement and commodity flow [18], [50]. The transport network is a point-to-point transmission system with multiple channels multiplexing different services together and route them from origin to destination. With the development of technology, the realization of transport networks has undergone great changes. The earliest transport network was designed to transmit voice data, which was named *public switched telephone network* (PSTN).The modern network, which is composed of fibre optic cables that are connected

to nodal switching devices, is able to route a variety of communication services including voice data, video data, and other data [18]. The backbone networks, which carry a vast amount of data, are typically composed of high-speed and high-capacity links. Due to their complexity, it usually takes several years to design and build a backbone network. The structure of a typical backbone network is illustrated in Figure 3.2 [46]. This is a US backbone network from AT&T with the nodes representing various cities and the spans representing fibers connecting them. The backbone network takes the form of mesh-like structure and connects many major cities.

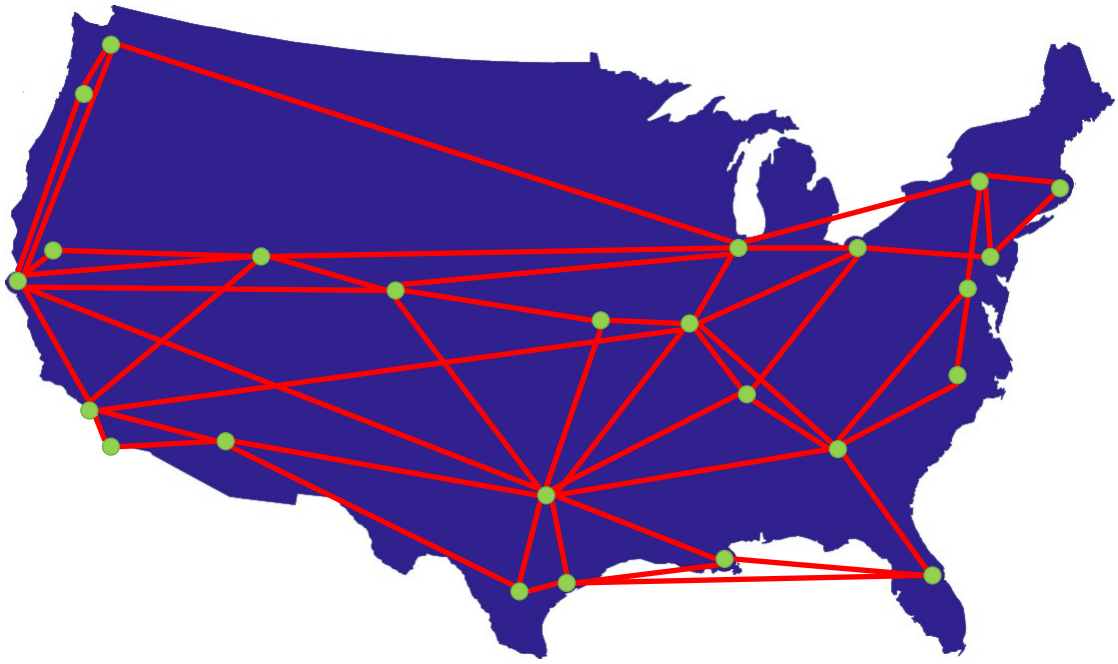


Figure 3.2 – Topology of a real backbone network [50]

3.3 MESH NETWORK SURVIVABILITY

As stated in Chapter 1, we adopt network survivability mechanisms to ensure a certain level of survivability under specified failure scenarios. Mesh network survivability mechanisms are further divided into two categories: *localized mechanism* and *end-to-end*

mechanism [18]. Localized mechanism restores affected demands with a set of backup routes between the end nodes of the failure itself. End-to-end mechanism establishes the replacement routes between origin and destination of the demands. The replacement routes of end-to-end mechanism can be completely disjoint with the original working routes. They can also reuse the surviving portion of the original working routes. End-to-end mechanism is more capacity efficient compared to localized mechanism. Mesh survivability mechanism can be classified in more detail as mentioned in Chapter 1. In this section, we only focus on those employed in this thesis.

3.3.1 SPAN RESTORATION

Span restoration is the most common form of localized mechanism where a set of local replacement routes between the end nodes of the failed span is used to restore the demands [47]-[48]. A mesh network designed with span restoration is referred to as a *span-restorable* network. Since span restoration acts in the vicinity of the failure, we only need to know the status of this failed span and we do not need to be concerned about the status of other spans on the failed working route. This makes it simple to implement in reality [24]. Additionally, the set of local preplacement routes can be defined ahead of failures, which makes if fast to take effect upon failures [24]. However, since it works locally, more spare capacity is usually required for establishing replacement routes.

Figure 3.3 shows an example of span restoration. As shown in the figure, a least one backup route is established for each working channel on the failed span between the end nodes of that failed span. More backup routes can be established as long as there is enough

spare capacity to accommodate the routes. We do not to have the knowledge of or consider for origin and destination of involved demands.

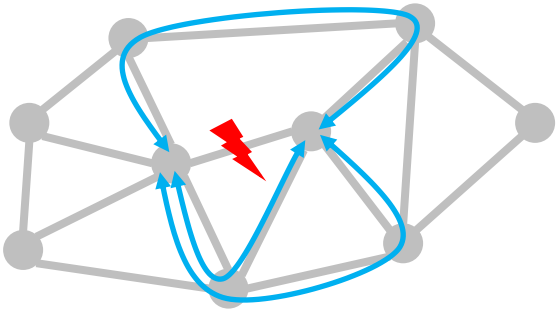


Figure 3.3 – An example of span restoration

The end nodes associated with the involved failure act to initiate the restoration response between them, so they are referred to as the *custodial nodes* with respect to the involved span failure [24]. Unlike rings, the replacement routes in span restoration do not need to be via a single route, rather, they can follow any available routes [18]. This enables us to adopt any potential replacement routes more flexibly. A factor that affects capacity efficiency of span restoration is *backhaul* (or *loopback*), which occurs when a backup route traverses any span on the corresponding working route, as the red line shown in Figure 3.4. The solid blue line represents a working route between node pair A-B. The dotted line denotes a backup route of failed span CD. The backup route traverses span BD and produces a backhaul BD on the replacement route of node pair AB. This backhaul creates redundant spare capacity on span BD and thus reduces spare capacity efficiency of span restoration.

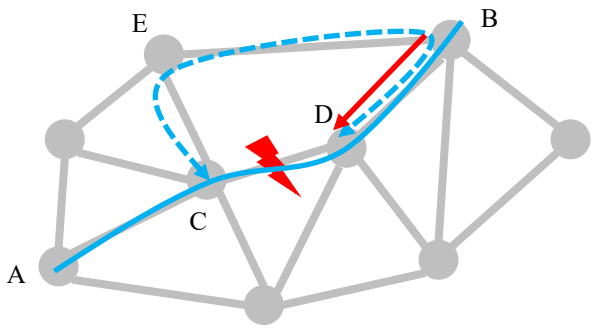


Figure 3.4 – An example of backhaul

3.3.2 SHARED BACKUP PATH PROTECTION

Shared backup path protection (SBPP), also known as *failure independent path protection* (FIPP) and *shared path protection* (SPP), is a promising member of end-to-end mesh survivability mechanism [24], [51]-[54]. Demands routed through disjoint working routes will not claim the backup capacity simultaneously. Thus, SBPP allows sharing of spare capacity on backup routes by various disjoint working routes. SBPP is failure independent in that the same backup route is adopted regardless of where the actual failure occurs on the working route [18]. This simplifies the activation process of backup routes, but abandons the reuse opportunity of the surviving portion on the failed working routes. Besides, when a failure occurs, we need to know the state of each node including the spare capacity sharing relationships throughout the entire network. If we make the backup routes node-disjoint with their corresponding working routes, SBPP is capable of protecting the network against node failures.

Figure 3.5 illustrates examples of SBPP. As shown Figure 3.5 (a), two pairs of nodes (node pair A-B and node pair C-D) are routed through disjoint working routes (solid lines) separately. Their backup routes (dotted lines) are disjoint with their corresponding working

routes. Since the working routes are disjoint for these two node pairs, they will not be hit by the same span failure simultaneously. As such, the two backup routes can share spare capacity on their common span EF. In Figure 3.5 (b), node pair G-H and node pair H-I are routed through their respective working routes (solid lines) separately, but they have one common span HK. Their corresponding backup routes are shown with dotted lines. That being the case, the two backup routes cannot share spare capacity on their common span HJ. In other words, the spare capacity on span HJ should be enough to accommodate the flow on both backup routes.

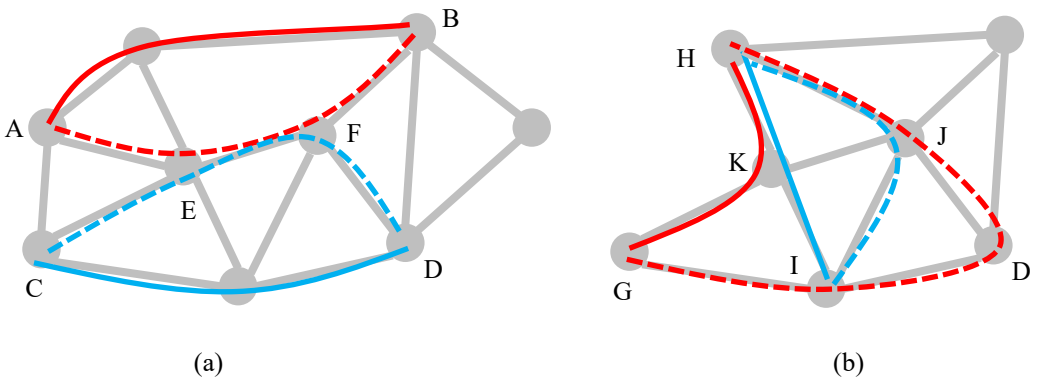


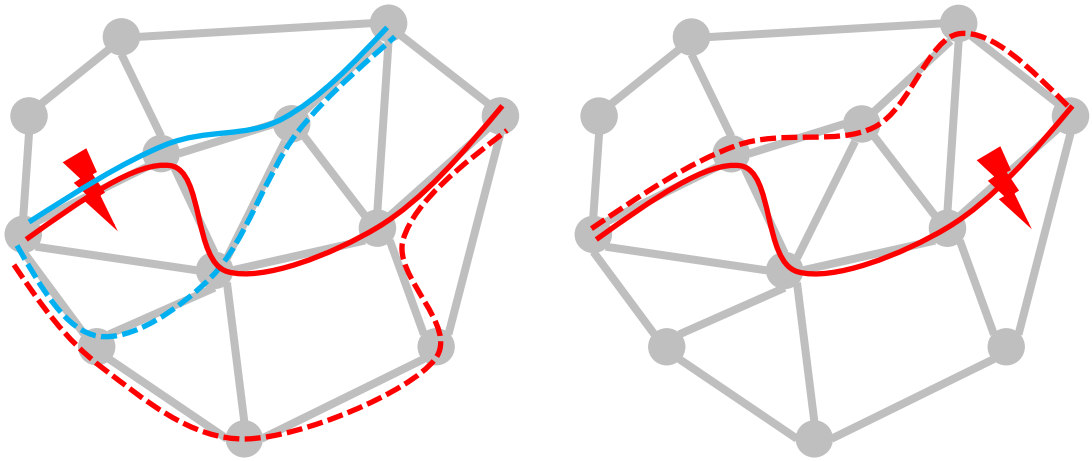
Figure 3.5 – An example of shared backup path protection

The term "protection" in SBPP is derived from *automatic protection switching* (APS) protection mechanisms [55]. At first glance, SBPP is very similar to 1+1 APS where the signal is dual-fed onto two mutually disjoint routes, and the best signal is selected at the end node [56], [57]. This is because in both cases, the failed working flow between origin and destination of a demand can be rerouted via a predefined backup route that is distinct with the original working route. However, with SBPP, the capacity for establishing the backup routes can be shared by failure-disjoint working routes [24]. This reduces spare capacity redundancy greatly and makes SBPP more cost-effective.

3.3.3 PATH RESTORATION

Similar to SBPP, *path restoration* belongs to the family of end-to-end mechanism and it replaces the failed working route with a set of backup routes from the end-nodes of the failed span [14]-[15], [32]. The end-to-end backup routes in path restoration depend on where the failure occurs on the corresponding working route, so path restoration is also called *failure-dependent path protection* (FDPP) [18]. Typically, path restoration allows *stub-release* where the surviving portion of the failed working route is allowed to be reused for establishing backup routes of the corresponding demands or for any other demands, as needed [24]. Compared to span restoration, the replacement backup routes are distributed throughout a much wider range. In addition, path restoration avoids *backhauls* in its backup routes, so it is more capacity efficient than span restoration [50].

Two situations of path restoration are illustrated in Figure 3.6. Two working routes (solid lines) and their corresponding backup routes (dotted lines) under the specified failure are shown in Figure 3.6 (a). Since the specified failure affects the two working routes simultaneously, the two backup routes have a common span but they cannot share spare capacity on this common span. Note that both backup routes have common span with their corresponding working routes. With stub-release allowed, working capacity on the common span can be reused as backup capacity on corresponding backup routes. In Figure 3.6 (b), a different failure occurs, and the red working route employs a different backup route as shown in the figure. This time, the backup route can share spare capacity with the blue dotted backup route on their common span.



(a) (b)

Figure 3.6 – An example of path restoration with stub-release

As illustrated above, the backup routes for the same demand in path restoration are not necessarily to be the same under different failure scenarios, which makes it more flexible than SBPP. With stub-release, path restoration is more capacity efficient than SBPP and span restoration [18]. When the backup routes are designed to be node-disjoint with the corresponding working route, node failures can be protected with path restoration.

3.4 Related Work

Extensive research has been completed in the areas of network design. Since the value of capacity (both working capacity and backup capacity) is integer, the involved LP design models are all ILP, which has been prove to be NP-hard [24], [58]. As a result, the associated design problem is extremely difficult to solve for large-scale networks. To tackle this issue, a variety of approaches have been proposed and investigated towards network design in the form of deterministic (exact) or heuristic approaches [59]. Heuristic approaches (e.g., *genetic algorithms*, *simulated annealing*, *tabu search*, etc.) are generally faster but are not guaranteed to arrive at a strictly optimal solution (or even a good solution

if used improperly), while deterministic approaches (e.g., *linear programming*, *mixed integer programming*, etc.) are likely to reach a satisfactory optimality gap at the cost of runtime [59]. Considering their respective advantages, both approaches are employed widely in network design.

Heuristic approaches lend themselves well to finding disjoint routes for SBPP networks and have a widespread application [60]-[71]. Earlier work by Józsa, and Orincsay combined on-line and off-line optimizations to achieve a faster routing scheme [62] via three algorithms: *cut down maximums* (CDM), *adaptive method* (AM), and *iterative method* (IM). Shen and Grover studied dynamic provisioning methods for SBPP survivable networks [63]. They considered two provisioning approaches: *sharing with partial routing information* (SPI) and *sharing with complete routing information* (SCI). Qu et al. conducted similar work through developing a heuristic algorithm called CAFES to provision capacity for dynamic SBBP networks [65]. They focused on the challenges of finding two disjoint routes in trap topology situations. Nguyen et al. used a set of disjoint candidate routes to assign one working route and one backup route for each given demand to achieve full single-failure restorability [66]. This model treated working and backup routes collectively as candidate route pairs. Józsa et al. proposed a heuristic algorithm to design a network so it can sustain two simultaneous failures [67]. They combined the *Dijkstra algorithm* and the *Edmonds-Karp algorithm* to find three disjoint paths for each demand, and used them as the working route, the primary backup path, and the secondary backup path. To improve efficiency, they attempted to find paths for each connection one by one. Haahr et al. made a comparison between various heuristic algorithms (i.e., a naive two-step algorithm, simulated annealing, and an adaptive large neighborhood search) and

a lower bound algorithm [68]. Walkowiak and Klinkowski described six heuristic algorithms to solve SBPP problems and employed a traditional ILP single-flow model as the benchmark for evaluating the heuristic algorithms [69]. Wang et al. argued that lightpath physical distance should be considered when assigning *frequency slot* (FS) to the working route and backup route pair [70]. More recent research on dynamic routing of SBPP-based optical networks can be found in [70]-[71]. Lau and Jha developed a heuristic algorithm, which was referred to as *service path local optimization* (SPLO) for the online path restoration problems [72]. Ruepp et al. compared the *Ford and Fulkerson algorithm* and the *Dijkstra algorithm* in path restoration networks [73]. They concluded that the Dijkstra algorithm performs better in terms of capacity usage.

Deterministic approaches mainly take the form of ILP models in solving network designs, and are also well documented in the literature, although not as common as heuristic approaches. The *arc-flow* (also referred to as *node-arc* or *transshipment*) and *arc-path* approaches are two different paradigms for formulating an ILP model [24], with their major difference being whether routes are explicitly enumerated (in the arc-path approach they are, but not in the arc-flow approach). Most ILP models are based on an arc-path approach. Numerous examples of network design ILP models can be found in the literature [24], [74]-[76]. A common objective is the total cost of capacity on each span in the network, and the model is subjected to a number of constraints based on various assumptions. Work in [77] pointed out that for SBPP networks if the numbers of working and backup routes per demand are both restricted to one, an extensive number of binary variables are produced, which eventually increases computing complexity of the model. Allowing multiple working routes and backup routes for each demand has the potential to

enhance solving efficiency and reduce runtime by eliminating the associate binary variables (and replacing them with simple integer variables). This brings us a new perspective to look at the application of SBPP mechanisms. To avoid confusion, the original SBPP model is referred to as *traditional single-flow model*, and the model proposed in [77] is referred to as *traditional multi-flow model*. Kodialam et al. studied the path restoration routing problems and proposed an LP-based algorithm using two-phase routing scheme [78]. They provided resiliency against link failures for the end-to-end backup paths.

Network availability has been investigated by a number of works as well with the help of physical experiments or mathematical models. This thesis only focuses on the methods through mathematical models. With mathematical methods, researchers either aim to obtain the value of network availability or seek to guarantee a specified level of availability. As for the latter, some researchers embed availability requirements as one of the constraints, and other researchers employ network availability as the objective function. Clouqueur and Grover [26] analyzed service availability in span-restorable mesh networks with both theoretical framework and computational routing trials. They first determined the network average dual-failure restorability by the ratio of total non-restored working capacity to total affected working capacity over all the dual-failure scenarios, then interpreted the results in terms of end-to-end service path availability based on a series system availability. They concluded that the level of dual-failure restorability is relatively high for a mesh network designed to be fully restorable for all single-failure scenarios. Doucette et al. [55] provide a method to investigate the relationships between service availability and total capacity in the SBPP survivability scheme. They gave the definition

of dual-failure restorability for SBPP and built an ILP model to limit the number of working routes that allow sharing the same backup link explicitly. Zhou and Held [79] developed an ILP model to improve average path availability for span-restorable networks under dual-failure scenarios. This model minimizes the total non-restored working capacity over all dual-failure combinations. The major contribution of this model is that the restoration sequence is considered explicitly. Li et al. [80] designed a *p*-cycle network with an ILP model to minimize dual-failure restorability. They found that involving dual-failure restorability explicitly in the constraints would result in lower cost in richly connected networks. Herker et al. [81] embedded availability constraints into their model to guarantee expected service availability. Azim and Kabir [54] propose a mathematical model to investigate service availability for WDM networks under multiple link failures. This model only requires part of the shared backup paths, leading to a lower computational complexity. Alashaikh et al. [82] provided spine as a new concept to enhance network availability. The idea is to divide the links in the network into two types, those with low physical availability, and those with high physical availability. The working routes are all routed on the high-availability links. The spine of a network in nature is a spanning tree of the network. They also developed a heuristic to select a proper spanning tree to form the spine of that network. Conway combined the path rerouting algorithm and the *dynamic path failure importance sampling* (DPFS) scheme, and then developed a scheme to evaluate service availability in mesh networks with dynamic path restoration [83]

CHAPTER 4 NETWORK AVAILABILITY BASICS

The analysis of network availability is based on the basics regarding restorability and availability. In this section, we provide an overview of basics regarding restorability and availability. We combine the characteristics of mesh networks to investigate these basics, in terms of different types of survivability mechanisms.

4.1 UNAVAILABILITY OF SPANS

In reality, we are usually provided with unavailability of a product rather than availability [24]. From an availability standpoint, the life of a network span effectively alternates between *uptime* (i.e., the working state) and *downtime* (i.e., the failed state) [108]. When one span fails, repair actions are performed to return it to operation until a subsequent failure occurs, and the cycle repeats. The probability that a working span will be failed in a unit of time is referred to as *failure rate* (denoted by λ); and the probability that a failed span will be fixed in a unit of time is *repair rate* (denoted by μ) [108]. The unavailability of a span k, accordingly, can be calculated as in Eq. (4-1) [24].

$$U_{\text{span}}^{k} = \frac{\lambda}{\mu+\lambda} \tag{4-1}$$

For optical networks, a typical timeframe for repairing a failed span is in the order of 12 hours [77], and so we use a repair rate, μ, equal to the reciprocal of 12 hours. The failure rate of a span is proportional to its length (denoted by l_{span}^{k}) [24]. We define the failure rate per unit length as a unit-length failure rate (denoted by λ_{unit}). The values of span's failure rate are listed in Table 4.1. Note that some of the values are directly obtained from

the corresponding literature, and some are calculated via the information provided by the corresponding literature.

Table 4.1 – The values of span's failure rate from literature

Sources	Unit failure rates (# of failures per hour per 1 km)
X. Wang, G. Shen, et al. [109]	2.0×10^{-7}
B. Todd, J. Doucette [77]	3.4×10^{-7}
A. J. Vernon, J. D. Portier [110]	3.4×10^{-7}
S. Verbrugge, D. Colle, et al. [111]	3.8×10^{-7}
W. Ni, J. Wu, et al. [112]	2.0×10^{-7}~8.0×10^{-7}

From the table, we can see that the order of magnitude for span's failure rate is 10^{-7} per hour in transport networks. It does not matter which specific value we pick as long as we choose the same order of magnitude[1]. In Appendix I, we tested the unit failure rate ranging from 2.0×10^{-7} to 8.0×10^{-7} as shown in Table 4.1. From the results, we can tell that the trends of the figures for those experimental results are the same. In other words, the failure rate selection within this failure rate range will not affect the experimental results. In this thesis, we adopt the unit failure rate as documented in [77]. As such, the value of unavailability of a span k can be obtained as in

$$\lambda = l_{\text{span}}^{k} \cdot \lambda_{\text{unit}} \tag{4-2}$$

4.2 SPAN-ORIENTED MESH NETWORKS

Span-oriented mesh networks are the mesh networks that protect or restore a failed working route with backup routes that share the same end nodes of the failed span. In this thesis, we only focus on one type of span-oriented mesh networks, i.e., span restoration

[1] This will be demonstrated in Appendix I.

networks.

4.2.1 DUAL-FAILURE RESTORABILITY

Although the network has been designed for full single-failure restorability by default, dual-failure scenarios can still strike the network and cause service breakdowns. In other words, some working capacity on these failed working routes cannot be restored during dual-failure scenarios. In span-oriented mesh networks, the amount of the *non-restored working capacity* on both failed spans under each dual-failure scenario (i, j) is denoted by $NWC_2(i, j)$. The sum of $NWC_2(i, j)$ over all the dual-failure scenarios is then defined as *dual-failure non-restored working capacity* (NWC_2) [24], in Eq. (4-3), where S is the set of spans in the network.

$$NWC_2 = \sum_{i,j \in S | i \neq j} NWC_2(i, j) \tag{4-3}$$

Due to the existence of non-restored working capacity, the overall restorability under dual-failure scenarios is less than unity. In span-oriented networks, dual-failure restorability for a specific failure scenario (i, j) refers to *specific dual failure restorability*, denoted by $R_2(i, j)$, which is the ratio of the amount of the restored working capacity to that of the total originally affected working capacity. It can be calculated by Eq. (4-4) [113].

$$R_2(i, j) = 1 - \frac{NWC_2(i,j)}{w_i + w_j}, \quad i, j \in S | i \neq j \tag{4-4}$$

Here, w_i is the amount of working capacity on span i, and w_j is the amount of working capacity on span j.

In span-oriented networks, the weighted average of $R_2(i,j)$ over all the dual-failure combinations is referred to as *dual-failure restorability*, denoted by R_2 and calculated through Eq. (4-5) [24].

$$R_2 = 1 - \frac{NWC_2}{2(|S|-1)\sum_{i\in S} w_i} \tag{4-5}$$

In that calculation, $|S|$ is the number of spans in the network.

4.2.2 DUAL-FAILURE AVAILABILITY

Network availability is commonly calculated by considering each service path individually, as this will provide a picture of the availability (or unavailability) experienced by individual customers [8]. If calculated on the network as an average, the service availability for a specific customer might be very low even if the availability for the entire network is relatively high [8]. We therefore consider it to be more meaningful to investigate the availability of each service path than that of the overall network. The most common way for calculating availability of a specific service path is from series system reliability theory [114]. More specifically, for an un-survivable network, a service path is available only if all of its components (spans) are available. For a span-oriented survivable network, the failure rate of each span is reduced, owing to the effect of survivability schemes, which decreases the unavailability of the span. This decreased unavailability for a specific span i is dubbed equivalent channel unavailability of span i, denoted by U_i^* and estimated with Eq. (4-6) in span-oriented networks [24].

$$U_i^* = U_i^{\text{phy}} \sum_{j\in S|j\neq i} U_j^{\text{phy}} \big(2 - R_2(i,j) - R_2(j,i)\big),\ i \in S \tag{4-6}$$

Here, U_i^{phy} and U_j^{phy} are the unavailability of span i and span j, respectively.

Moreover, because the unavailability of each span is extremely small, the unavailability of a specific service path p is approximately Eq. (4-7) [115].

$$U_p = \sum_{i \in S_p} U_i^* , \ \ p \in P \tag{4-7}$$

Here, P is the set of working routes in the network and S_p is the set of spans on service path p.

Dual-failure service paths unavailability (SPU_2) for span-oriented networks is the average of service path unavailability over all dual-failure scenarios [24], calculated by

$$SPU_2 = \frac{\sum_{p \in P} U_p}{|P|} \tag{4-8}$$

In this calculation, $|P|$ is the number of service paths in the network.

4.3 PATH-ORIENTED MESH NETWORKS

Path-oriented mesh networks protect or restore a failed working route with backup routes that share the end nodes with corresponding demand. We focus on two types of path-oriented mesh networks in this thesis, i.e., path restoration mesh networks and SBPP mesh networks.

4.3.1 DUAL-FAILURE RESTORABILITY

Since the backup routes in span-oriented networks are designed on a per demand basis, whereas the backup routes in path-oriented networks are designed on a per span basis, the definition of $R_2(i,j)$ and R_2 for span-oriented networks is not suitable any more for path-oriented networks. To address this issue, we redefine *specific dual-failure*

restorability $R_2(i,j)$ for path-oriented networks under dual-failure scenario (i,j), as follows in

$$R_2(i,j) = 1 - \frac{wf_{\text{lost}}(i,j)}{wf_{\text{aff}}(i,j)} \tag{4-9}$$

In this equation, wf_{lost} is the total working flow that is not restorable after the failures of span i and span j, in that order. The order matters here. Working routes that are affected by the first failure can be fully restored, but routes affected by the second failure will generally incur some amount of outage, owing to the fact that available backup resources may no longer be sufficient. The variable $wf_{\text{aff}}(i,j)$ represents the total working traffic that is affected by either or both of the two span failures. This definition is route-oriented, rather than span-oriented, and reflects the real lost service in the network.

For SBPP networks, this is theoretically applicable in both single-flow and multi-flow models, but the practical implementation is slightly different in the two. In the single-flow model, a working route is protected by a single pre-defined backup route, so when a second failure occurs, we can know for certain whether this working route is restored or failed simply by checking the status of its backup route. That is, if the backup route crosses the second failed span, then this working route cannot be restored; otherwise it is restored. In the multi-flow model, multiple backup routes protect a working route, so the protected working route is more likely to be partially restored.

4.3.2 DUAL-FAILURE AVAILABILITY

For path-oriented mesh networks, based on dual-failure restorability, we define the *dual-failure availability of a working route* p (denoted by $A_2(p)$) as the availability of the

working route that arises when only dual-failure scenarios are considered, which can be calculated as per the equation

$$A_2(p) = 1 - \sum_{i,j \in S | (i \cup j) \in p} U^i_{\text{span}} \cdot U^j_{\text{span}} \cdot [1 - R_2(i,j)] \quad (4\text{-}10)$$

To calculate $A_2(p)$, we do not need to consider all the dual-failure scenarios, but only need to consider all the dual failures affecting working route p, as suggested in the lower bound index of the summation, i.e., $i,j \in S | (i \cup j) \in p$. This is because only those dual failures that affect the working route p will contribute to the dual-failure availability of that specific route.

The network dual-failure availability (denoted by A_2) is subsequently defined as the average of the dual-failure availability for all the working routes in the network with dual-failure scenarios being the only contributor to failures, as shown in

$$A_2 = \frac{\sum_{p \in P} A_2(p)}{|P|} \quad (4\text{-}11)$$

CHAPTER 5 EXPERIMENTAL NETWORKS AND SETUP

5.1 CONCEPTS OF LARGE NETWORKS

There is no specific definition regarding large networks. In this thesis, the keyword "large" in the title means two aspects. On one hand, it represents network scale, which is represented by the number of nodes in the network. From the literature, many researchers focus on networks with less than 40 nodes networks, so we refer to networks with more than 40 nodes as large networks. On the other hand, it represents network connectivity. The higher connectivity is, the more intensive a network is. In this thesis, we refer to networks with connectivity more than 3.0 as large networks.

5.2 CONCEPTS OF NETWORK FAMILY

There have been a number of experimental networks available in the literature [8], [31], [38], [104]-[107], but most of them are separate real networks and not in any systematic manner. The work in [11] propose the concept of network family to create a series of related networks in a systematic manner. In order to obtain a series of experimental networks to better suit our needs, we follow their rules to create new experimental networks.

The tools we applied are Inkscape software and SVG script. We use Inkscape software to create the draft for mesh networks. For example, in order to create a mesh network with specified number of nodes and spans, we randomly draw the specified number of dots and lines as required with Inkscape. After that, we obtain its SVG script

from Inkscape and fine tune the shape of the dots and lines we created to finish the network. Finally, we use Python to create *.top files and *.dem files by obtaining the information from the created networks.

Each network family consists of 11 test case networks sharing a common set of nodes. To create each family, we start with a network of average nodal degree 5.0 (which we refer to as a master network), remove several spans to create a new but related network with average nodal degree 4.8, remove several more spans to create another new but related network with average nodal degree 4.6, and so on until we have a network with average nodal degree 3.0. We apply a uniform random demand between 1 and 10 to each node pair in a network, and the demands applied to each network within a family are the same within that family. Figure 5.1 shows the members in the 10-node network family.

Note that these 15 network families are our pool of networks, and it is not necessarily for us to use them all for each topic. We will specify the employed networks where applicable explicitly.

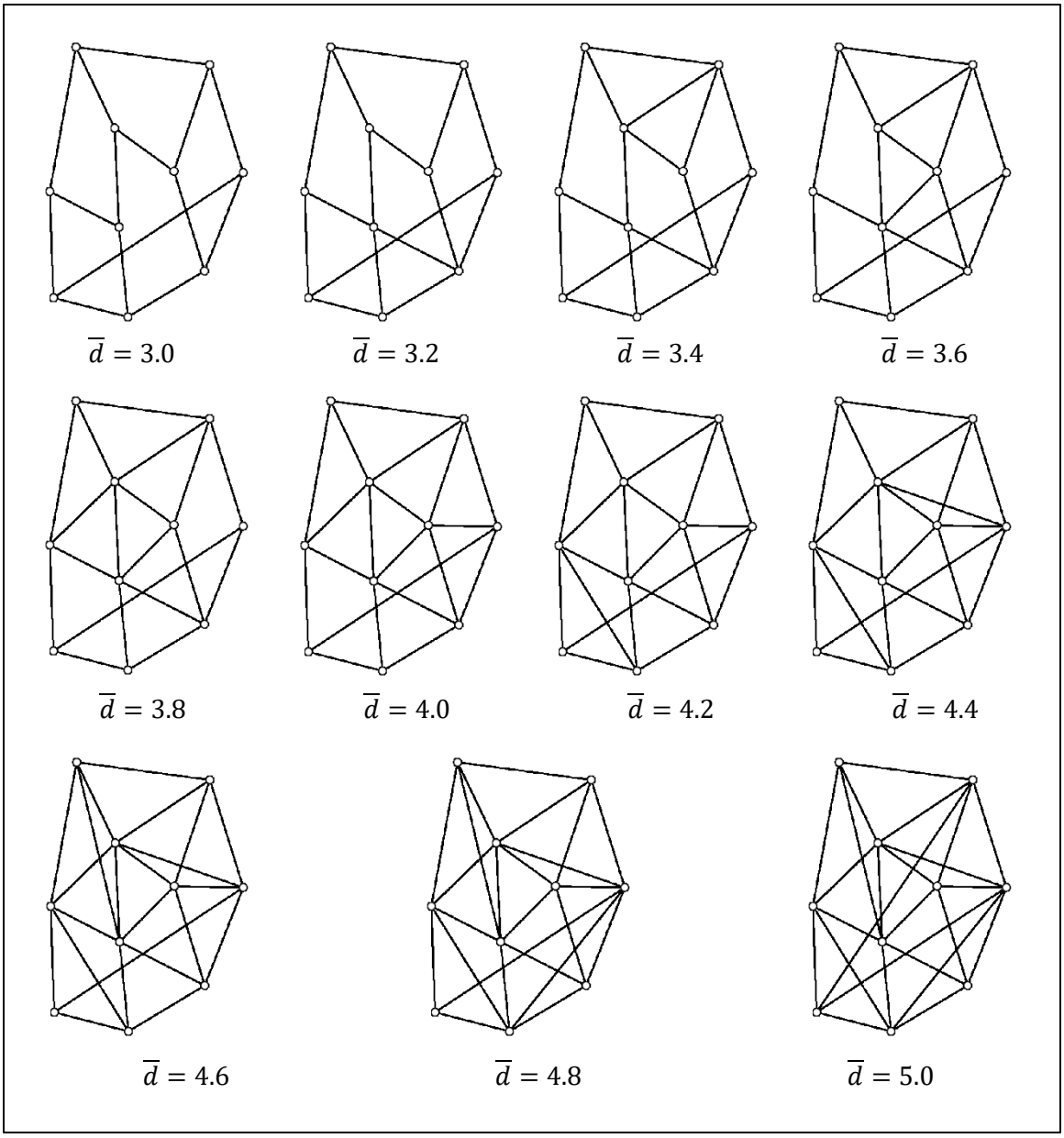


Figure 5.1 – Network members in 10-node network family

5.3 TOPOLOGIES OF MASTER NETWORKS

Finally, we have created a total of 165 test case networks, divided into 15 network families. The topologies of the 15 master networks are shown in Figure 5.2.

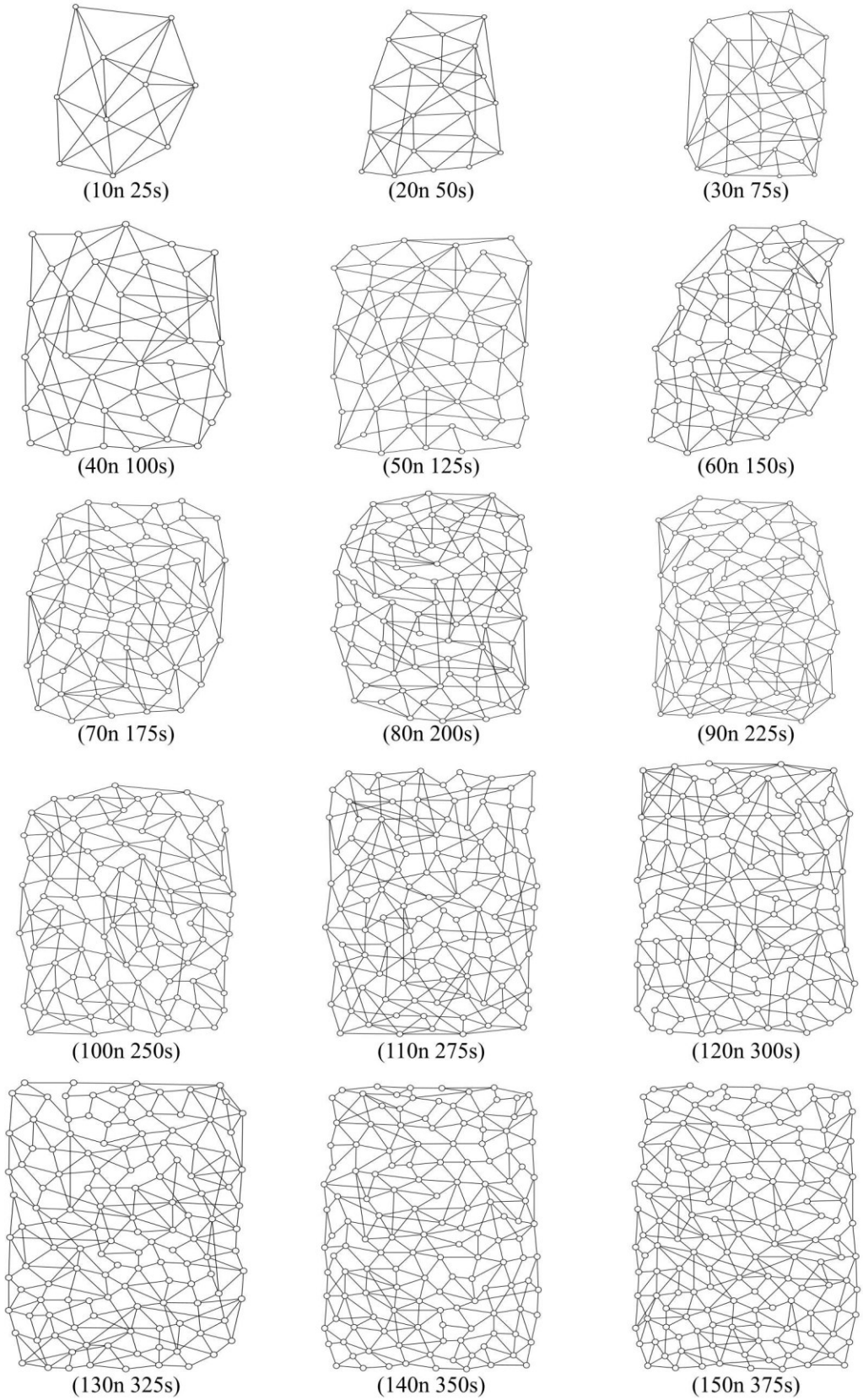


Figure 5.2 – Topologies of master networks

5.4 ASSUMPTIONS

The ILP models we formulate in this thesis makes the following assumptions:

(1) We assume a full mesh of demands between each O-D pair in the network.

(2) We assume a linear relationship between capacity and cost (i.e., no economy-of-scale considerations [2]).

(3) We assume that the amount of capacity on each span is equal to the number of wavelengths routed on it (i.e., no modularity considerations [2]).

(4) We assume a static design situation where the demands are stable and fixed over the design; there is no existing demand release or new demand arrival.

(5) The network topology is known and fixed in advance of the design process.

(6) The order of two failures in a specified dual-failure scenario is considered. We follow "first come first served" rule when dealing with two failures. That is, the restoration of the first failure has priority over the second failure.

(7) The network is designed to be fully single-failure restorable.

(8) Only working capacity can be restored upon failure; spare capacity cannot.

Other than the above common assumptions for all ILP models, if there are others required, we will discuss them where applicable.

5.5 EXPERIMENT SETUP

All ILP models are implemented in AMPL [100] solved with Gurobi 6.5.0 [101] and availability analysis is implemented in Python 2.7 [103] on a computer with 128 GB RAM and Intel(R) Xeon(R) E5-2650 v3 CPU running at 2.3 GHz. In some topics, we call Gurobi

within Python in each loop where an ILP model is involved. This will be specified explicitly where applicable.

CHAPTER 6 DESIGN AND AVAILABILITY OPTIMIZATION OF SPAN-RESTORABLE NETWORKS[2]

As shown in the literature, network availability can be calculated both directly and indirectly. For span-restorable networks, in the direct calculation, network service path unavailability is calculated explicitly using a number of means (unavailability is preferred to availability for calculation simplicity), while in the indirection calculation, availability metrics are calculated to evaluate network availability [31]. In this chapter, we investigate the conventional methods (i.e., both direct and indirect methods) and propose a new way to evaluate network availability for span-restorable networks.

6.1 MOTIVATIONS AND GOALS

There are two major drawbacks of the conventional methods and we elaborate each of them in this following.

(1) One drawback of the conventional methods is that the calculation stems from an understanding of the behavior of the survivability scheme under single-failure scenarios, where one failed span corresponds to one failed service path. In general, only one failed span is involved, and there is no explicit concern about the interaction mechanism among various dually failed spans.

Equations (4-3) through (4-7) show that current methods are all closely related to

[2] This chapter is adapted from our journal paper: W. Wang, J. Doucette, "Dual-Failure Availability Analysis of Span-Restorable Mesh Networks," *Journal of Network and Systems Management*, vol. 24, no. 3, pp. 534–556, July 2016.

$NWC(i,j)$. Nevertheless, for a given value of $NWC(i,j)$, different connection circumstances among channels may arise, as demonstrated in Figure 6.1. The solid lines in the figure indicate a direct link between adjacent nodes, while the dashed lines represent an indirect or multi-hop link (i.e., there are some other nodes between them). In either case, whether connected by a solid line or a dashed line, each represents a service path between the two nodes. Figure 6.1(a) shows a three-node and three-span network with explicit working channels, and Figure 6.1(b), 6.1(c), and 6.1(d) display three distribution circumstances of $NWC(1,2)$, the value of which is 2. Under the circumstances shown in Figure 6.1(b), the two non-restored working capacities are distributed on two different service paths of the two failed spans. Thus, the number of non-restored service paths is two. Under the circumstances in Figure 6.1(c), the two non-restored working capacities are distributed on the same service path, i.e., the service path crosses both failed spans. Hence, the number of non-restored service paths becomes one. Finally, in Figure 6.1(d), the two units of non-restored working capacity are both on one failed span, leading to a single failed service path. That is, with the same design of working capacity on each span and the same value of $NWC(i,j)$ under each dual-failure scenario, the actual number of non-restored service paths in the network can differ according to different distributions of $NWC(i,j)$ on each failed span. This is because the number of units of non-restored working capacity does not necessarily correspond to the same number of units of non-restored service paths for dual-failure scenarios. Instead, the actual number of non-restored service paths is determined by the distribution of each non-restored working capacity on the failed spans. Note that the failed spans in Figure 6.1(b) and 6.1(c) are shown to be adjacent here merely to simplify showing the shared service paths on the involved failed spans explicitly. In the

general case, this example can be extended to non-adjacent failed spans and similar observations would be made. Also note that Figure 6.1 depicts only working channels. More specifically, the working channels marked with an "X" are not restored, while the others are restored using spare channels not shown.

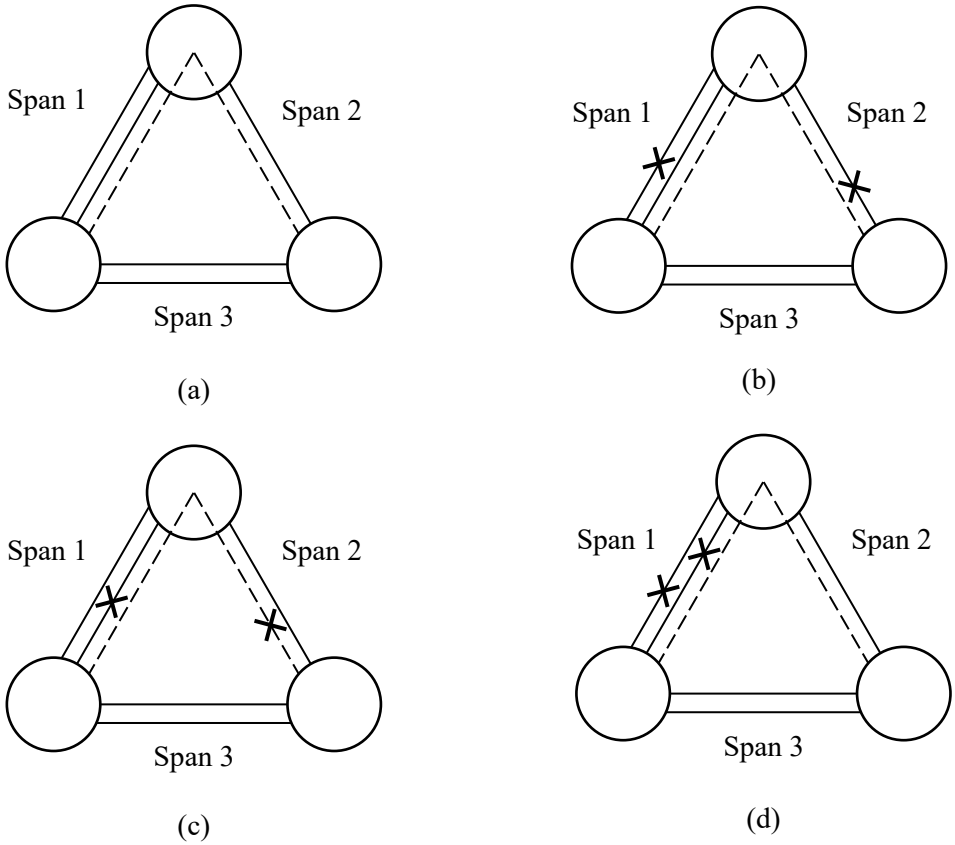


Figure 6.1 – Illustrating different distribution circumstances of non-restored working capacity

(2) Another drawback of the conventional methods is that the order of the two span failures in a specified dual-failure scenario is ignored. For example, the variable $NWC(i,j)$ is usually calculated as the summation of $NWC(i) + NWC(j)$, in which case, the value of $NWC(i,j)$ and $NWC(j,i)$ are assumed to be the same. As a matter of fact, the order of the span failures is of great consequence in terms of dual-failure availability. This is because

the restoration of the first failure is our priority in a specified dual-failure scenario, as discussed in Section 3.6.

Current methods, therefore, cannot reflect the accurate service path unavailability. As such, we will present a new method in order to obtain more accurate network availability.

6.2 SPECIFIC NUMBER OF LOST PATHS

The number of service paths that traverse both span i and span j is defined as *the number of shared service paths* and denoted by $SP(i,j)$. The best-case and worst-case distributions are illustrated in terms of $SP(i,j)$ in Figure 6.2. The dashed line represents the shared service path by span i and span j. In the best case (Figure 6.2 a), two of the four non-restored units of working capacity are on one shared service path, and the other two are on the other shared service path. That is, the i and j segments of the two shared service paths are all occupied by the non-restored working capacity, resulting in the smallest number of non-restored service paths. Conversely, in the worst case, neither of two non-restored units of working capacity are on the i and j segments of the same service path, leading to the maximum number of non-restored service paths.

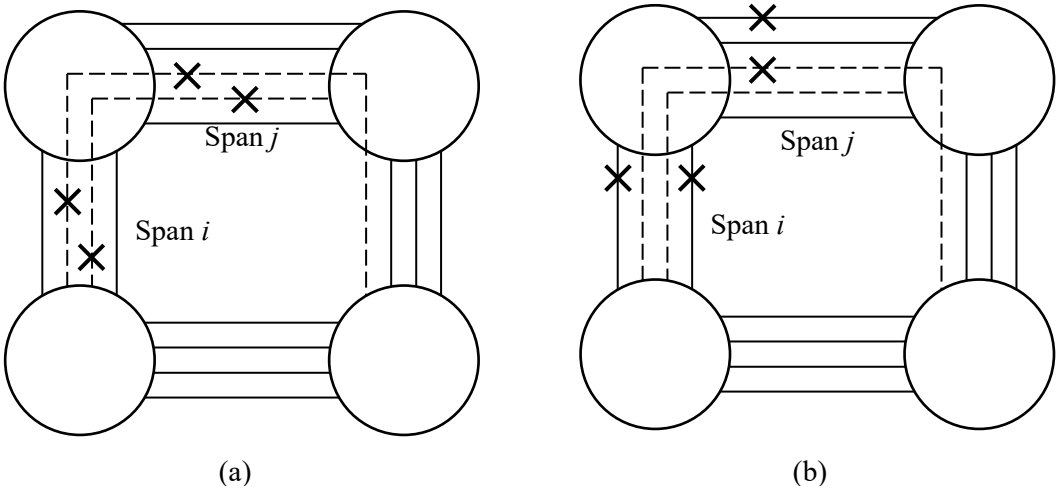


Figure 6.2 – The best-case and worst-case distribution of non-restored working capacity

Based on the best-case distribution of each non-restored working capacity, i.e., the distribution designed towards the smallest number of non-restored service paths, the specific *number of lost paths,* denoted by $NLP(i,j)$, can be used to represent the number of lost service paths (i.e., non-restored service paths), under dual failure (i,j). Therefore, qualitatively, $NLP(i,j)$ is based on the best-case distribution of non-restored working capacity. Quantitatively, however, this best case needs parameterizing to calculate $NLP(i,j)$. The service paths that cross both failed spans simultaneously are employed as an intermediate parameter for quantifying $NLP(i,j)$. Consequently, the specific *number of shared service paths*, denoted by $SP(i,j)$, is defined to represent the number of service paths that traverse span i and span j simultaneously under dual-failure scenario (i,j). Through simple derivation, the value of $NLP(i,j)$ can be acquired by

$$NLP(i,j) = \begin{cases} \max\left(N_{i,j}^{i}, N_{i,j}^{j}\right), & SP(i,j) \geq \min\left(N_{i,j}^{i}, N_{i,j}^{j}\right) \\ N_{i,j}^{i} + N_{i,j}^{j} - SP(i,j), & \text{otherwise} \end{cases} \tag{6-1}$$

Here, $N_{i,j}^{i}, N_{i,j}^{j}$ are the number of units of non-restored working capacity on span i and span j under dual-failure scenario (i,j), respectively. The definition and calculation of $NLP(i,j)$ manifests in such a way that $NLP(i,j)$ is closely related to individual $N_{i,j}^{i}$ and $N_{i,j}^{j}$, rather than their sum. In this regard, we are able to take into account the order of two span failures in a specified dual-failure scenario.

6.3 NEW DUAL-FAILURE SERVICE PATH UNAVAILABILITY

From previous discussion, current methods are span-based due to that NWC_2, R_2, and SPU_2 all concern the non-restored working capacity on each span. This is also the reason for their inaccuracy in measuring dual-failure availability. In our proposed method, the focal point is the individual service path, rather than spans. Due to its relationship with $NLP(i,j)$ and to avoid confusion with the original dual-failure service path SPU_2, we refer to this new dual-failure unavailability as the ***s**pecific-number-of-lost-paths-based **d**ual-failure service path **u**navailability* (SDU). Under this definition, the conditions that a service path is unavailable under a specific dual-failure scenario (i,j) are:

(1) span i and span j are failed physically, and

(2) the lost number of service paths cannot be fully restored.

Hence, $SDU(i,j)$, or, SDU under a specific dual-failure scenario (i,j), can be obtained through Eq. (6-2).

$$SDU(i,j) = U_i^{\text{phy}} \cdot U_j^{\text{phy}} \cdot \frac{NLP(i,j)}{w_i + w_j - SP(i,j)} \tag{6-2}$$

SDU is thus the average of each specific $SDU(i,j)$, as shown in

$$SDU = \frac{\sum_{i,j \in S | i \neq j} SDU(i,j)}{|P|} \quad (6\text{-}3)$$

6.4 MNDF-ML MODEL

In this section, we present our model framework to analyze and evaluate the various availability metrics. This model framework is designed specifically for our purpose in this chapter and has not been reported in the literature. For clarity and simplicity in the coming discussion, ($NWC_2,\ R_2,\ SPU_2, SDU$) will be referred to *tuples*, and individually is *tuple elements*. Before introducing this model framework, we will first state the assumptions, notations, and ILP models related to it.

6.4.1 ASSUMPTIONS

Our interest is to investigate the relationships between *tuple elements*, so whether or not the design itself is optimal with respect to capacity is not our concern. In fact, the design suffices as long as it is feasible, i.e., the restoration scheme works in restoring the failed spans with the reserved network designed for full single-failure restorability. We simply wish to show the relative values of the availability indicators for some network as designed, optimal or otherwise. Except the assumptions specified in Section 3.6, we assume all the spans have the same physical failure rate, i.e., we make use of an average physical failure rate for all spans.

6.4.2 NOTATIONS

Sets, parameters, and variables used in this model are illustrated as follows:

Sets:

S is the set of spans in the network.

B_i is the set of eligible backup routs for span i under single-failure scenario i.

Parameters:

w_i is the amount of working capacity on span i.

c_k is the cost per unit capacity on span k.

$\delta_{i,k}^{b}$ is a binary variable and takes 1 if backup route b of span i crosses span k.

C_∞ is a positive large constant.

B is the minimum cost to guarantee full single-failure restorability.

NWC_{tar} is the target value of the objective function at the first level.

$L(m)$ is the level of each model in the iterations.

$step$ is the difference of the lower bound of NWC_2 between adjacent levels.

Variables:

s_k is the amount of spare capacity on span k.

f_i^b is the amount of flow routed on backup route b for restoration of span i under single-failure i.

$f_{i,j}^{b,i}$ is the amount of flow routed on backup route b for restoration of span i under dual-failure (i,j).

6.4.3 MNDF-SL FORMULATION

The core ILP model in this chapter is the *single level minimum non-restored working capacity under dual-failure scenarios* (MNDF-sl) ILP model. The objective function of this new model is to minimize the total non-restored working capacity towards the target value under dual-failure scenarios, with the available resources tailored for full single-failure restorability. The ILP model itself proceeds as follows:

Minimize:

$$NWC_2 = \sum_{i,j \in S | i \neq j} NWC_2(i,j) \tag{6-4}$$

Subject to:

$$NWC_2 \geq NWC_{\text{tar}} + step \cdot (L(m) - 1) \tag{6-5}$$

$$NWC_2(i,j) = w_i + w_j - \sum_{b \in B_i} f_{i,j}^{b,i} - \sum_{b \in B_j} f_{i,j}^{b,j} \quad \forall i,j \in S | i \neq j \tag{6-6}$$

$$\sum_{b \in B_i} f_i^b \geq w_i \quad \forall i \in S \tag{6-7}$$

$$s_k \geq \sum_{b \in B_i} f_i^b \cdot \delta_{i,k}^b \quad \forall i,k \in S | i \neq k \tag{6-8}$$

$$\sum_{b \in B_i} f_{i,j}^{b,i} \leq w_i \quad \forall i,j \in S | i \neq j \tag{6-9}$$

$$\sum_{b \in B_j} f_{i,j}^{b,j} \leq w_j \quad \forall i,j \in S | i \neq j \tag{6-10}$$

$$s_k \geq \sum_{b \in B_i} f_{i,j}^{b,i} \cdot \delta_{i,k}^b + \sum_{b \in B_j} f_{i,j}^{b,j} \cdot \delta_{j,k}^b \quad \forall i,j,k \in S | i \neq j \neq k \tag{6-11}$$

$$f_{i,j}^{b,i} \leq C_\infty \cdot \left(1 - \delta_{i,j}^b\right) \quad \forall i,j \in S, b \in B_i | i \neq j \tag{6-12}$$

$$f_{i,j}^{b,j} \leq C_\infty \cdot \left(1 - \delta_{j,i}^b\right) \quad \forall i,j \in S, b \in B_j | i \neq j \tag{6-13}$$

$$\sum_{k \in S} c_k . s_k \leq B \tag{6-14}$$

$$R_2(i,j) = 1 - \frac{NWC(i,j)}{w_i + w_j} \qquad \forall i,j \in S | i \neq j \tag{6-15}$$

$$N_{i,j}^{i} = w_i - \sum_{b \in B_i} f_{i,j}^{b,i} \qquad \forall i,j \in S | i \neq j \tag{6-16}$$

$$N_{i,j}^{j} = w_j - \sum_{b \in B_j} f_{i,j}^{b,i} \qquad \forall i,j \in S | i \neq j \tag{6-17}$$

Constraint (6-4) confines the lower bound of the objective function so that the network can be designed close to this value in each level. Constraint (6-6) calculates the amount of non-restored working capacity under each dual-failure scenario (i,j), by subtracting the restored working capacity from the original working capacity affected by the dual-failure scenarios. Constraint (6-7) asserts sufficient flow for restoring single-failure fully through enabling the sum of flow on all the restoration routes of each span to be larger than its working capacity value. Constraint (6-8) translates the flow requirement from the restoration routes to the amount of spare capacity of the corresponding span. This guarantees enough amount of spare capacity on each span for full single-failure restorability. Constraint (6-9) and (6-10) assign the restoration flow on the restoration routes to each failed span in each specific dual-failure scenario. Similar to single-failure scenario, Constraint (6-11) translates the flow requirements on the restoration routes from Constraints (6-9) and (6-10) to the amount of spare capacity of each span. Constraints (6-12) and (6-13) asserts that the flow on the two failed spans in a given dual-failure scenario cannot be used to restore the failed working flow on each other, owing to their simultaneous failure. Constraint (6-14) ensures that the sum of spare capacity on all the spans should not excess the budgeted resources. In this chapter, the available budget is the capacity resources

that utilized to afford full single-failure restorability. Constraint (6-15) computes dual-failure restorability for each dual-failure scenario, which will be used to calculate R_2 at the calculation model. Constraints (6-16) and (6-17) calculate non-restored working capacity on each failed span under dual-failure scenario, which will be employed to calculate NLP at calculation model. Additional constraints not shown, namely integer and non-negativity, are included in the model as well.

6.4.4 ASSISTANT ILP MODELS

To obtain the values for B and NWC_{tar}, two subsidiary ILP models are used. The *minimum cost single-failure* (MCSF) model from [24] is employed to obtain the value of B and the *minimum non-restored working capacity under dual-failure* (MNDF) model from [79] is applied to acquire the value of NWC_{tar}. The MCSF model aims to minimize total cost while satisfying full single-failure restorability [24]. Thus, its ILP objective function is to minimize $Cost = \sum_{i \in S} c_k \cdot s_k$, and it utilizes only constraints (5-7) and (5-8), above. Its objective function value is used as the budget limit in the MNDF model and our MNDF-sl ILP model. The MNDF model seeks to minimize the total non-restored working capacity over all dual-failure combinations in the network [79]. Its objective function is to minimize $NWC_2 = \sum_{i,j \in S | i \neq j} NWC_2(i,j)$, and it utilizes constraints (6-5) through (5-14). The value of the objective function can herein be used as the value of NWC_{tar} in our MNDF-sl model.

6.4.5 MNDF-ML MODEL FRAMEWORK

We can present our new model framework, which we call the *multiple level minimum non-restored working capacity under dual-failure scenarios* (MNDF-ml) model framework. Figure 6.3 shows the framework of this model framework and the interactions among its components.

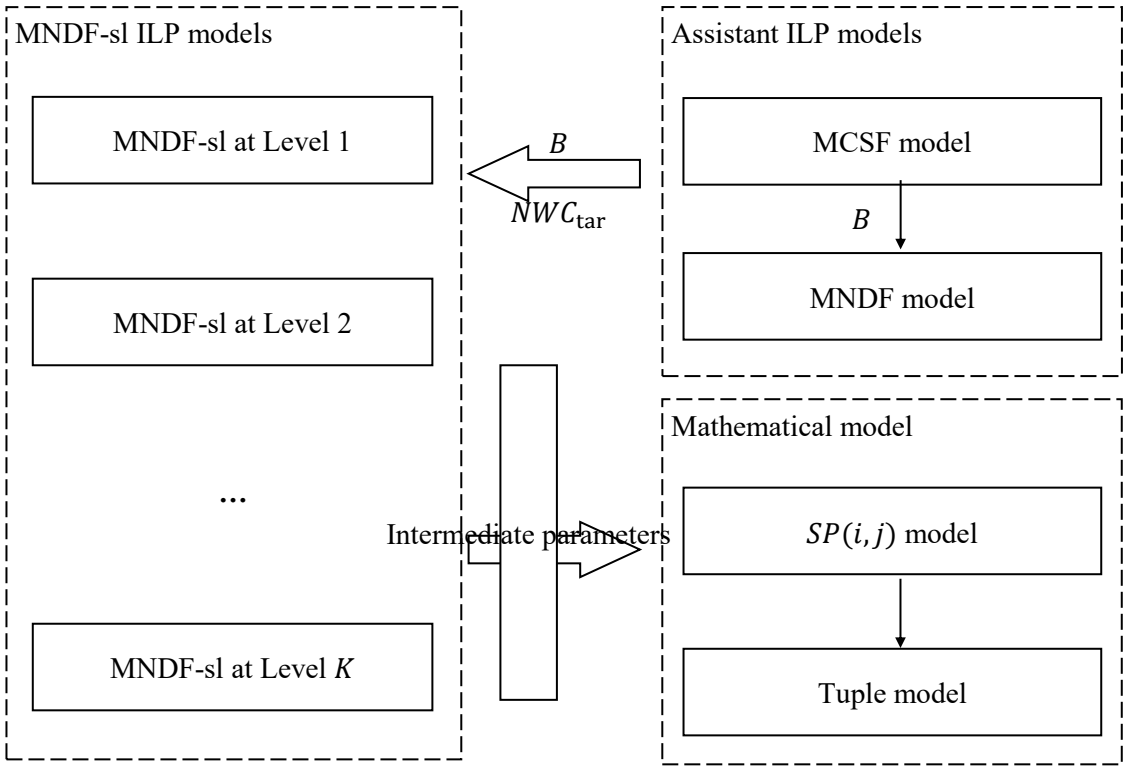


Figure 6.3 – The framework of MNDF-ml model

As shown in Figure 6.3, the MNDF-ml model framework is made up of a series of our MNDF-sl ILP model, the MCSF and MNDF ILP models, and two mathematical calculating models (one for calculating $SP(i,j)$ and one for calculating tuples). To obtain a series of tuple values, NWC_2 is chosen as the reference to calculate the remainder of the tuple elements. More specifically, we use NWC_2 as the objective to perform the network

design, and then calculate the other tuple elements after the design. Each instance of our MNDF-sl ILP model is used to obtain one NWC_2 value and the values of corresponding intermediate variables that are necessary to calculate the other tuple elements. However, one tuple value cannot adequately reflect the relationships among its elements (i.e., their relative trends), so multiple tuple values are needed. Let K $(K \geq 3)$ be the number of tuples for analyzing the relationships among the tuple elements. Each tuple corresponds to one ILP model, so K tuples corresponds to K ILP models. To distinguish the various constituent ILP models, each model is referred to as a *level* and denoted by $L(k)$ $(k = 1,2,\cdots,K)$. The MCSF model provides the value of budget B to MNDF model, and the MNDF model produces the value of NWC_{tar} to each instance of our MNDF-sl ILP model. K groups of intermediate data are generated by K MNDF-sl ILP models and passed to the mathematical models to calculate the final tuple value.

6.5 MNDF-ML MODEL IMPLEMENTATION

6.5.1 ITERATIVE MNDF-ML APPROACH

As previously stated, the MNDF-ml model consists of three major components. The first is a series of MNDF-sl implementations. Each instance of the MNDF-sl model obtains the value of B from the MCSF model and the value of NWC_{tar} from the MNDF model. After solving each instance of MNDF-sl, the results are used to calculate the value of the *tuple* at each iteration. The MNDF-ml model is illustrated in Figure 6.4.

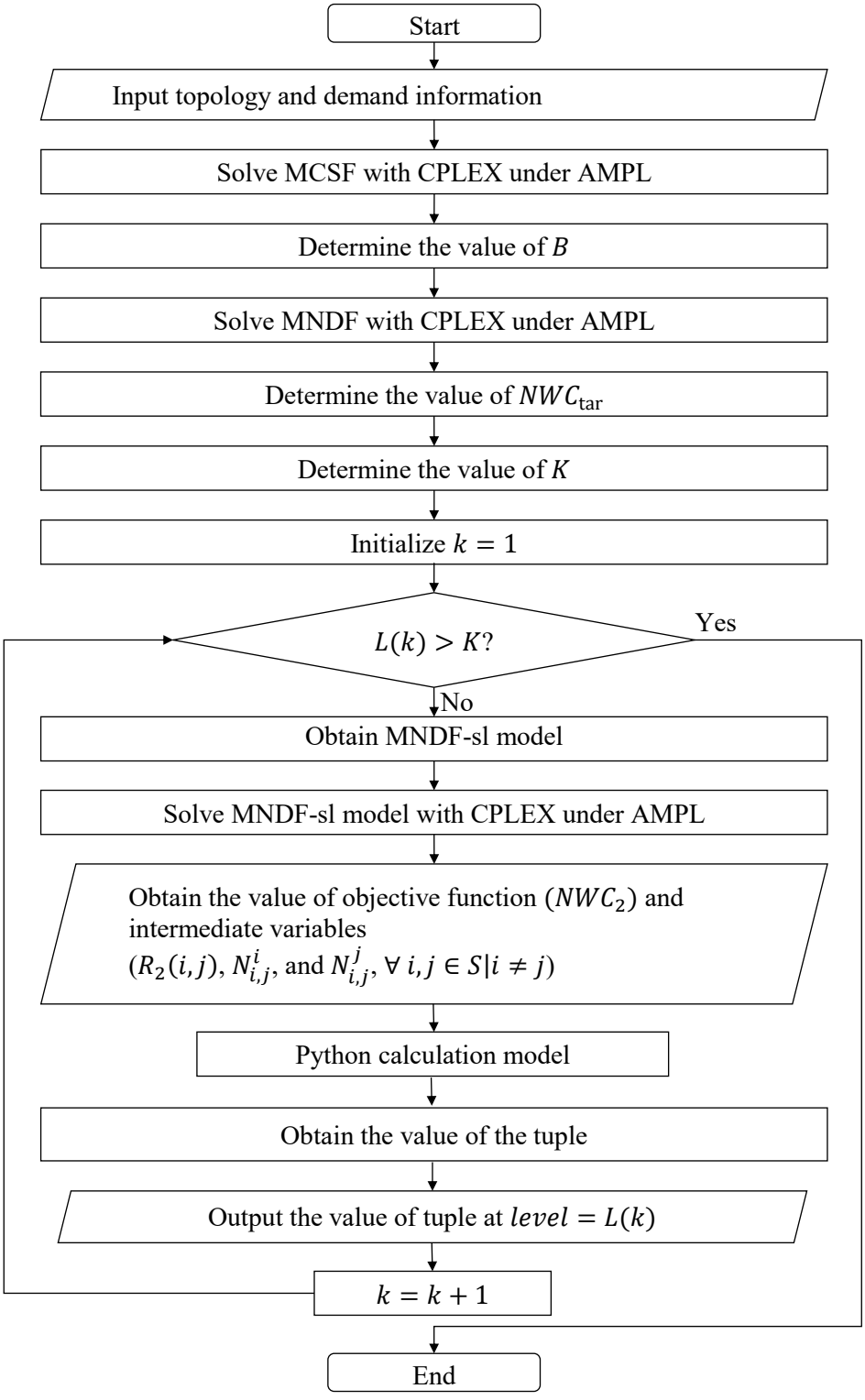


Figure 6.4 – The flowchart of calculating MNDF-ml model

First, the MCSF model is solved using a known network topology and traffic demands, and solved for minimum cost under full single-failure restorability. The minimized cost obtained is the value subsequently used for B.

Second, the MNDF model is solved using the previously obtained value of B to determine the minimum non-restored working capacity. The value obtained here is used as NWC_{tar}.

Then we determine the values of K and $step$, initialize the value of k, and cycle through k to solve each subsequent MNDF-sl model. We solve for $R_2(i,j)$, $N_{i,j}^{i}$, and $N_{i,j}^{j}$ under each dual-failure scenario (i,j). Once the loop index k is larger than that of the target loop size K, we exit to calculate the value of the *tuple*.

The MCSF and MNDF models are implemented in AMPL and solved with CPLEX 12.6.2 on a 12-core ACPI multiprocessor X64-based PC with Intel Xeon® CPU E5-2430 running at 2.2 GHz with 95 GB RAM. The iterative *tuple* calculations are implemented in Python 2.7.

6.5.2 IMPLEMENTATION OF CURRENT METHODS

Because the value of NWC_2 can be obtained from the objective function value of MNDF-sl model, only R_2 and SPU_2 remain to be calculated. The calculation procedure for R_2 and SPU_2 is illustrated in Figure 6.5. As in Figure 6.4 previously, K groups of data are obtained from K instances of the MNDF-sl model and used to produce tuple values. Within each group of data, the values of NWC_2, $|S|$ and w_i $(i \in S)$ are required to calculate R_2.

The value of NWC_2 can be obtained from the MNDF-sl model, and the values of $|S|$ and w_i ($i \in S$) can be obtained from the topology and demand inputs.

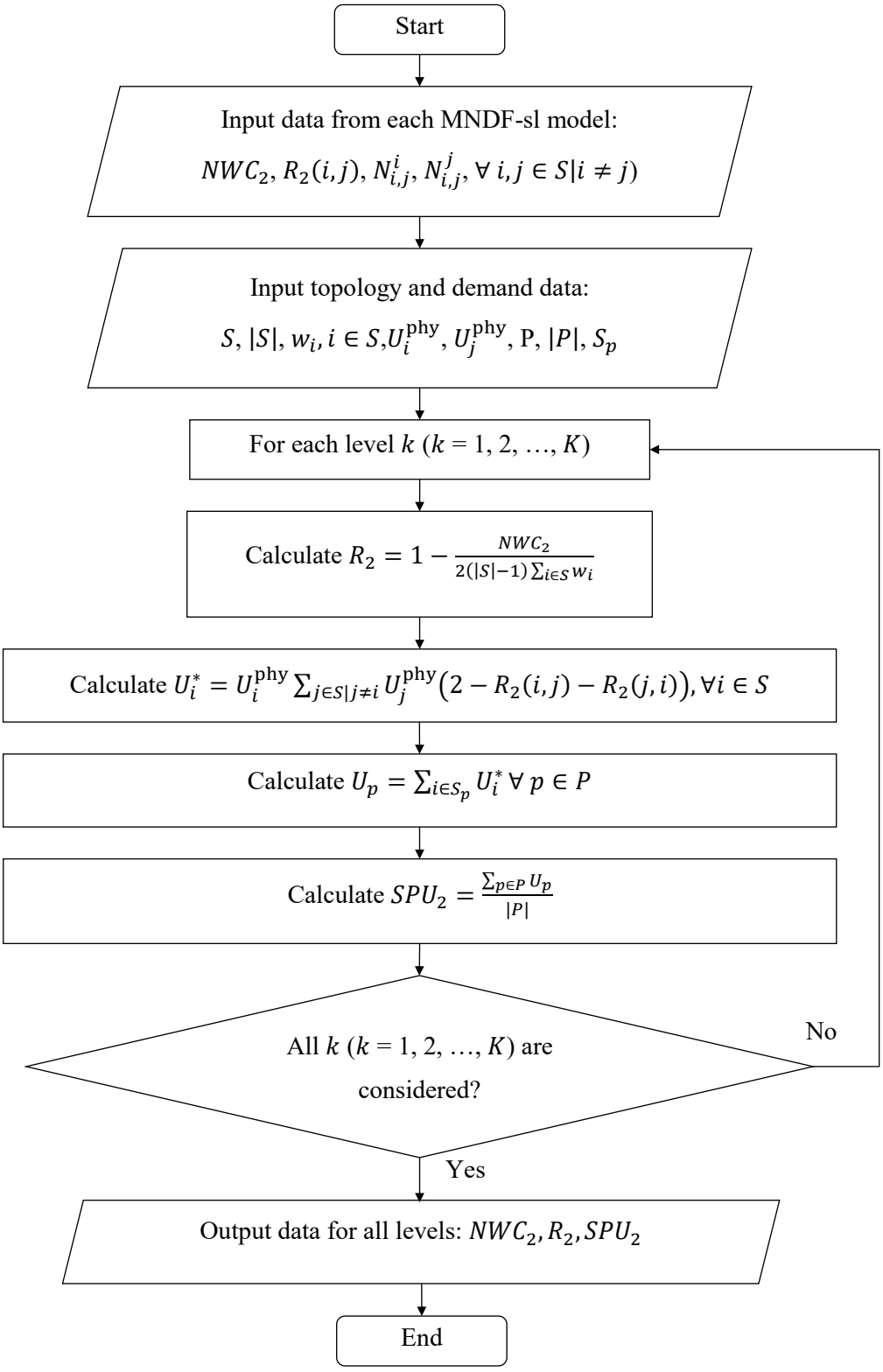


Figure 6.5 – Implementation for current availability methods

6.5.3 IMPLEMENTATION OF NEW METHOD

According to the calculation formulas for R_2 and SPU_2, all the variables and parameters can be obtained through the MNDF-sl model directly. SDU, however, cannot be obtained simply, as one of its intermediate calculations, $SP(i,j)$, is not known explicitly from the MNDF-sl model. Herein SDU cannot be calculated with direct iteration without the value of $SP(i,j)$. For two arbitrary distinct spans i,j in the network, $SP(i,j)$ can be obtained by adding up the demands of all the working routes that cross over i,j simultaneously. Figure 6.6 depicts the pseudo code for calculating $SP(i,j)$ in detail, where `getSP()` is the function for calculating $SP(i,j)$ for span i,j. $SP(i,j)$ is represented by `sp`, and we use three input arguments: `i` and `j`, which are the two unique spans in question, and `routes`, which is the set of working routes for each demand. For simplification, we assume all working traffic between any origin-destination node pair is routed through a single working route. Each working route is acquired through Dijkstra's algorithm [116] and expressed by a set of ordered spans. Thus, `route` indicates the individual working route composed by an ordered set of spans. The theory is to iterate through all the working routes in the network in search of the service routes crossing i and j. This process is implemented in Python 2.7.

```
01  function getSP(i, j, routes)
02      sp ← 0
03      for route in routes
04          if i in route and j in route
05              sp ← sp + route.demands
06          end if
07      end for
08  end function
```

Figure 6.6 – Pseudo code for calculating shared service paths

After obtaining $SP(i,j)$ for $(i,j \in S|i \neq j)$ in each level, SDU can be calculated through the procedure described in Figure 6.7. The values of $N_{i,j}^{i}$ and $N_{i,j}^{j}$ for $(i,j \in S|i \neq j)$ are obtained from the corresponding MNDF-sl model, and the values of $SP(i,j)$ for $(i,j \in S|i \neq j)$ are obtained from the above-mentioned mathematical calculations. As shown in the figure, if at any level the value of $SP(i,j)$ for $(i,j \in S|i \neq j)$ is larger than the smaller of $N_{i,j}^{i}$ and $N_{i,j}^{j}$, then the value of $NLP(i,j)$ is set to the larger of $N_{i,j}^{i}$ and $N_{i,j}^{j}$; otherwise, $NLP(i,j)$ is set to the sum of $N_{i,j}^{i}$ and $N_{i,j}^{j}$ subtracting $SP(i,j)$. $SDU(i,j)$ is then calculated for $(i,j \in S|i \neq j)$, and overall SDU is calculated as the average of $SDU(i,j)$ over $(i,j \in S|i \neq j)$. This calculation process is also implemented in Python 2.7.

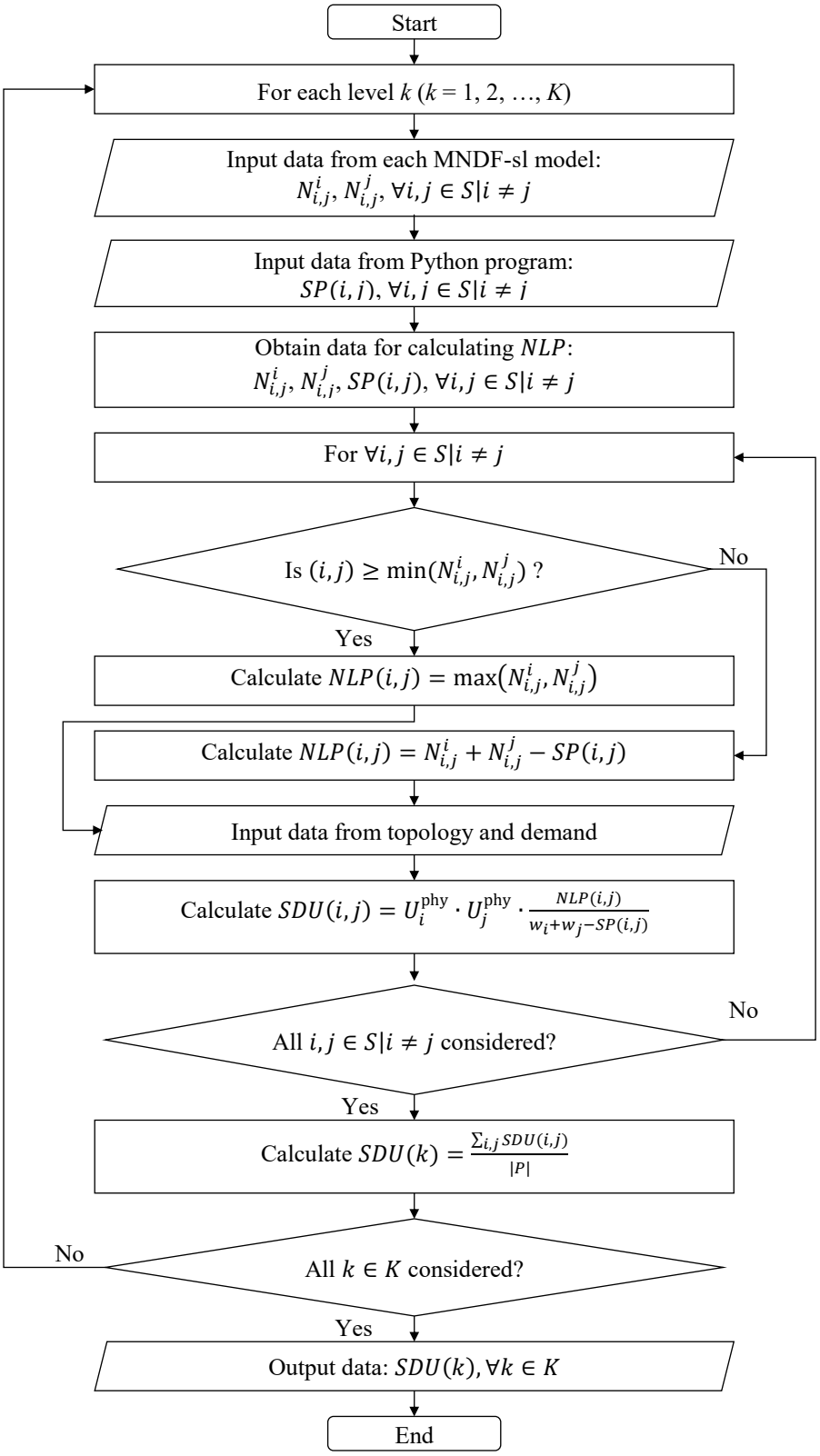
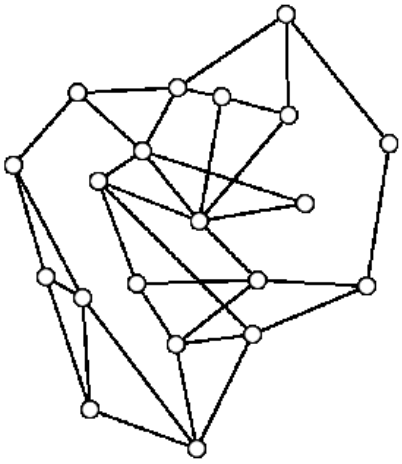


Figure 6.7 – Implementation for new dual-failure unavailability

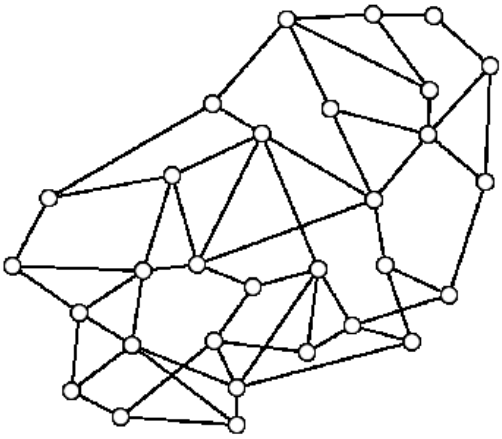
6.6 EXPERIMENTS

6.6.1 EXPERIMENTAL NETWORKS

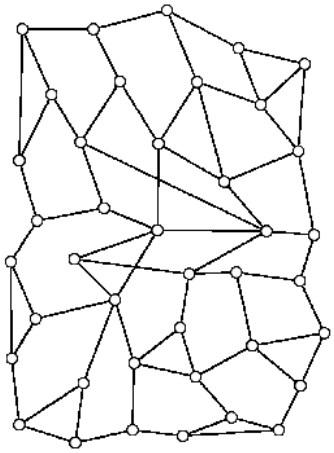
To evaluate the relationships among the aforementioned availability metrics, experiments were conducted using six networks with a variety of topologies and sizes. Networks (a) and (b) are reported in [18], and Networks (c) through (f) are created by ourselves. The topology of all the experimental networks are depicted in Figure 6.8. We assume the physical unavailability of each span is 3×10^{-4} [26], and the value of K is 20. The $step$ is 10 and 100 for the detailed range and general range (which will be explained in the results), respectively.



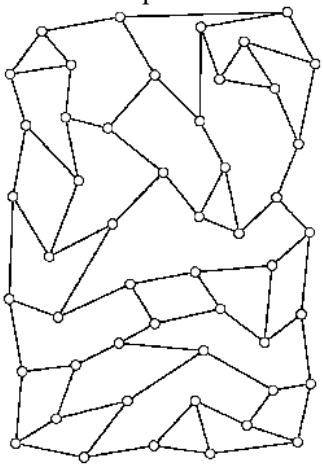
20-node 35-span network



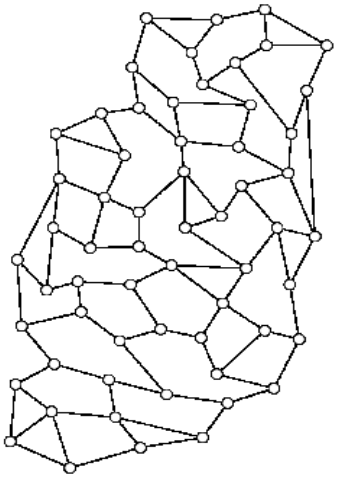
30-node 55-span network



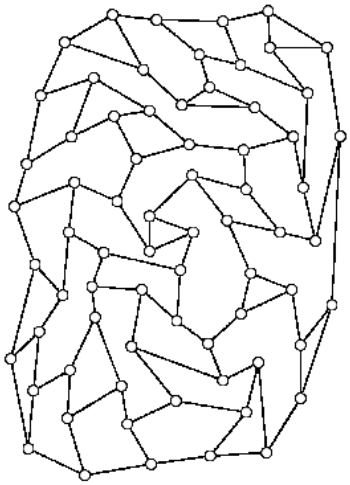
40-node 68-span network



50-node 75-span network



60-node 96-span network



70-node 105-span network

Figure 6.8 – The topology of experimental networks

6.6.2 EXPERIMENTAL RESULTS

Experimental results are shown from Figure 6.9 through Figure 6.32. Within each figure, the x-axis represents the number of non-restored working capacity in the event of dual-failures. The y-axis represents dual-failure restorability and dual-failure service path unavailability, or new dual-failures unavailability, respectively, with increasing non-restored spare capacity in the network. With respect to each experimental network, the first figure shows the variation trend of R_2 and SPU_2 with the changing values of NWC_2; the second figure shows the changing trend of SDU with the changing values of NWC_2; the third figure zooms in the first figure to display details; the fourth figure zooms in the second figure to describe its details. We are doing this zoom-in is because in the original figures, the span of the number of non-restored working capacity in the x-axis is very large if we want to look at the relationship between the total non-restored working capacity and the other indicators closely.

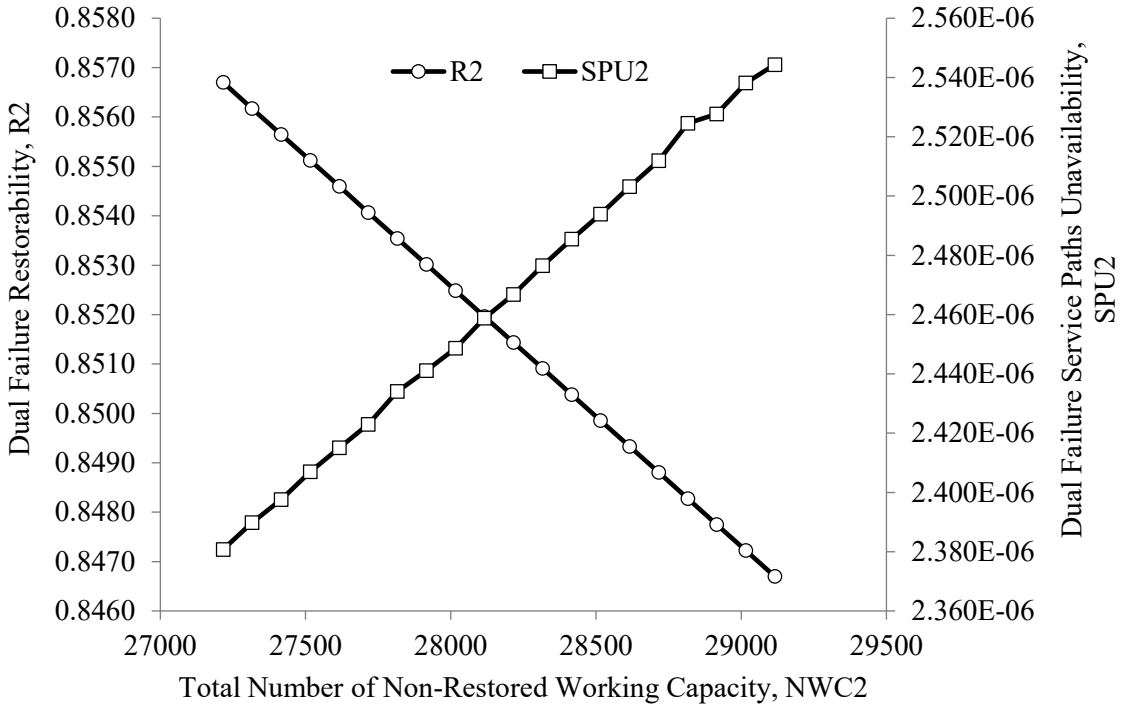


Figure 6.9 – Dual failure restorability and service path unavailability for 20-node 35-span network

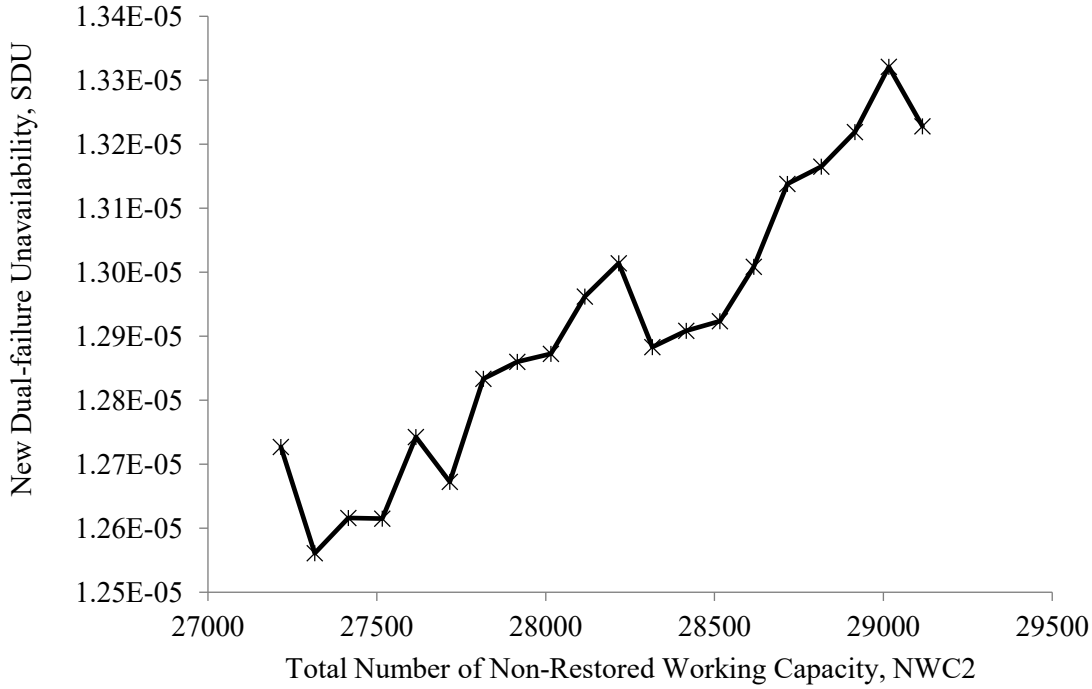


Figure 6.10 – New dual failure restorability for 20-node 35-span network

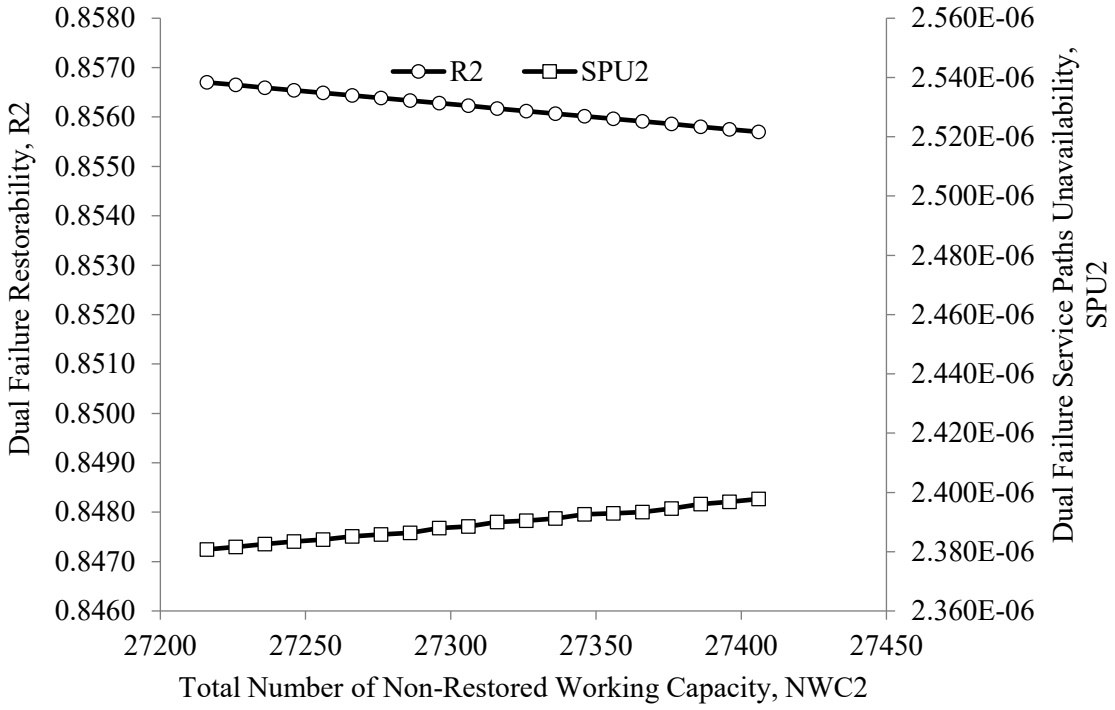


Figure 6.11 – Dual failure restorability and service path unavailability for 20-node 35-span network (zoom-in version)

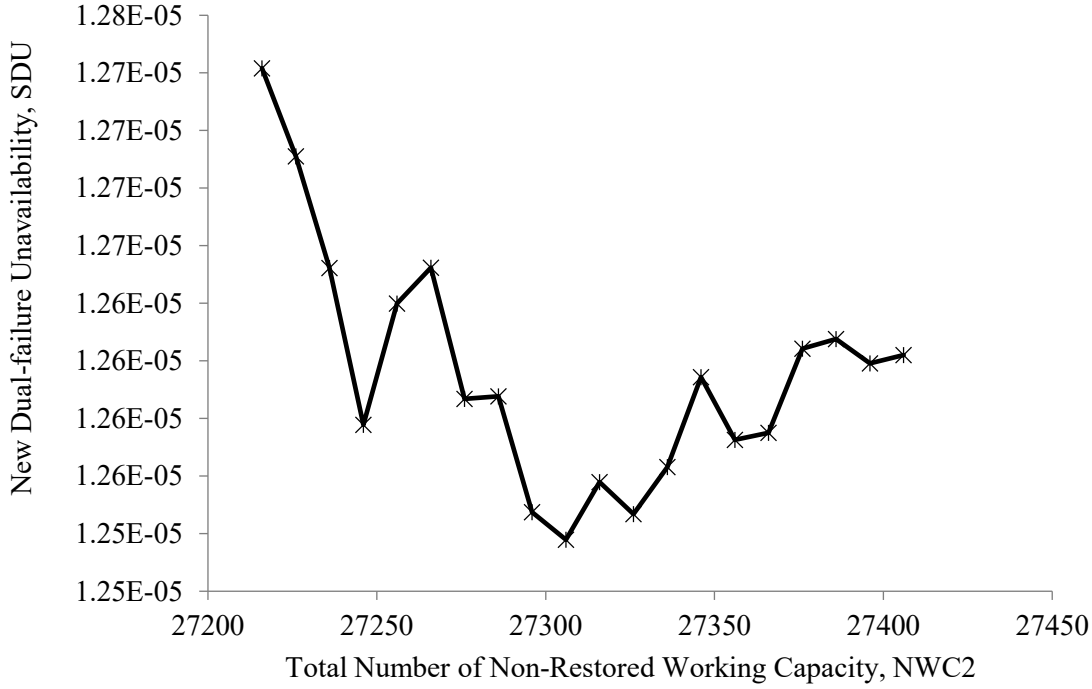


Figure 6.12 – New dual failure restorability for 20-node 35-span network (zoom-in version)

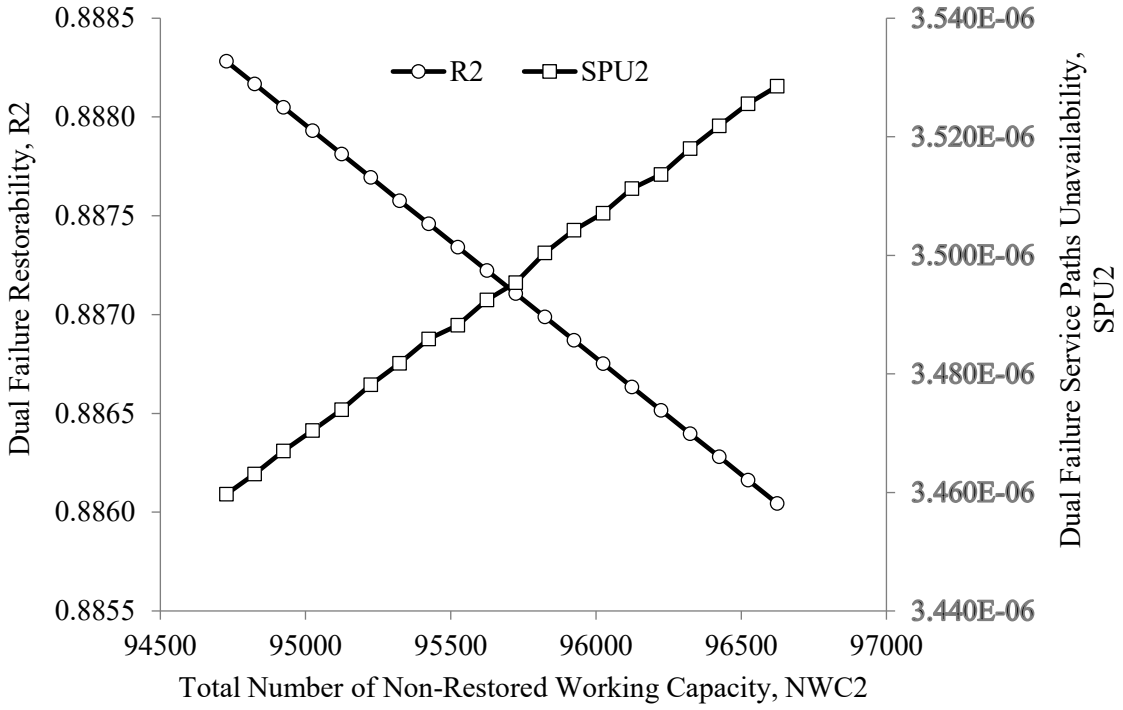


Figure 6.13 – Dual failure restorability and service path unavailability for 30-node 55-span network

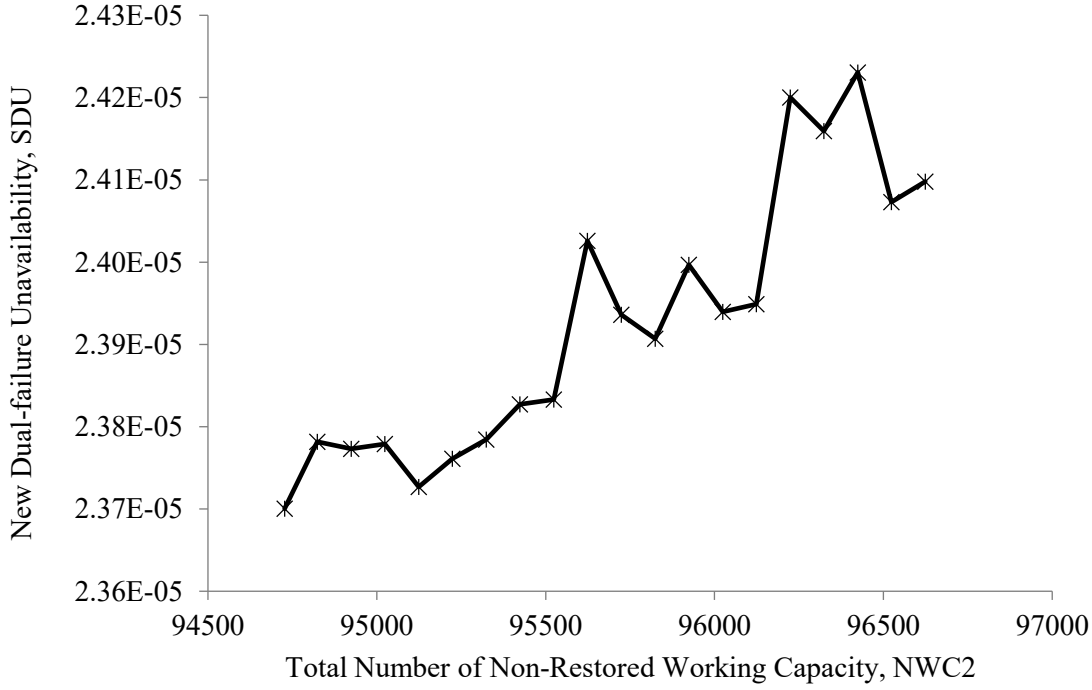


Figure 6.14 – New dual failure restorability for 30-node 55-span network

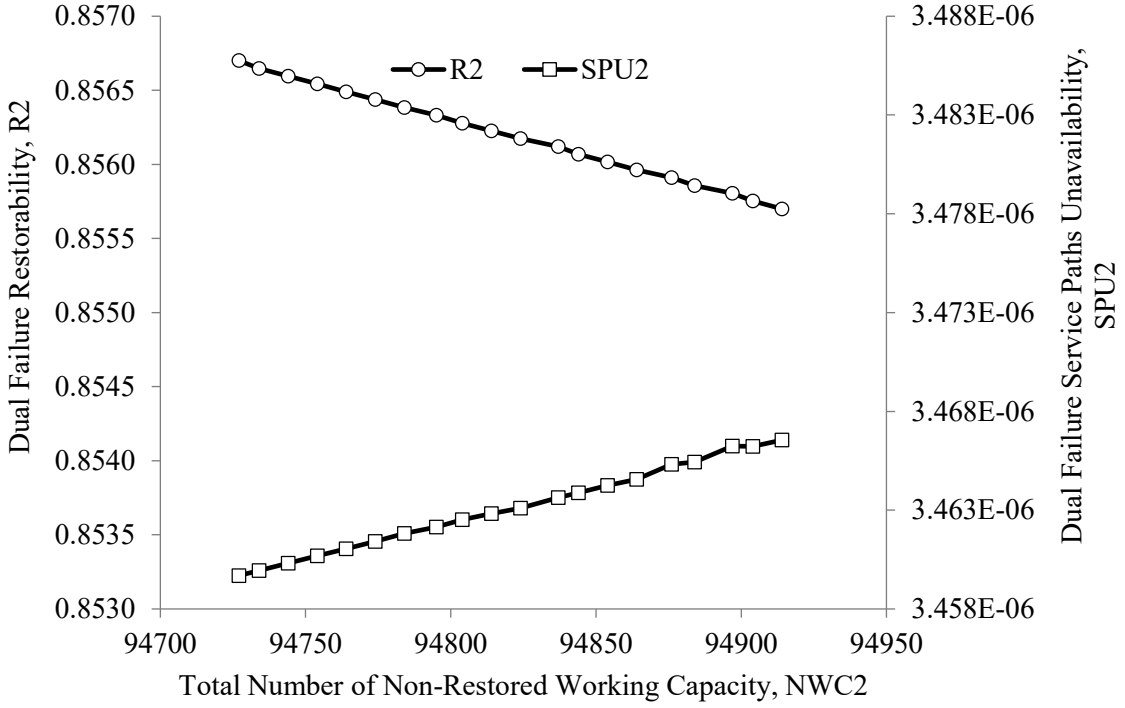


Figure 6.15 – Dual failure restorability and service path unavailability for 30-node 55-span network (zoom-in version)

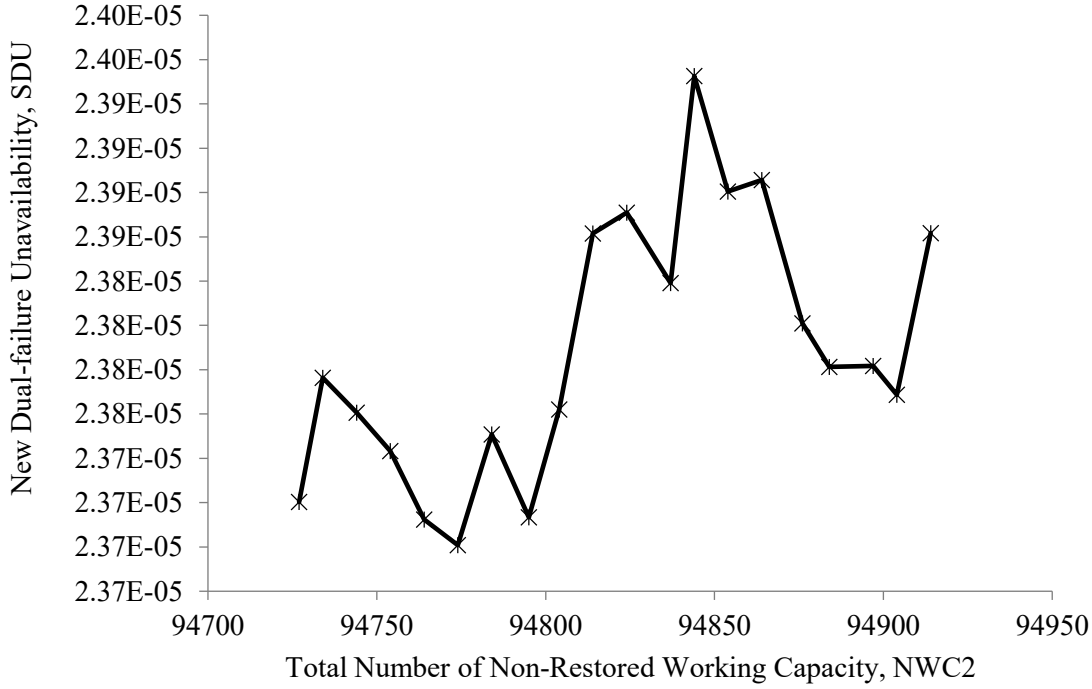


Figure 6.16 – New dual failure restorability for 30-node 55-span network (zoom-in version)

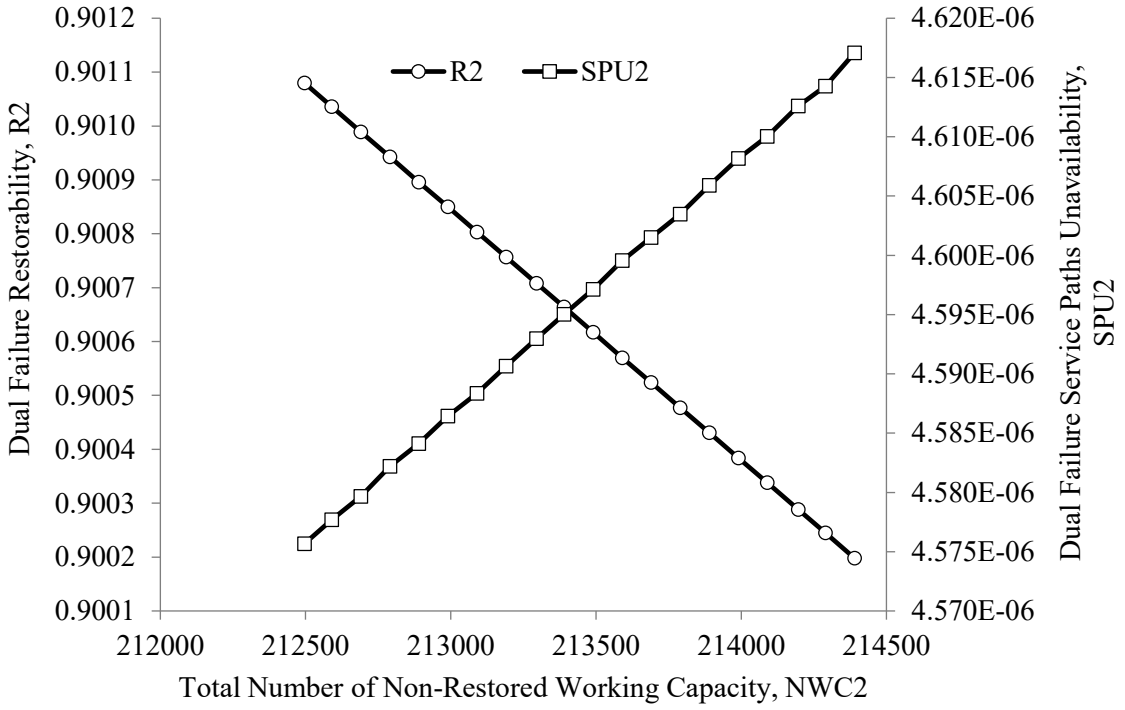


Figure 6.17 – Dual failure restorability and service path unavailability for 40-node 68-span network

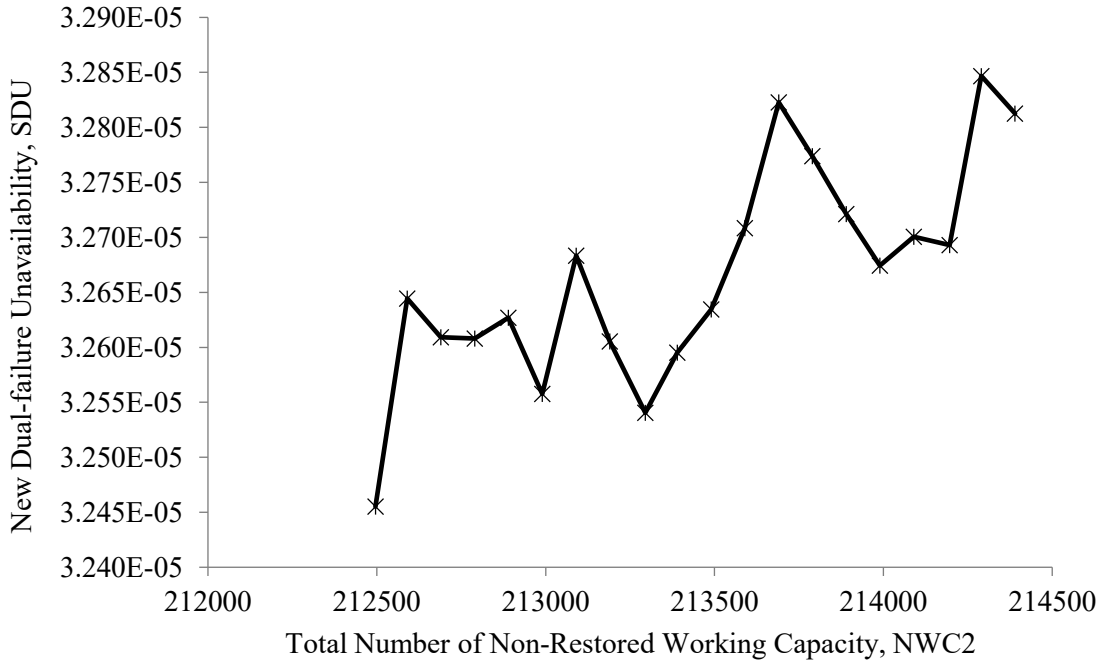


Figure 6.18 – New dual failure restorability for 40-node 68-span network

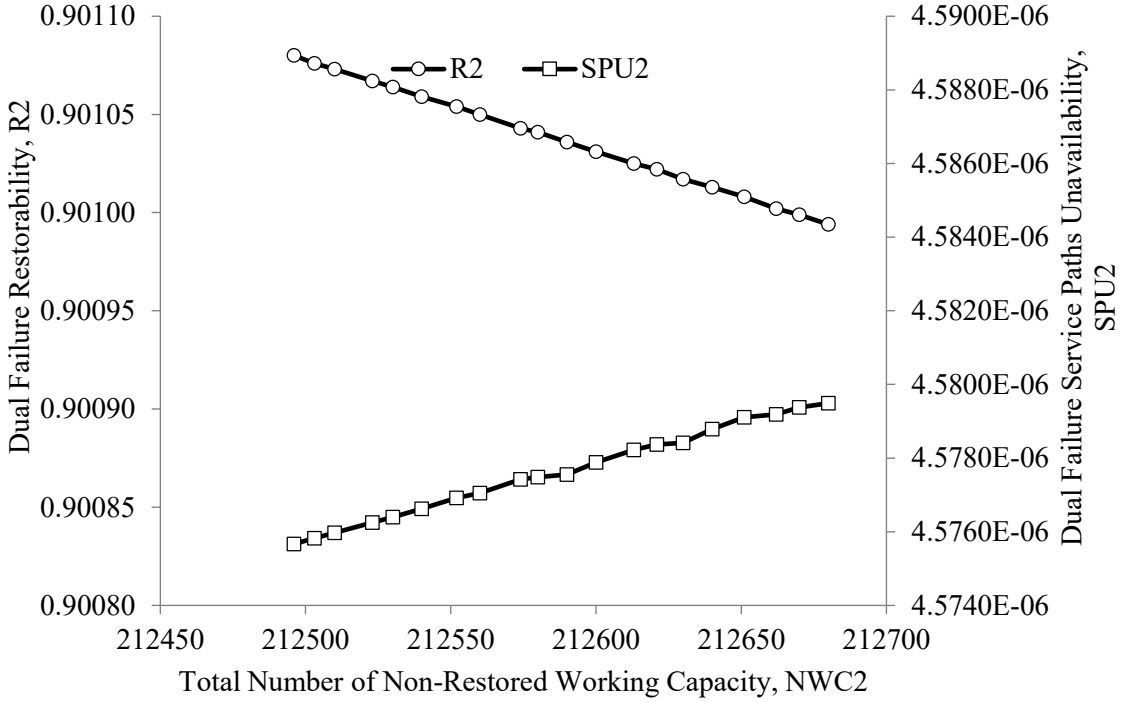


Figure 6.19 – Dual failure restorability and service path unavailability for 40-node 68-span network (zoom-in version)

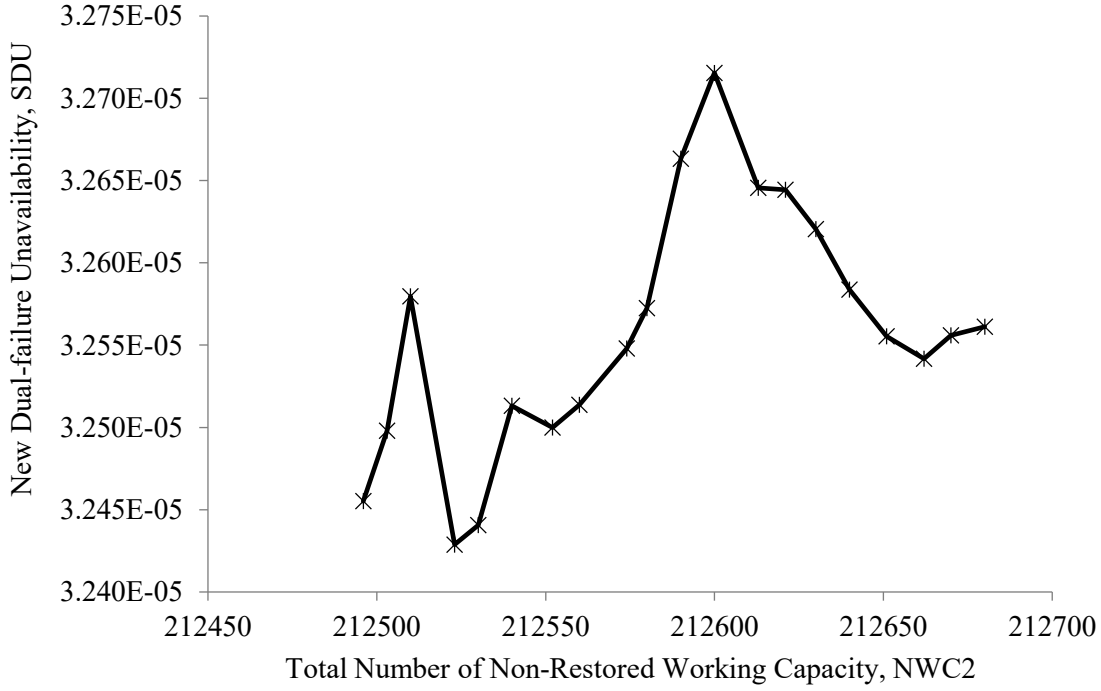


Figure 6.20 – New dual failure restorability for 40-node 68-span network (zoom-in version)

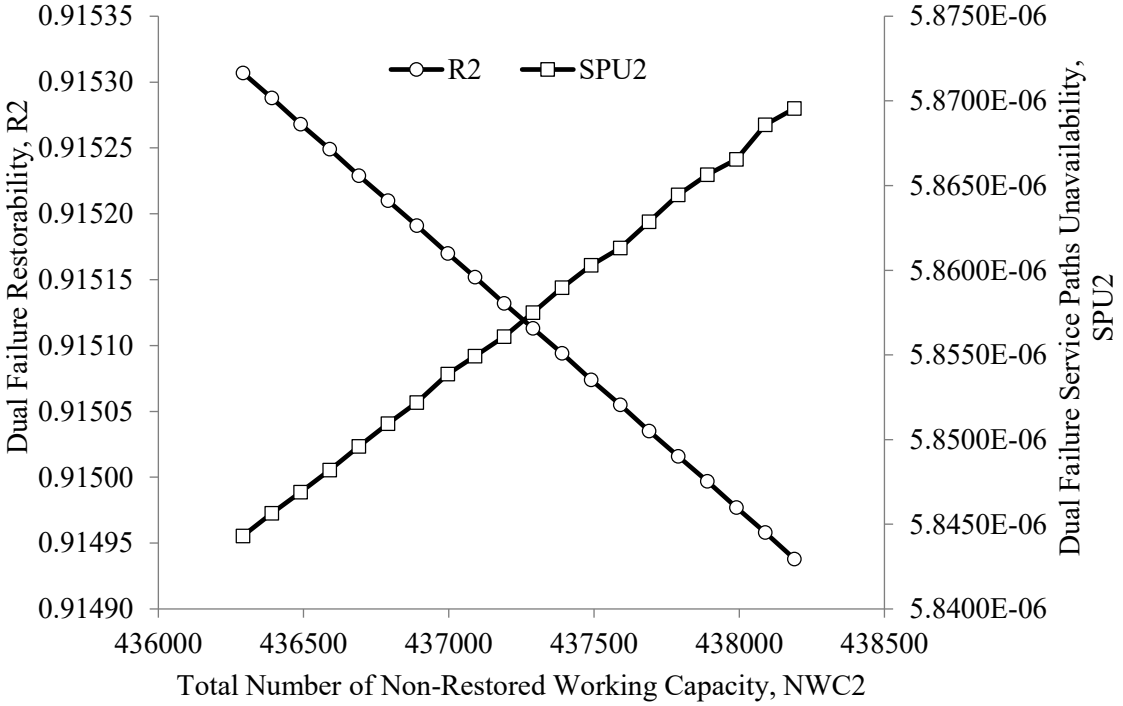


Figure 6.21 – Dual failure restorability and service path unavailability for 50-node 75-span network

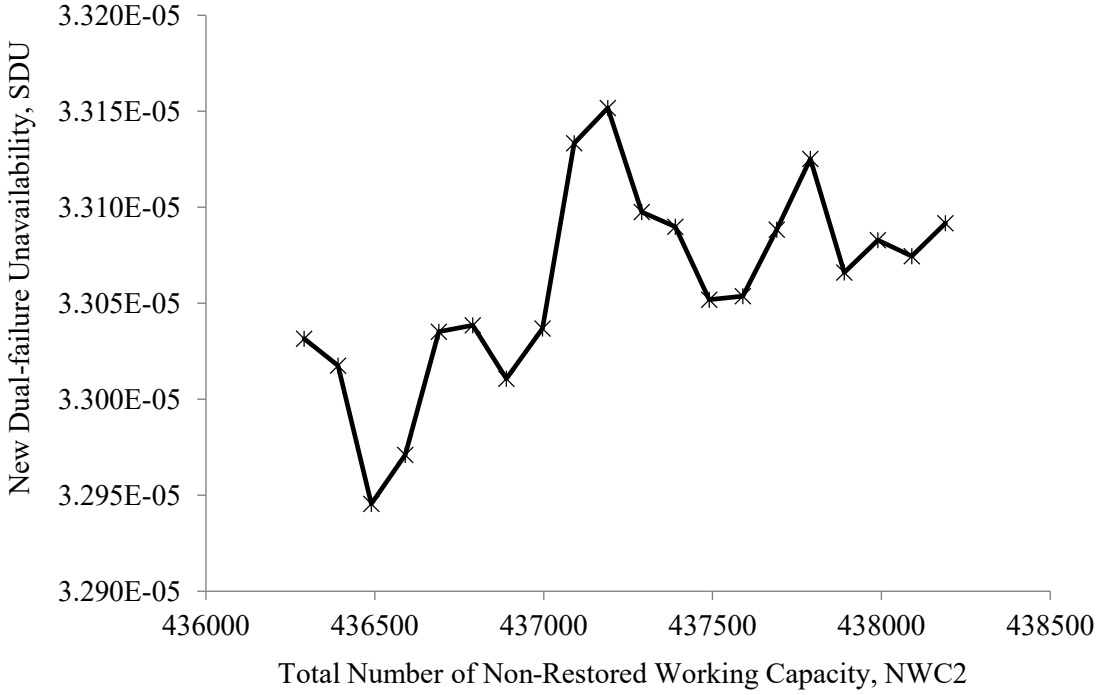


Figure 6.22 – New dual failure restorability for 50-node 75-span network

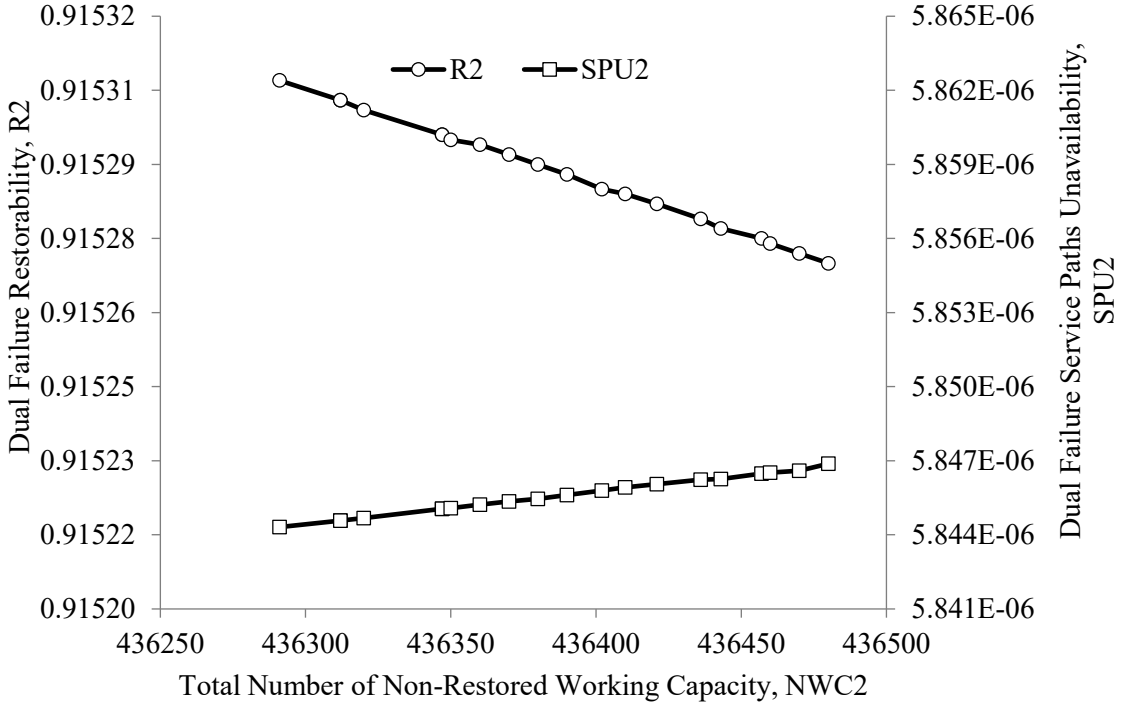


Figure 6.23 – Dual failure restorability and service path unavailability for 50-node 75-span network (zoom-in version)

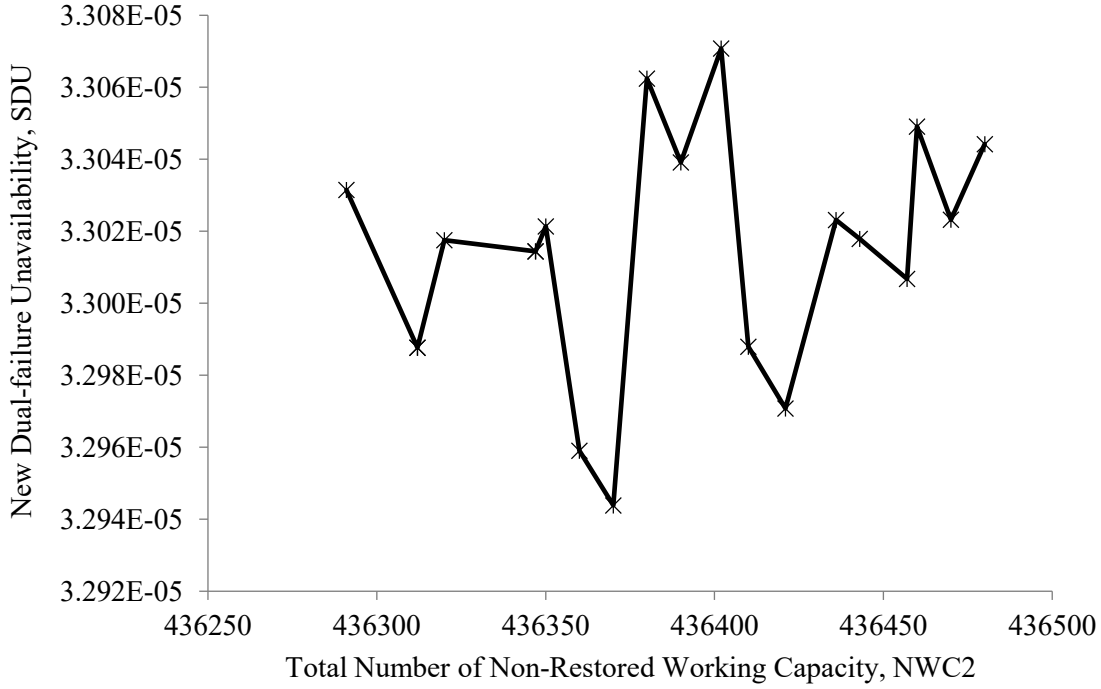


Figure 6.24 – New dual failure restorability for 50-node 75-span network (zoom-in version)

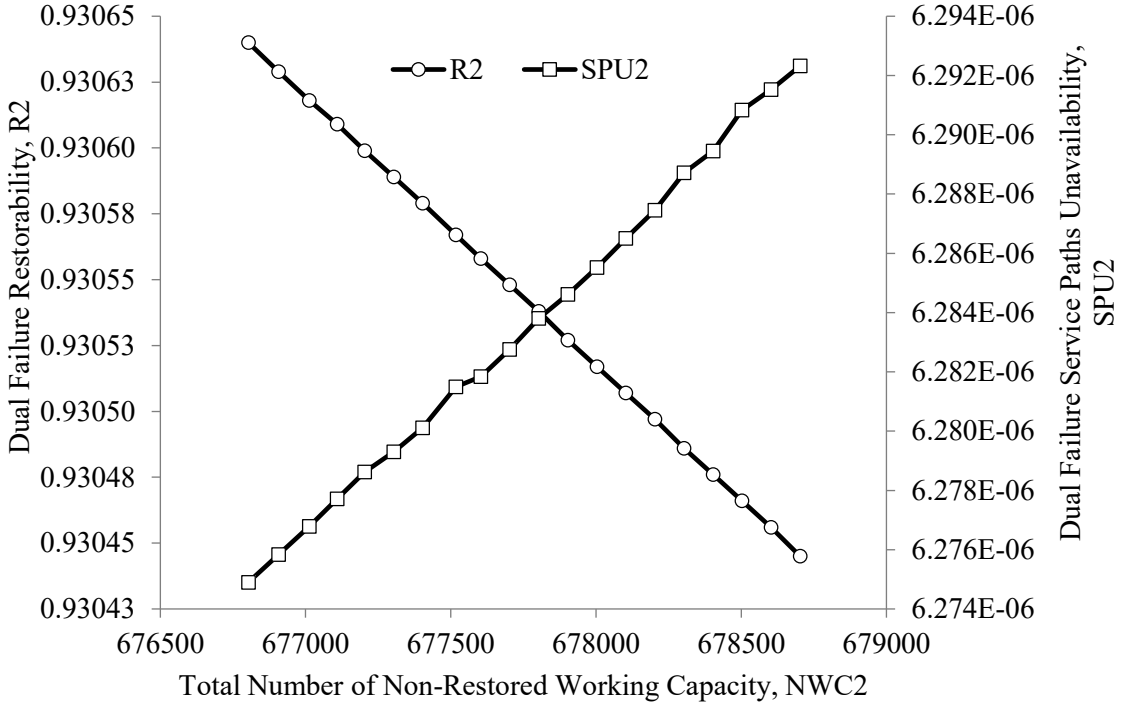


Figure 6.25 – Dual failure restorability and service path unavailability for 60-node 96-span network

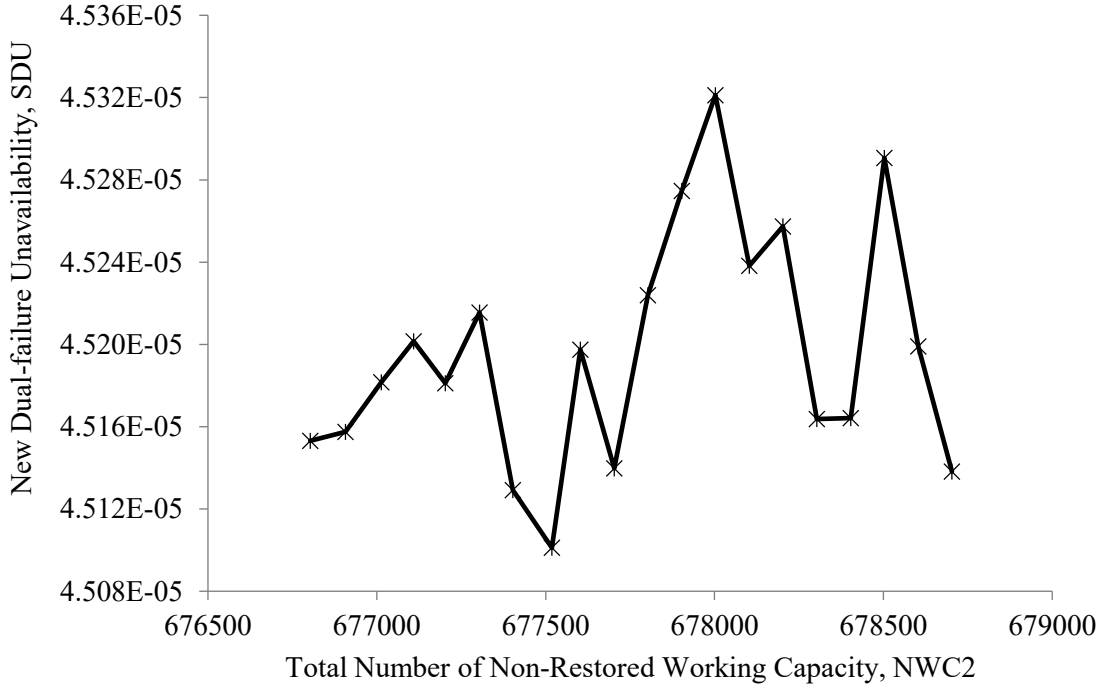


Figure 6.26 – New dual failure restorability for 60-node 96-span network

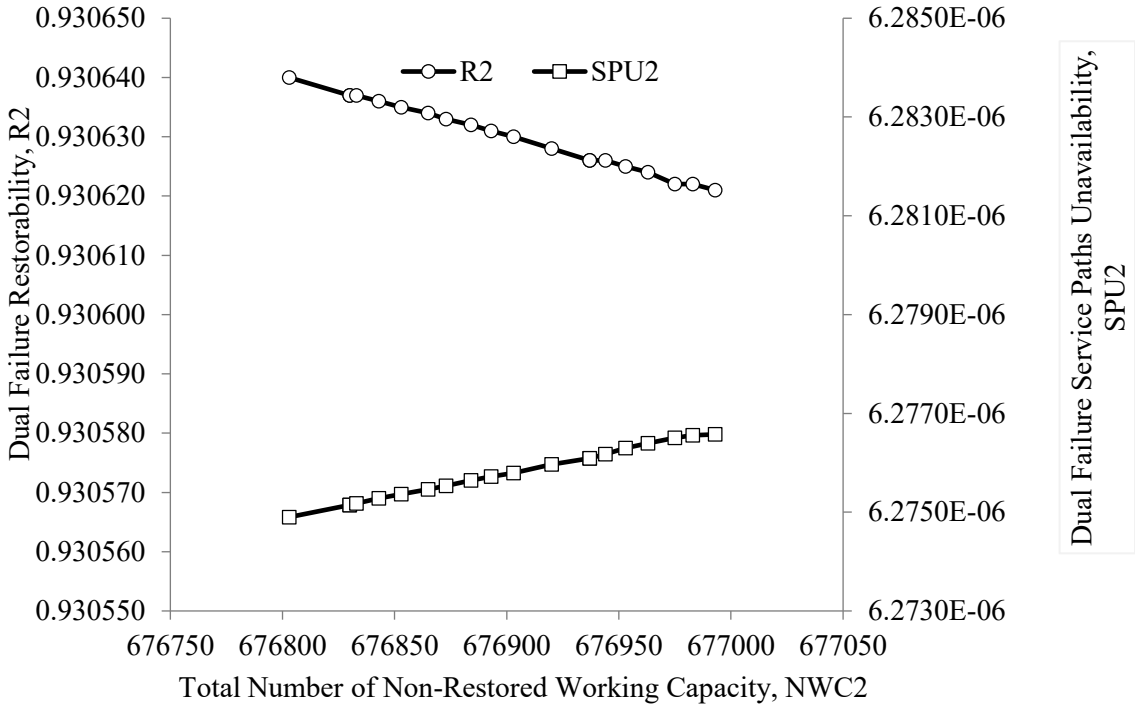


Figure 6.27 – Dual failure restorability and service path unavailability for 60-node 96-span network (zoom-in version)

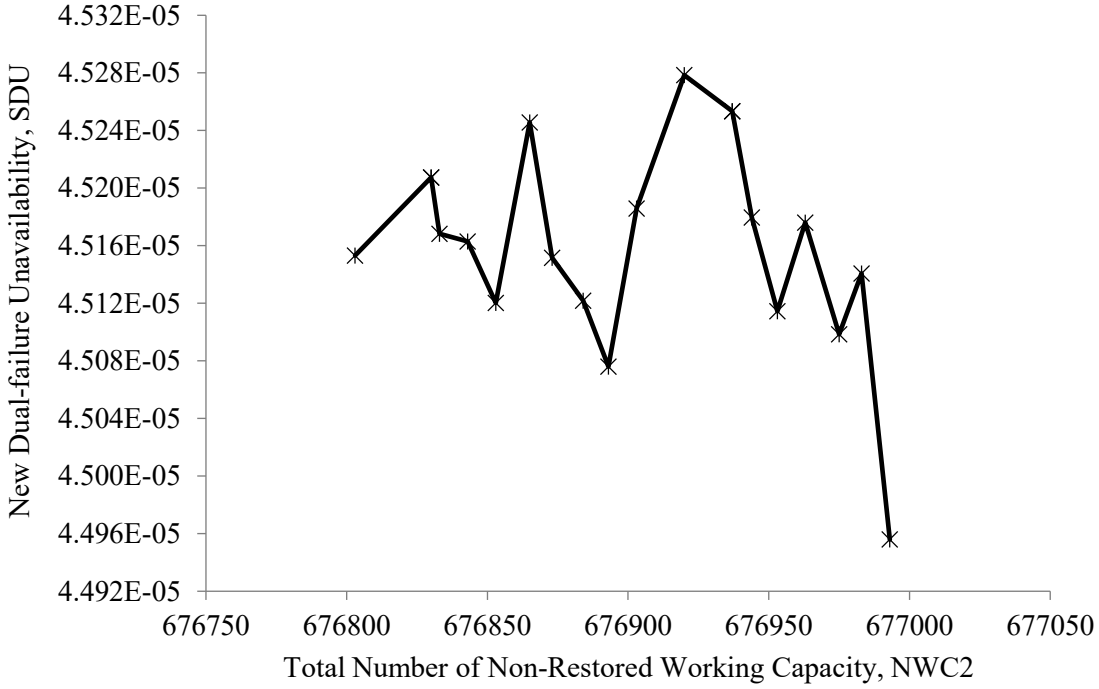


Figure 6.28 – New dual failure restorability for 60-node 96-span network (zoom-in version)

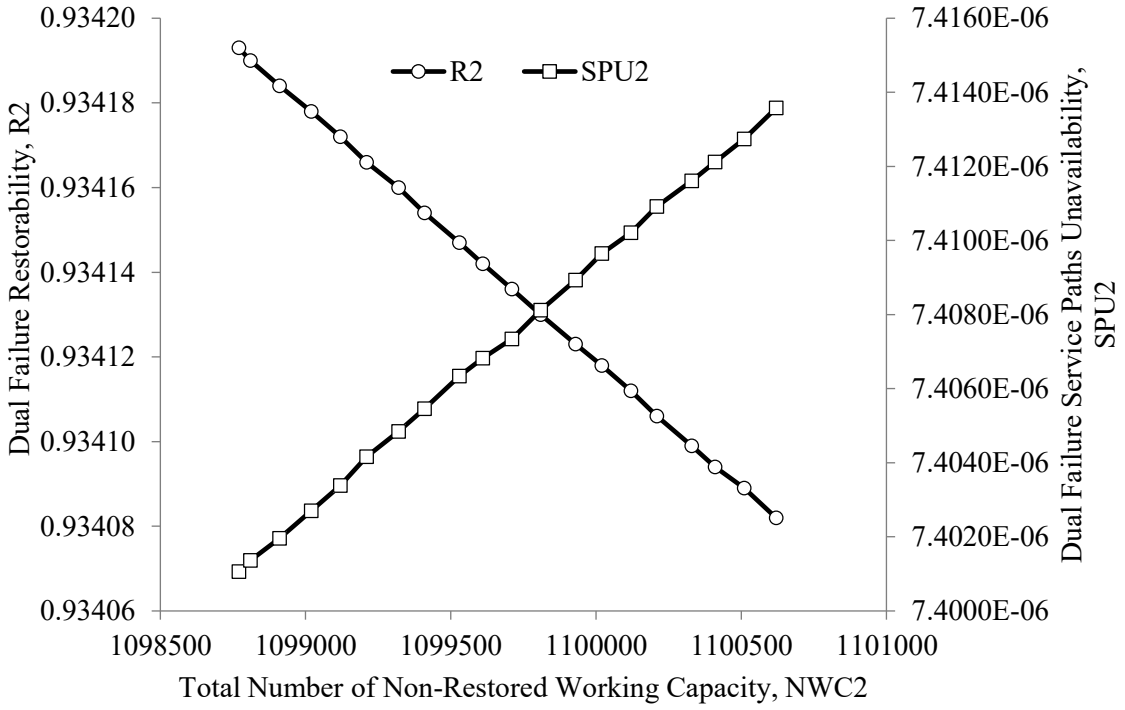


Figure 6.29 – Dual failure restorability and service path unavailability for 70-node 105-span network

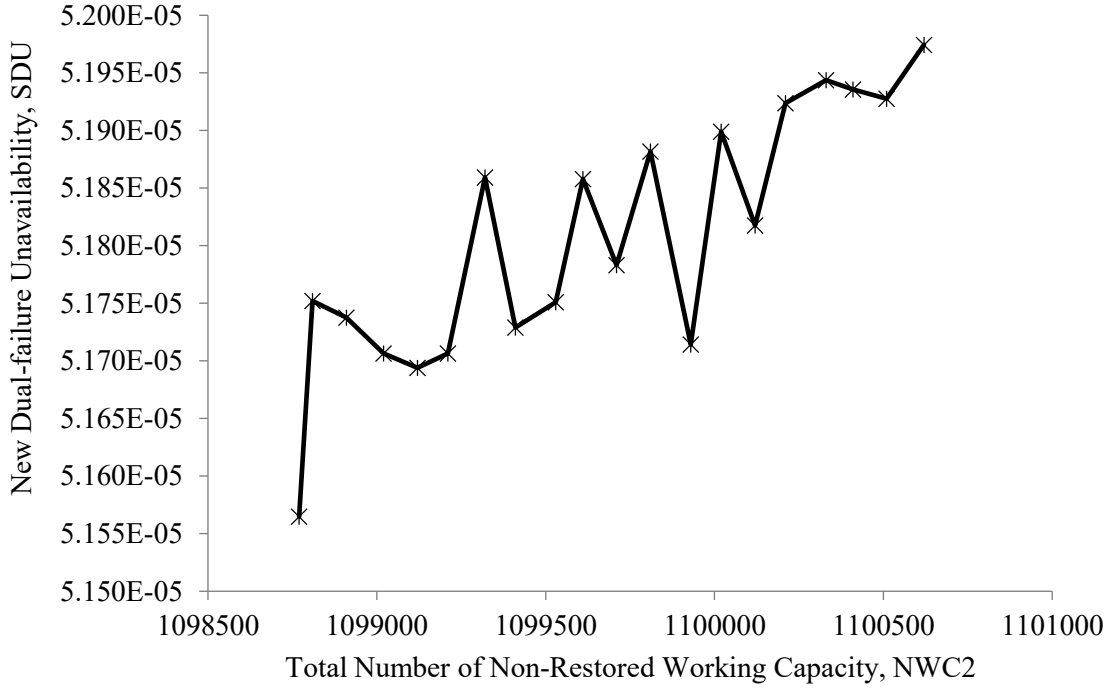


Figure 6.30 – New dual failure restorability for 70-node 105-span network

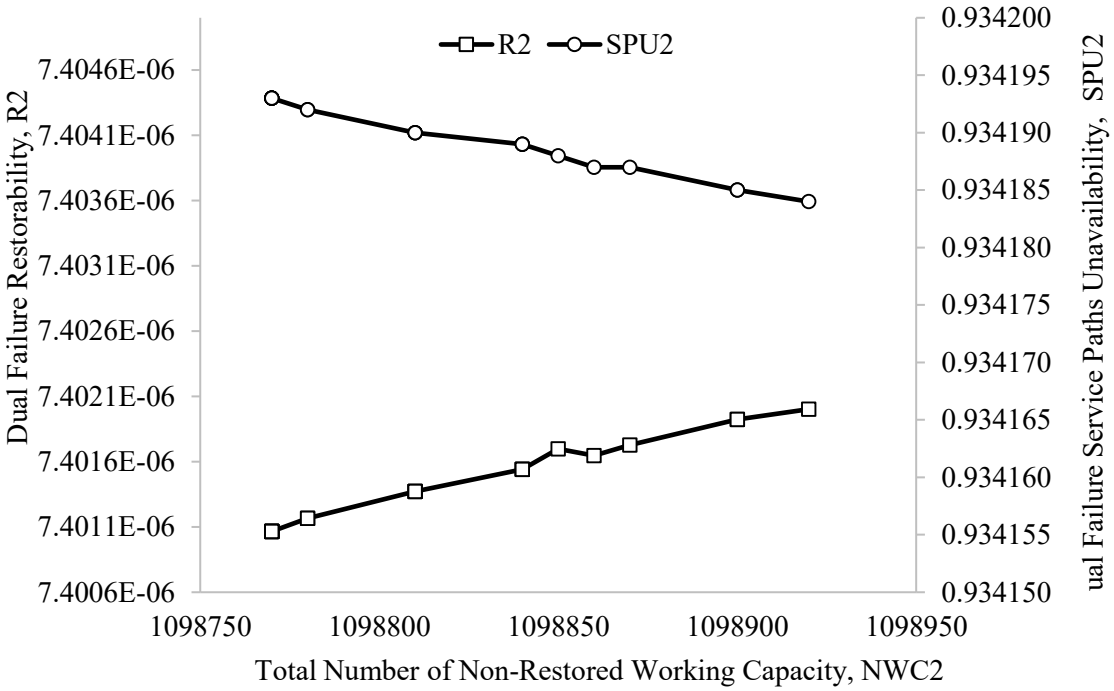
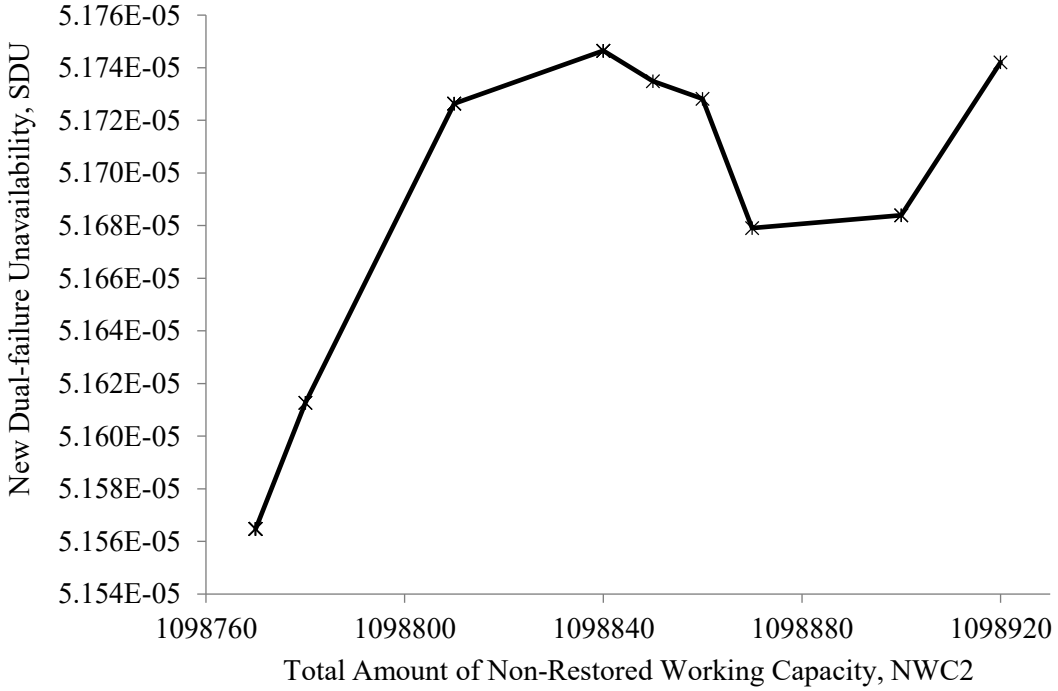


Figure 6.31 – Dual failure restorability and service path unavailability for 70-node 105-span network (zoom-in version)



(d)

Figure 6.32 – New dual failure restorability for 70-node 105-span network (zoom-in version)

In general, experimental results reveal similar patterns for corresponding methods in the experimental networks, suggesting that our findings are neither related to network topology nor size.

In detail, the plots regarding dual-failure restorability and dual-failure service path unavailability are nearly linear, one with positive slope (SPU_2) and the other with negative slope (R_2). That is to say, both SPU_2 and R_2 are approximately linearly correlated to NWC_2, but in opposite directions and with differing scale. While in terms of SDU, each curve shows a dramatic fluctuation as NWC_2 climbs linearly in its selected range. In other words, SDU does not have an obvious pattern towards changing NWC_2. On the other hand, SDU is smaller than its counterpart, SPU_2. This is reasonable because SDU is defined towards the smallest number of lost service paths, while NWC_2 takes into account each non-restored working capacity regardless of the failed spans' relationship.

6.7 CONCLUSION

Two overall approaches have been utilized to date for analyzing availability arising from dual-failure scenarios. The indirect method uses availability metrics (e.g., NWC_2, R_2 and SPU_2), and the direct approach explicitly calculates service path unavailability (SPU_2). Moreover, both SPU_2 and R_2 have been treated as though they are proportional to NWC_2, or at least strong metrics for it; it is generally assumed in the literature that minimizing SPU_2 is equivalent to maximizing R_2, and that both minimal SPU_2 and maximal R_2 can be achieved by minimizing NWC_2. Meanwhile, minimizing NWC_2 is the most common method for enhancing a network's service availability. This chapter investigated the correctness of those assumptions and current methods for determining service availability

and finds that this is not generally the case. SDU was proposed as a surrogate of current dual-failure unavailability to better assess network availability. To implement our evaluation method, an MNDF-ml framework was developed to investigate the relationships between the aforementioned metrics. Based on experiments conducted on four span-restorable mesh networks, we make the following conclusions.

(1) From theoretical analysis, NWC_2, R_2 and SPU_2 cannot directly lend themselves to evaluate dual-failure service availability. Conversely, the proposed SDU is a more accurate expression of dual-failure unavailability.

(2) Experiments on six test-case networks suggest that R_2 can be maximized by minimizing NWC_2 and SPU_2can be minimized through minimizing NWC_2, but there is no obvious relationship between SDU and NWC_2.

The advent of this new metric is helpful in both a theoretical and a practical manner. Theoretically, it helps to differentiate the concepts of total lost working capacity and total lost working path. In fact, these two are not necessarily to equal in value, which is as in most cases. And practically, we now have a more accurate metric to rely on when reporting network availability or unavailability.

CHAPTER 7 DESIGN AND AVAILABILITY ANALYSIS OF SHARED BACKUP PATH PROTECTION NETWORKS[3]

In this chapter, we look at the SBPP survivable mesh networks. We start from the capacity design of the network by investigating currently used ILP models, and then move on to network availability analysis for a network designed to be full single-failure restorable using SBPP mechanism.

7.1 TRADITIONAL SBPP ILP MODELS

7.1.1 NOTATION

We define sets, parameters, and variables in this section in the following:

Sets:

D is the set of all demands in the network.

S is the set of all spans in the network.

S_b is the set of spans on candidate backup route b.

P^r is the set of candidate primary routes for demand r.

P_i^r is the set of candidate primary routes that cross span i for demand r.

[3] W. Wang, J. Doucette, "Optimized Design and Availability Analysis of Large-Scale Shared Backup Path Protected Networks," Telecommunication Systems, accepted on 12th September 2017, available online at: https://doi.org/10.1007/s11235-017-0392-2.

B^r is the set of candidate backup routes for demand r.

B_k^r is the set of candidate backup routes that cross span k for demand r.

Parameters:

c_k is the cost of assigning one unit of working or backup capacity to span k.

d^r is the magnitude of demand r.

$\zeta_k^{p,r}$ is the working route vector, binary, taking on a value of 1 if candidate working route p of demand r crosses span k.

$\xi_k^{b,r}$ is the backup route vector, binary, taking on a value of 1 if candidate backup route b of demand r crosses span k.

$\xi_k^{b,p,r}$ is the backup route vector, taking on a value of 1 if candidate backup route b for candidate primary route p of demand r crosses span k, and 0 otherwise.

Variables:

w_k is the amount of working capacity assigned to span k.

s_k is the amount of backup capacity assigned to span k.

w_p^r is a binary variable in the single-flow design model, which equals 1 if the candidate primary route p of demand r is assigned as the actual working route and 0 otherwise; an integer variable in multi-flow design model, which represents the number of units allocated to candidate working route p of demand r.

$f_b^{p,r}$ is an integer variable representing the amount of units allocated to candidate

backup route b for candidate primary route p of demand r.

f_b^r is a binary variable indicating whether candidate backup route b is assigned as the actual backup route of demand r, with a value of 1 if assigned and 0 otherwise.

$\delta_b^{p,r}$ is a dummy binary variable denoting the multiplication of f_b^r and w_p^r, taking a value of 1 if and only if both f_b^r and w_p^r are equal to 1 and taking a value of 0 otherwise.

7.1.2 TRADITIONAL SBPP SINGLE-FLOW ILP MODEL

Traditional single-flow SBPP ILP models aim to minimize the total cost of allocating working and backup capacity for each span in the network while satisfying full single-failure restoration. Despite of the variety, they all follow common assumptions. First, the cost of assigning one unit of working capacity or backup capacity on each span is constant. Second, the demands are full mapping of nodes, i.e., there is always a demand between each pair of nodes in the network. Third, the failure of each span in the network is *independently and identically distributed* (i.i.d.). The ILP model for the single-flow SBPP can be formulated as follows [77].

The objective function is:

$$Cost = \sum_{k \in S} c_k (w_k + s_k) \tag{7-1}$$

The constraints include:

$$\sum_{p \in P^r} w_p^r = 1 \qquad \forall r \in D \tag{7-2}$$

$$\sum_{b \in B^r} f_b^r = 1 \qquad \forall r \in D \tag{7-3}$$

$$w_k = \sum_{r \in D} \sum_{p \in P_i^r} w_p^r \cdot d^r \qquad \forall k \in S \tag{7-4}$$

$$s_k \geq \sum_{r \in D} \sum_{p \in P_i^r} \sum_{b \in B_k^r} \delta_b^{p,r} \cdot d^r \qquad \forall i, k \in S | i \neq k \qquad (7\text{-}5)$$

$$\sum_{b \in B^r | i \in S_b} f_b^r \geq w_p^r \qquad \forall r \in D, p \in P_i^r, i \in S \qquad (7\text{-}6)$$

$$\delta_b^{p,r} \geq f_b^r + w_p^r - 1 \qquad \forall r \in D, p \in P^r, b \in B^r \qquad (7\text{-}7)$$

$$\sum_{p \in P^r} \sum_{b \in B^r} \delta_b^{p,r} = 1 \qquad \forall r \in D \qquad (7\text{-}8)$$

The constraint sets in (7-2) and (7-3) limit the number of working and backup routes for each demand. Constraints in (7-4) guarantee that the working capacity on each span k can accommodate the working flow on each primary route p of each demand r simultaneously. The constraint set (7-5) ensures that the amount of backup capacity on each span k can hold all the concurrently incurred backup flow crossing it under single failure i. The constraint set (7-6) ensures that the working route and its backup route are disjoint. To guarantee each constraint is linear, a dummy variable $\delta_b^{p,r}$ is exploited to represent the multiplication of f_b^r and w_p^r. Due to the binary property of f_b^r and w_p^r, only when them both take the value of 1, $\delta_b^{p,r}$ is equal to 1, and 0 otherwise. Constraints in (7-7) and (7-8) ensures this substitution equivalent such that $\delta_b^{p,r}$ has exactly the same function of $f_b^r \cdot w_p^r$.

7.1.3 TRADITIONAL SBPP MULTI-FLOW ILP MODEL

The formulation of the traditional multi-flow SBPP ILP design model can be found in [77]. This model adopts the general arc-path approaches whose major decisions are allocation of flow onto candidate working and backup routes. Additionally, the candidate working and backup routes are separated into different sets (as opposed to some approaches where they are selected from a common pool). The formulated SBPP ILP model determines

the final set of working and backup routes that optimize the objective function and all the constraints from all the input candidate working and backup routes. The benchmark model aims to minimize the total cost of allocating working and backup capacity onto each span, which is formulated as follows [77].

The objective is to minimize total cost:

Minimize:

$$Cost = \sum_{k \in S} c_k (w_k + s_k) \tag{7-9}$$

The constraints are:

$$\sum_{p \in P^r} w_p^r = d^r \qquad \forall r \in D \tag{7-10}$$

$$w_k = \sum_{r \in D} \sum_{p \in P^r} \zeta_k^{p,r} w_p^r \qquad \forall k \in S \tag{7-11}$$

$$\sum_{b \in B^r} f_b^{p,r} = w_p^r \qquad \forall r \in D, p \in P^r \tag{7-12}$$

$$s_k \geq \sum_{r \in D} \sum_{p \in P_i^r} \sum_{b \in B_k^r} \zeta_i^{p,r} \xi_k^{b,r} f_b^{p,r} \qquad \forall i, k \in S | i \neq k \tag{7-13}$$

$$\sum_{b \in B^r} \sum_{i \in S} \zeta_i^{p,r} \xi_i^{b,r} f_b^{p,r} = 0 \qquad \forall r \in D, p \in P^r \tag{7-14}$$

Constraints in (7-10) ensure enough working paths are assigned for each demand r. Constraints in (7-11) provide sufficient working capacity on each span to accommodate all the working paths that traverse it. The set of constraints in (7-12) assign sufficient backup paths for each primary route p of demand r. Constraints in (7-13) guarantee concurrent backup paths are fully restored. The disjointedness of working routes and relevant backup routes is enforced through the constraint set in (7-14).

Here, the model is formulated as a *joint capacity allocation* (JCA) model, where

working and backup routing is determined simultaneously. However, we can convert it to a *spare capacity allocation* (SCA) model where working is routed via shortest path or some other approach and only backup routing is optimized; this can be done by routing working paths via shortest paths, converting the associated w_k and w_p^r variables to input parameters, and removing constraints (7-10) and (7-11).

7.2 MOTIVATIONS AND GOALS

In the single-flow SBPP ILP design model, the solver selects the best pair of working and backup routes for each demand relation amongst a set of candidate working and backup routes, with consideration of minimal total allocation cost. To achieve this, there is a binary variable associated with each candidate working or backup route, representing whether or not it is assigned as the actual working or backup route in the optimized design. The number of binary variables increases substantially as network size increases, resulting in a rapid increase in complexity and solution time. As a result, the single-flow SBPP ILP model is not scalable.

In order to reduce solution time, the work in [77] showed that if we allow multiple working and backup routes (instead of just one of each), we can effectively do away with of all the binary variables associated with those variables. That work subsequently proposed an SBPP formulation where multiple working and backup routes are permitted for each demand pair in the network [77], which we will refer to as multi-flow *SBPP*. Each candidate working or backup route in this model has an *integer* variable associated with it, corresponding to the number of working or backup routes assigned to it, rather than a binary variable as with the traditional single-flow SBPP approach. This chapter focuses on this

multi-flow SBPP approach and improve on the associated ILP formulation.

The traditional single-flow model has been studied extensively. With the introduction of the multi-flow model in, we feel it is now time to investigate the availability implications of that model. Furthermore, since the multi-flow model is still quite new, there is an opportunity investigate its approach for improvement. Therefore, we will seek to address the following two goals:

(1) We will propose a novel multi-flow model to enhance the performance of the model relative to the benchmark and formulate a new multi-flow SBPP ILP model that is faster to solve and convenient to operate.

(2) We will develop an algorithm to analyze the availability performance of the multi-flow design, and compare its performance to the benchmark.

7.3 NEW MULTI-FLOW SBPP ILP DESIGN MODEL

7.3.1 NOTATION

New set used in this new model is $B^{p,r}$, which is the set of available backup routes for candidate primary route p of demand r. Other symbols used in this new model have been defined already in Section 7.1.1.

7.3.2 ILP FORMULATION

As previously discussed, the fundamental difference between the traditional multi-flow SBPP model and the single-flow SBPP model is with regards to the nature of the variables characterizing the working and backup routing (they are binary in the single-flow

model, and integer in the multi-flow model). We can regard the traditional multi-flow ILP model as an integralization of the traditional single-flow ILP model. However, this leaves us an opportunity for further improvement in other aspects; they both separate candidate working routes and backup routes into distinct input sets separated only according to demand [77].

In our new ILP model, we further split the candidate backup routes into subsets according to the working routes they aim to protect. By doing so, the constraints for enforcing disjointedness of working routing and relevant backup routes (i.e., the constraints in Eq. (7-14)) become redundant. In addition, the sets P_i^r and B_k^r become redundant as well. Accordingly, both the input data and constraints required to formulate the model become smaller. The complete formulation of our new model is as follows.

The objective function is identical to that in the benchmark model, as shown in Eq. (7-15):

Minimize:

$$Cost = \sum_{k \in S} c_k (w_k + s_k) \tag{7-15}$$

The constraints are simplified as follows:

$$\sum_{p \in P^r} w_p^r = d^r \qquad \forall r \in D \tag{7-16}$$

$$w_k = \sum_{r \in D} \sum_{p \in P^r} \zeta_k^{p,r} w_p^r \qquad \forall k \in S \tag{7-17}$$

$$\sum_{b \in B^{p,r}} f_b^{p,r} = w_p^r \qquad \forall r \in D, p \in P^r \tag{7-18}$$

$$s_k \geq \sum_{r \in D} \sum_{p \in P^r} \sum_{b \in B^{p,r}} \zeta_i^{p,r} \xi_k^{b,p,} f_b^{p,r} \qquad \forall i, k \in S | i \neq k \tag{7-19}$$

The constraint sets in (7-16) and (7-17) are identical to those in (7-10) and (7-11), respectively. The constraints in (7-18) are similar to those in (7-12), except that the set of backup route is now specified for each potential working route p of demand r. Here, we specify the relationship between working routes and their backup routes explicitly in the data pre-processing procedure. Constraints in (7-19) are equivalent to those in (7-13), but they substitute the sets $p \in P^r$ and $b \in B^{p,r}$ for $p \in P_i^r$ and $b \in B_k^r$. The changes in the model not only simplify the formulation, but also enhance the solution process of the model. The new model is simplified by reducing usage of notation, which makes the new formulation more concise in format. The scale of the data file is greatly reduced in two ways. First, without the sets of P_i^r and B_k^r, it is not necessary to specify the set of candidate working routes that traverse span i for demand r or the set of candidate backup routes that cross span k for demand r. In fact, the sets of P_i^r and B_k^r are subsets of P^r and B^r, respectively, for $\forall i, j \in S$, so it is unnecessary to have both the sets and its subsets.

As stated, the major difference between the new model and the benchmark model is that the relation between a specified working route and its backup routes are explicitly specified in the new model. In practice, this is easily achieved in pre-processing by temporarily removing a working route from the network topology when enumerating its associated candidate backup routes. A by-product of this separation of candidate backup routes according to the working route they aim to protect is that disjointedness of working route and its backup routes is satisfied automatically.

7.4 NETWORK AVAILABILITY ANALYSIS

We now propose an algorithm to analyze the dual-failure availability for an SBPP

multi-flow network designed for full single-failure restoration. In developing this algorithm, we make the following assumptions:

(1) Restoration rule: We restore failed working routes, but we do not restore the failed backup routes; once a backup route is failed, it remains failed.

(2) Earmark rule: The surviving portions of backup capacity on a failed backup route is released and can be used for restoration of other failed working routes (e.g., we use stub-release [24]). On the contrary, the surviving portions of a failed working route are not reused for restoring other failed working routes.

(3) Predefine rule: Upon failures of a span, only the predefined backup routes are adopted and no new backup routes are sought. More specifically, the exact set of backup routes and the exact amount of backup flow on them, which are predefined from the single-failure design process, are adopted to restore the failed working routes regardless of the failure scenario (single or dual).

(4) Maximum rule: Because of full restoration of single failures in the design, the first failure in a dual-failure scenario is fully restored. Nevertheless, it is necessary to examine whether sufficient backup flow is available for the second failure in that dual-failure scenario. We restore the second failure by making the best of available backup capacity.

(5) Priority rule: we do not take into account the priority of each demand; rather, we treat them equally. This can be modified if the priority of demands is provided.

(6) Working path oriented rule: Unlike in the single-flow model, a backup route is specified for an individual working route instead of a specific demand. Hence, it is more

reasonable to focus on each working route rather than each demand for the purpose of analyzing dual-failure availability.

The algorithm is illustrated in Figure 7.1 based on the above-mentioned six assumptions and rules. Step 1 calculates $R_2(i,j)$ for $\forall i,j \in S | i \neq j$, Step 2 computes availability $A_2(p)$ for $\forall p \in P$, and Step 3 calculates network overall dual-failure availability A_2. In the first step, the outer loop iterates over each dual-failure scenario. Within each loop, i.e., for a given dual-failure scenario (i,j), we first obtain the design data from design model, including span backup capacity s_k, the working routes P^r, the working flow w_p^r, the backup routes $B^{p,r}$, and the backup flow $f_b^{p,r}$. We initialize the values of wf_{aff} and wf_{lost} to be zero. Next, we utilize the restoration scheme of dual failure (i,j), which is composed of two sub-steps, i.e., Major Sub-Step 1 and Major Sub-Step 2 as shown in the figure. Major Sub-Step 1 deals with the first failure i, and cycles through each working route, p, that is affected by failure j (the set of such working routes is denoted by WR_i) until all the affected working routes are considered. Within each cycle, the working flow on p (i.e., w_p^r) is added up to the value of wf_{aff}, p is restored by its pre-defined backup route(s), and the value of s_k on each backup route b of p is updated by removing the used backup capacity for restoration of failure i. Major Sub-Step 2 attempts to restore the second failure j. It examines each working route, p, that is affected by failure j but not by failure i (the set of such working routes is denoted by WR_j and satisfies $WR_i \cap WR_j = \emptyset$). For a given p, the value of wf_{aff} is updated by adding up the working flow of p, and the available backup flow on each of its backup routes is recalculated due to the update of s_k. To do so, we iterate through each ($b \in B^{p,r}$) and update its flow with the $\min(s_k)$ where $k \in b$. This is because the largest flow that backup route b can accommodate depends on the spans

with the smallest available spare capacity. If the sum of the backup flows on all its backup routes is larger than its working flow (i.e., $\sum_{b\in B^{p,r}} f_b^{p,r} \geq w_p^r$), p can be fully restored and the value of wf_{lost} remains unchanged. Otherwise, the value of wf_{lost} is increased by $(w_p^r - \sum_{b\in B^{p,r}} f_b^{p,r})$. Once each $p \in WR_j$ is considered, we calculate $R_2(i,j)$ through (4-9). Step 1 will be concluded when each dual failure (i,j) is cycled through. We have now obtained the values of $R_2(i,j)$ for all dual-failure scenarios, and then in step 2 we calculate the dual-failure availability for all the working routes (i.e., $\forall p \in P$), based on the calculation method described in Eq. (4-10). Finally, in step 3, we calculate the dual-failure availability for the entire network via the calculation method described in Eq. (4-11).

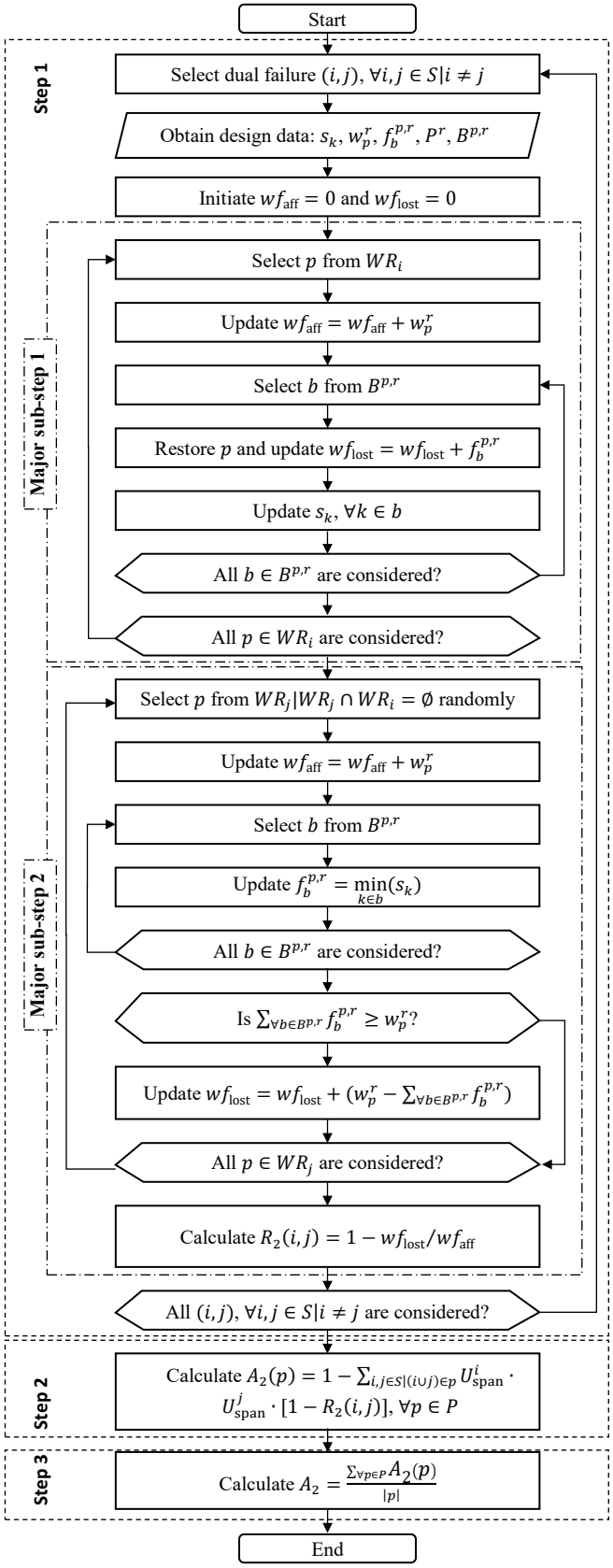


Figure 7.1 – Algorithm for calculating dual-failure availability

7.5 EXPERIMENTS AND DISCUSSION

7.5.1 NETWORK DESIGN RESULTS

The total network design costs of all networks in a family are normalized as per the equation in (7-20):

$$Cost_{\text{norm}} = \frac{Cost_{\text{actu}}}{Cost_{\text{min}}} \tag{7-20}$$

Here, $Cost_{\text{actu}}$ is the actual total cost of assigning working and spare capacity to a network, $Cost_{\text{min}}$ is the smallest actual total cost of the networks within its network family, and $Cost_{\text{norm}}$ is the normalized total cost. This effectively scales all of the network design costs such that the member of a family with the lowest-cost design is deemed to have a cost of 1.0. The normalized total costs of the 15 network families for the new multi-flow model and benchmark multi-flow model are shown in Figure 7.2 through Figure 7.16***Figure 7.17***. Each panel of the figure corresponds to a single network family, where each data point represents the normalized total working and spare capacity cost of the member of the family with the average nodal degree indicated on the x-axis for the benchmark multi-flow SBPP model and the new multi-flow SBPP model. We note that the normalized capacity costs for both multi-flow models have a similar trend as network connectivity increases, i.e., they both decrease rapidly with increasing network average nodal degree. This is generally due to an increased ability to use shorter and more efficient working and backup routes [18].

Figure 7.17 through Figure 7.31 show the total runtime values of the experimental networks for both multi-flow models. Each panel of the figure corresponds to a single

network family, where each data point represents the runtime (in seconds) of the member of the family with the average nodal degree indicated on the x-axis for the benchmark multi-flow SBPP model and the new multi-flow SBPP model, respectively. As with costs above, the runtime values decrease with network connectivity as well. As shown in the figure, runtime values of the new model are well below that of the benchmark model. On average, the new model is 51% faster approximately in comparison with the benchmark model. This value may seem to be a small improvement for a single model, however, the total time savings are substantial in terms of large number of models and repetitive experiments that might be typical of a comprehensive network design process. Furthermore, the runtime improvements ratio of the new model increases with the scale of the network with respect to the number of nodes, as shown in Figure 7.32. The x-axis is the network scale ranging from 10-node network family to 150-node network family, and the y-axis is the average runtime improvement ratio for the network family size (i.e., number of nodes) indicated on the x-axis.

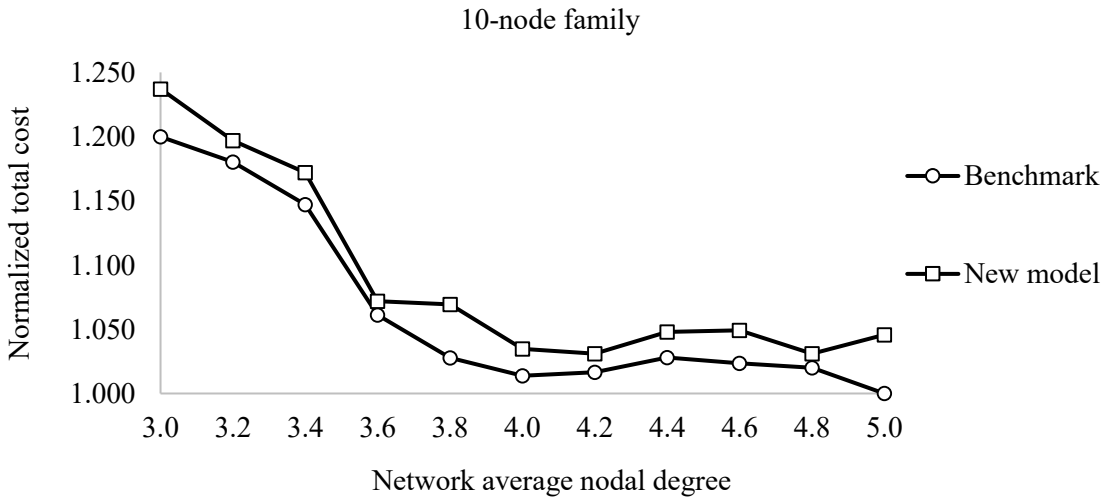


Figure 7.2 – Normalized cost of the 10-node network family

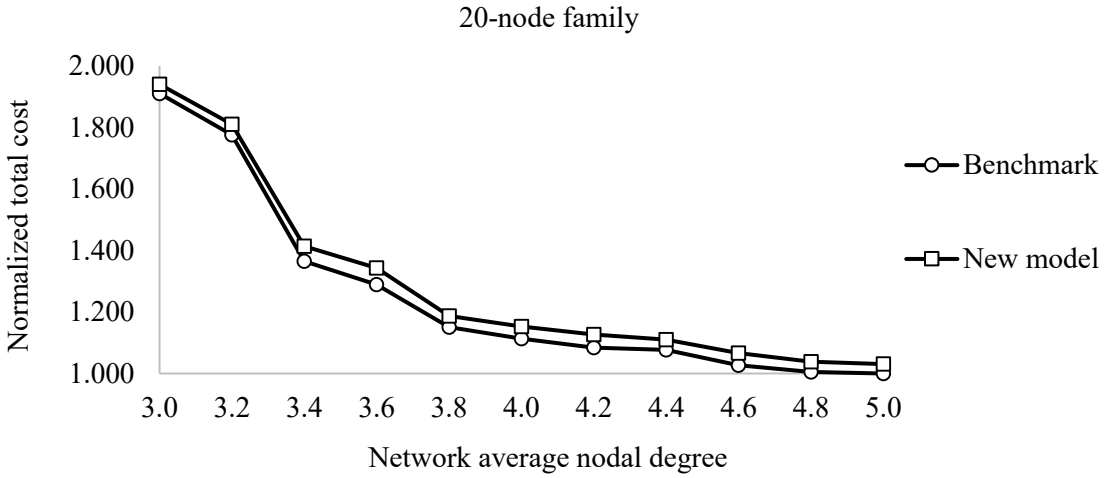


Figure 7.3 – Normalized cost of the 20-node network family

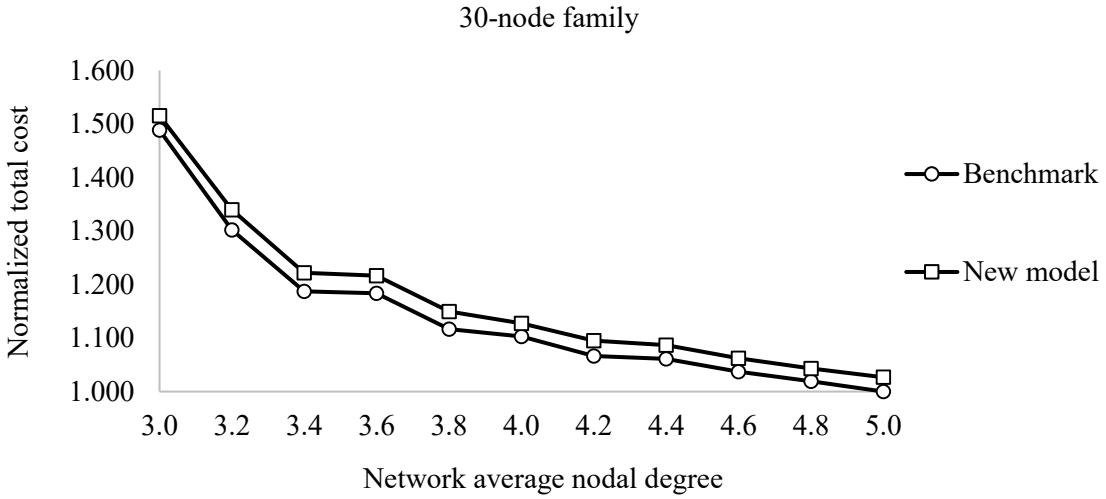


Figure 7.4 – Normalized cost of the 30-node network family

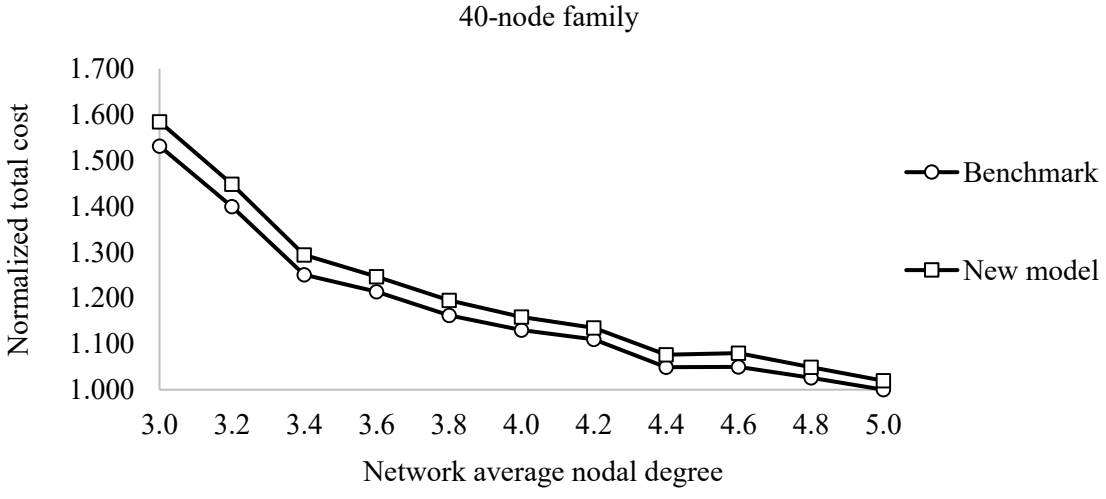


Figure 7.5 – Normalized cost of the 40-node network family

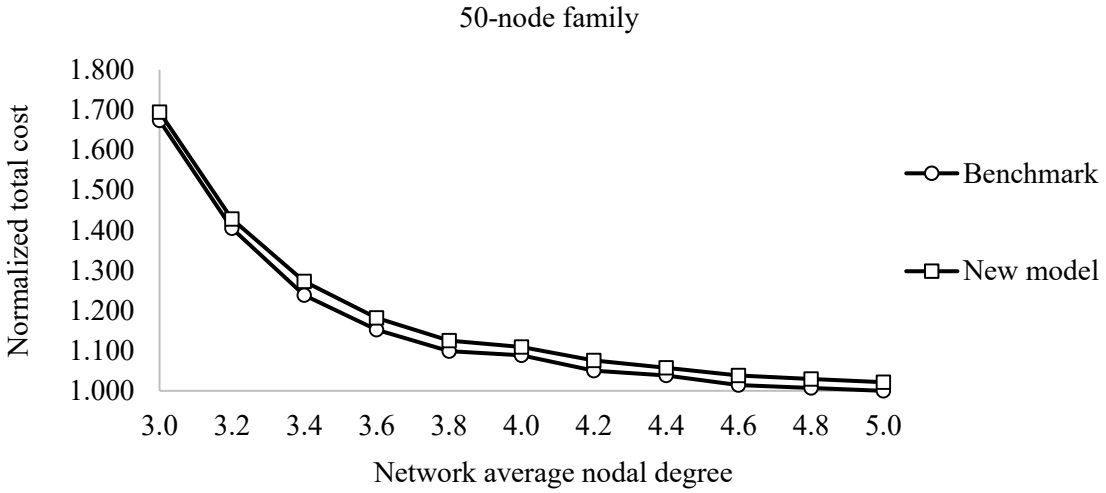


Figure 7.6 – Normalized cost of the 50-node network family

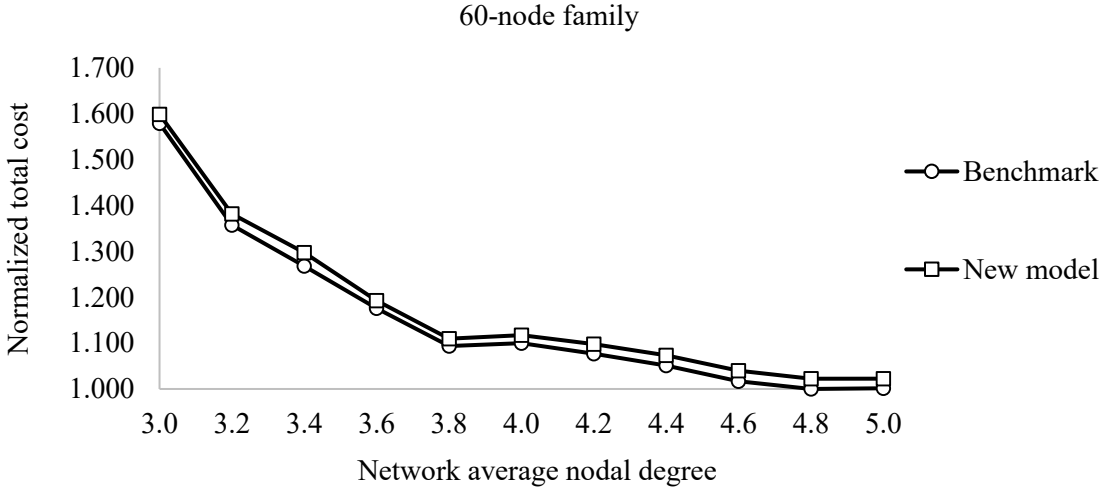


Figure 7.7 – Normalized cost of the 60-node network family

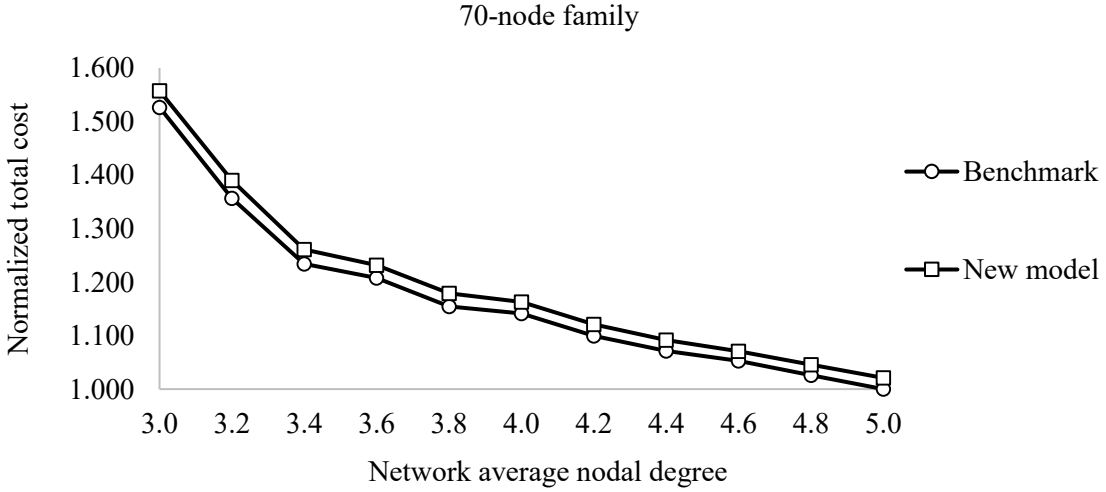


Figure 7.8 – Normalized cost of the 70-node network family

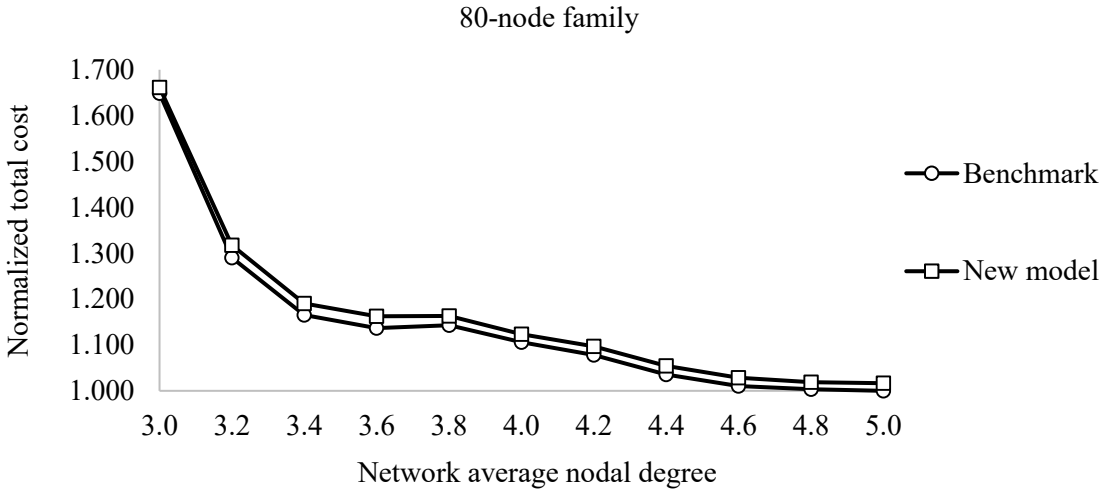


Figure 7.9 – Normalized cost of the 80-node network family

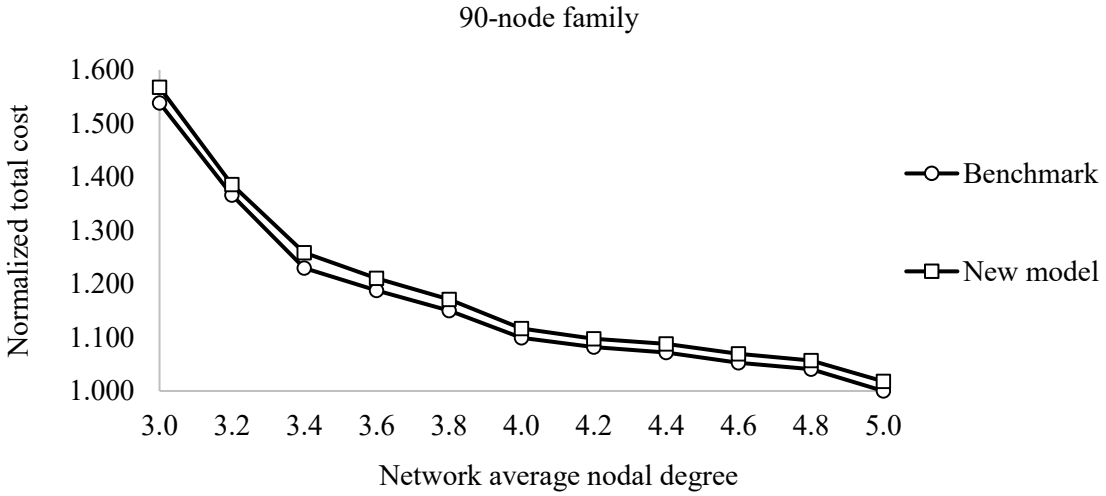


Figure 7.10 – Normalized cost of the 90-node network family

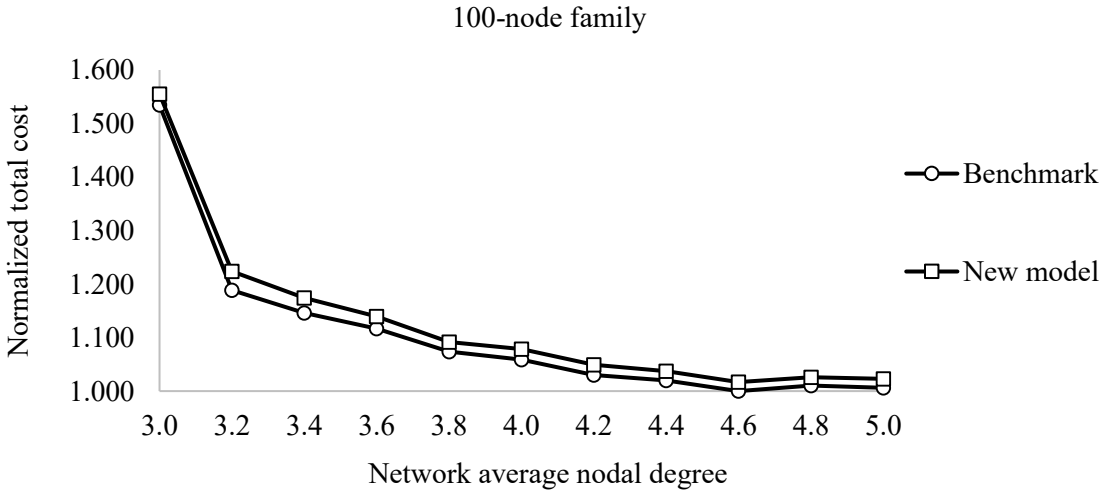


Figure 7.11 – Normalized cost of the 100-node network family

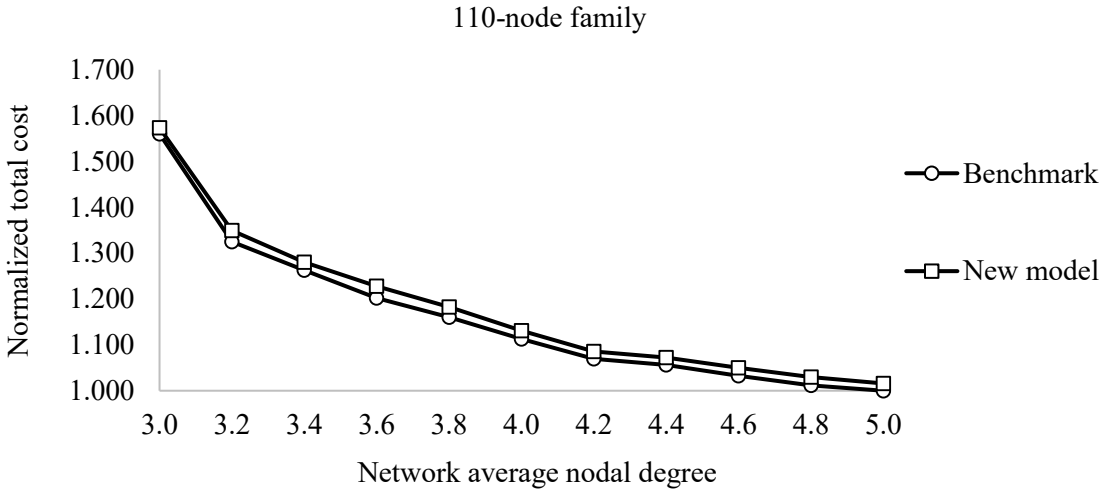


Figure 7.12 – Normalized cost of the 110-node network family

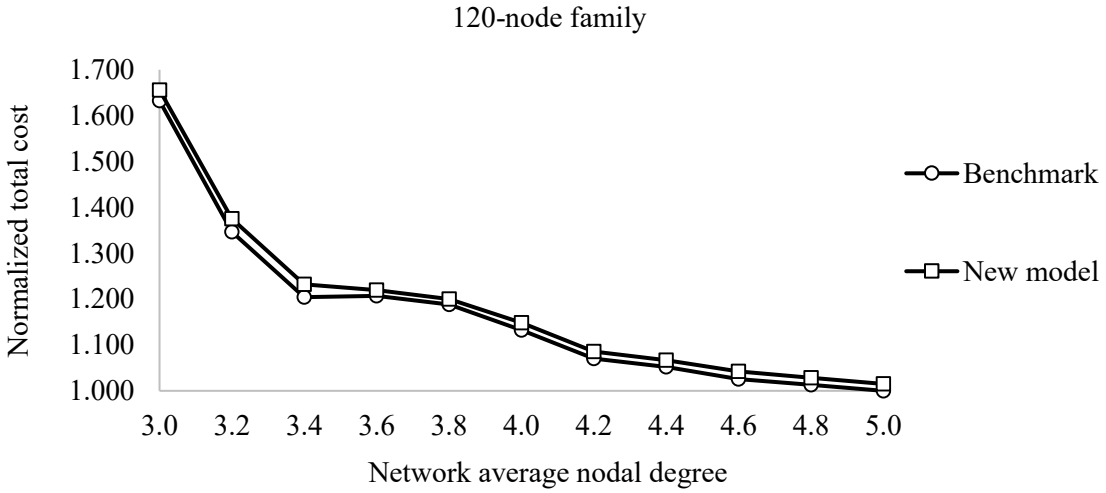


Figure 7.13 – Normalized cost of the 120-node network family

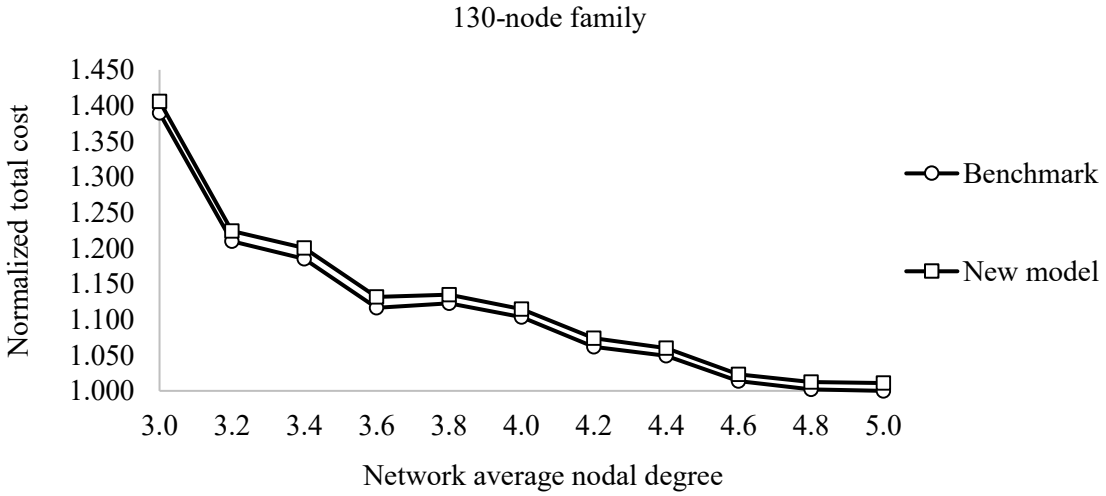


Figure 7.14 – Normalized cost of the 130-node network family

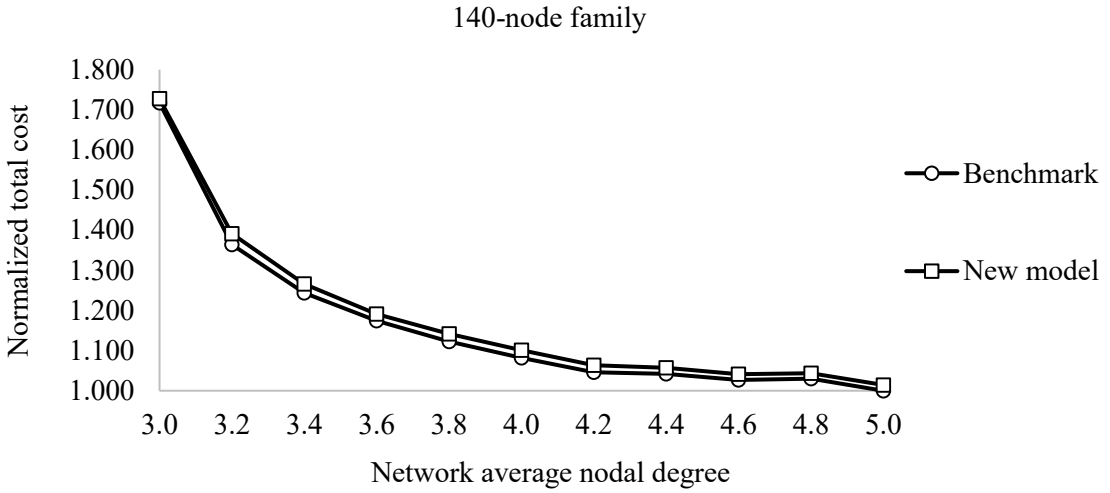


Figure 7.15 – Normalized cost of the 140-node network family

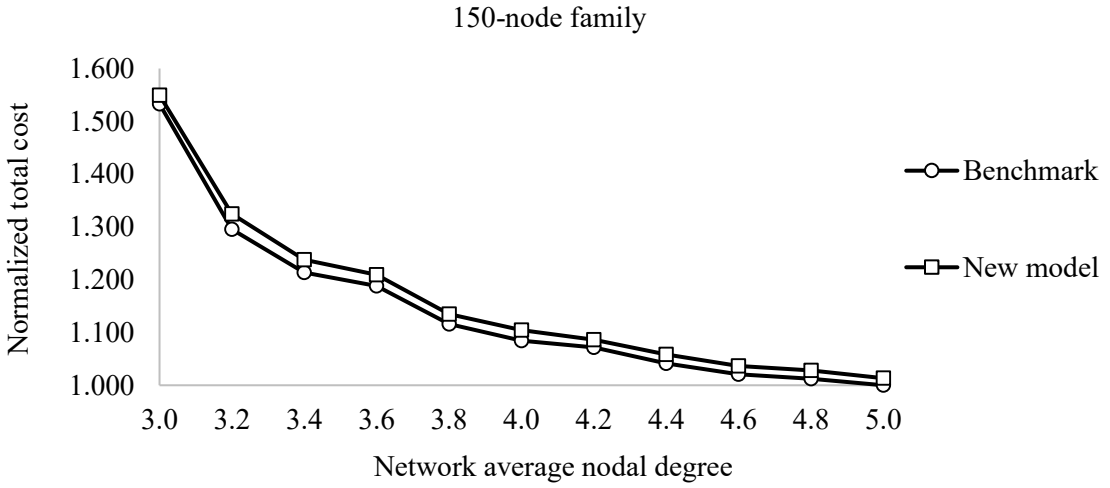


Figure 7.16 – Normalized cost of the 150-node network family

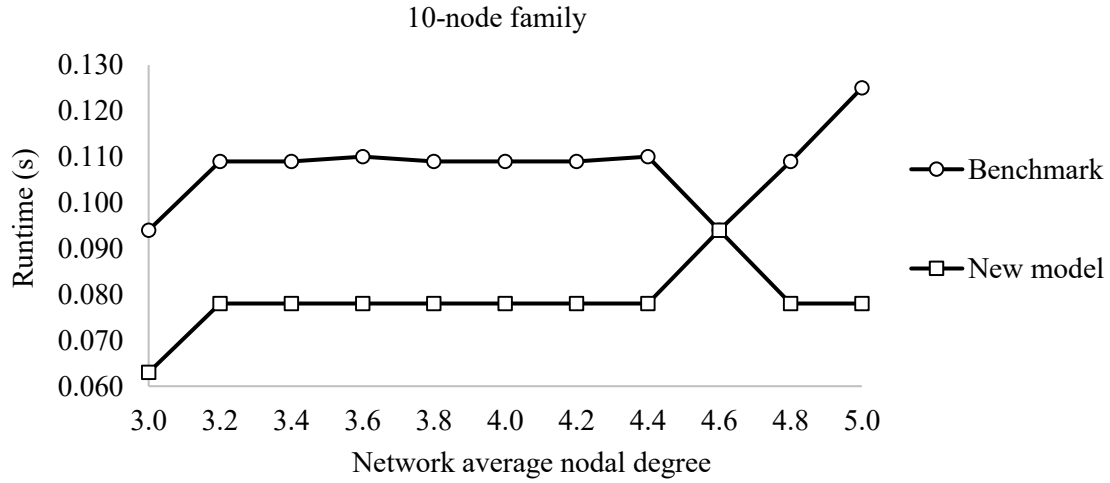


Figure 7.17 – Runtime of the 10-node network family

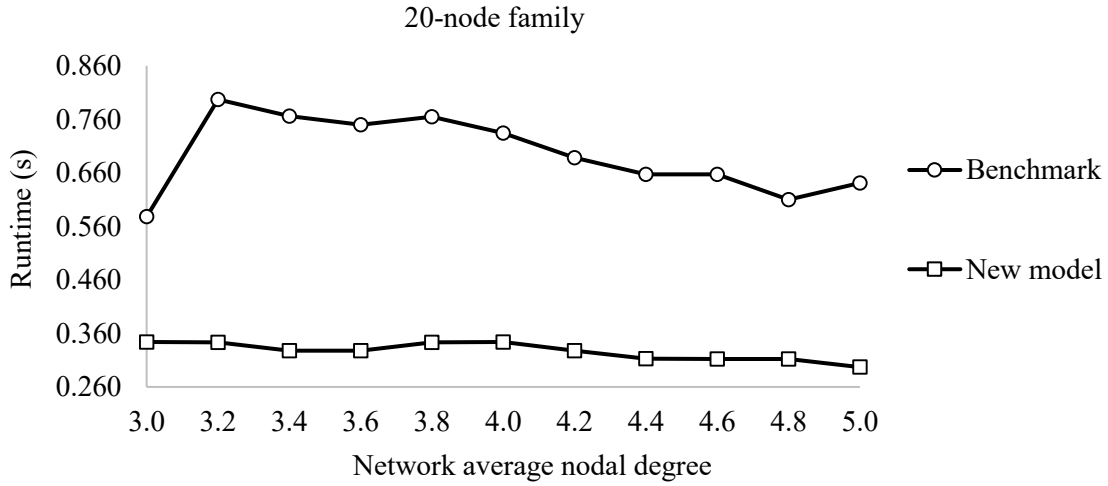


Figure 7.18 – Runtime of the 20-node network family

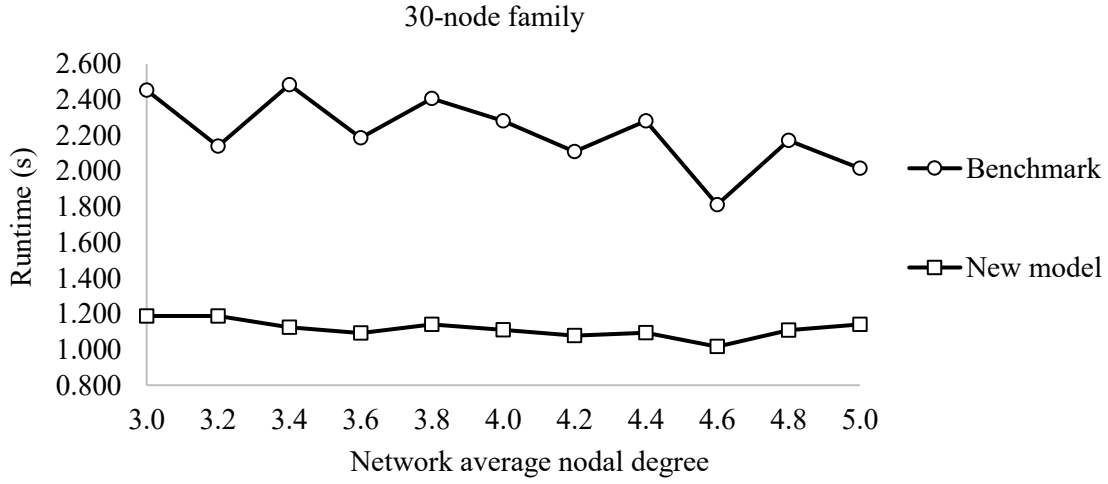


Figure 7.19 – Runtime of the 30-node network family

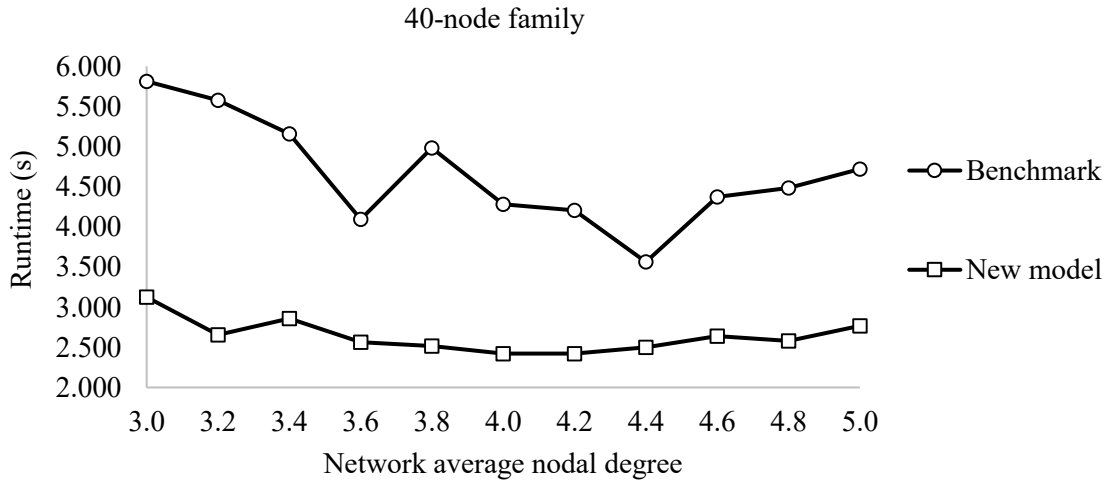


Figure 7.20 – Runtime of the 40-node network family

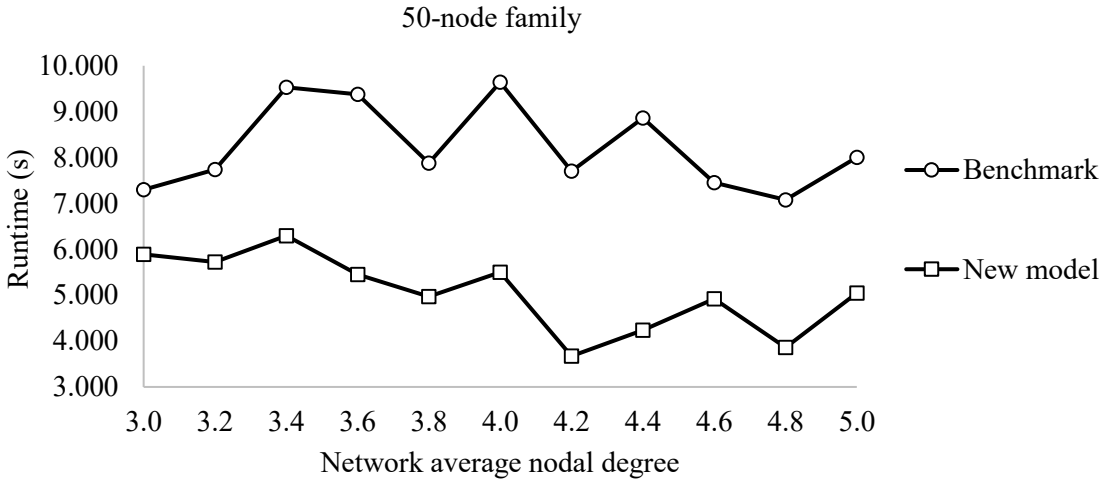


Figure 7.21 – Runtime of the 50-node network family

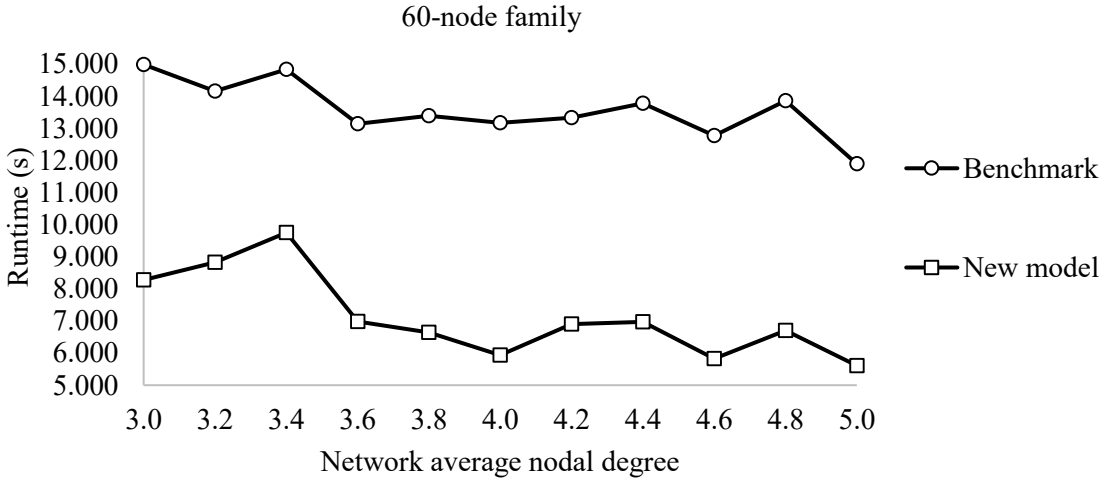


Figure 7.22 – Runtime of the 60-node network family

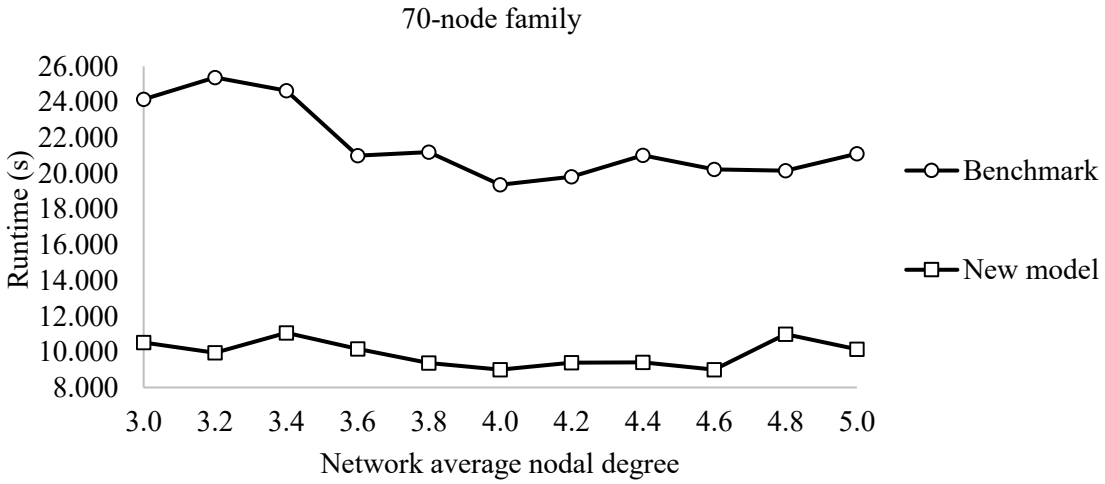


Figure 7.23 – Runtime of the 70-node network family

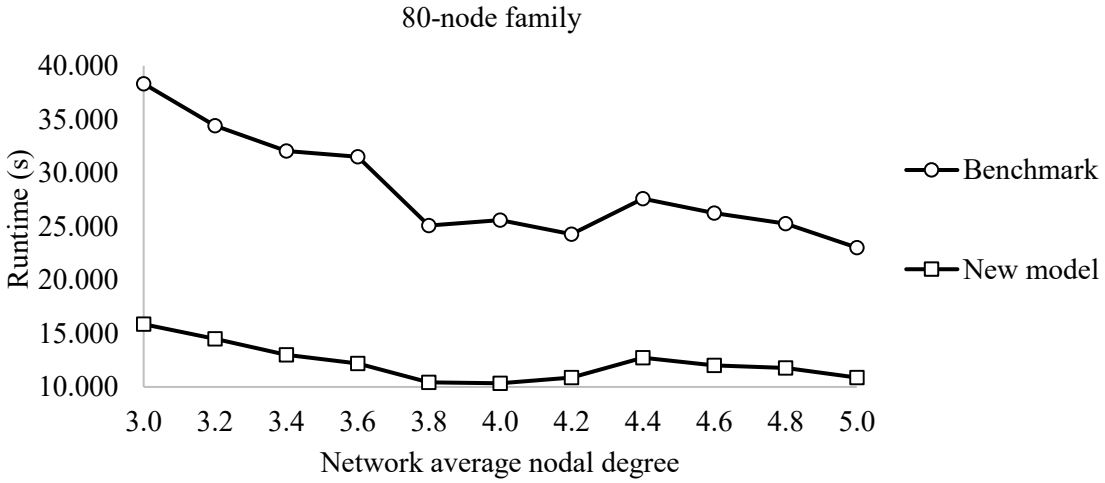


Figure 7.24 – Runtime of the 80-node network family

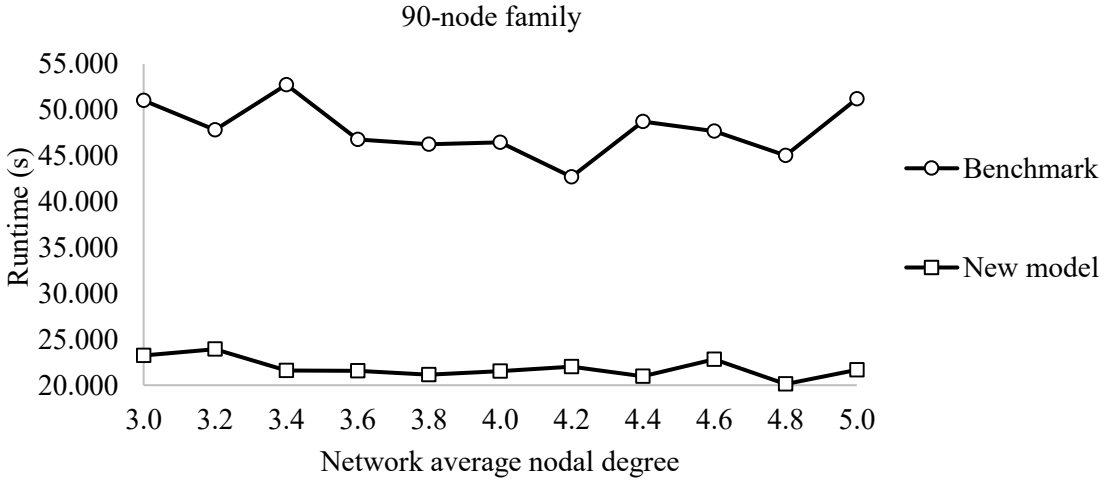


Figure 7.25 – Runtime of the 90-node network family

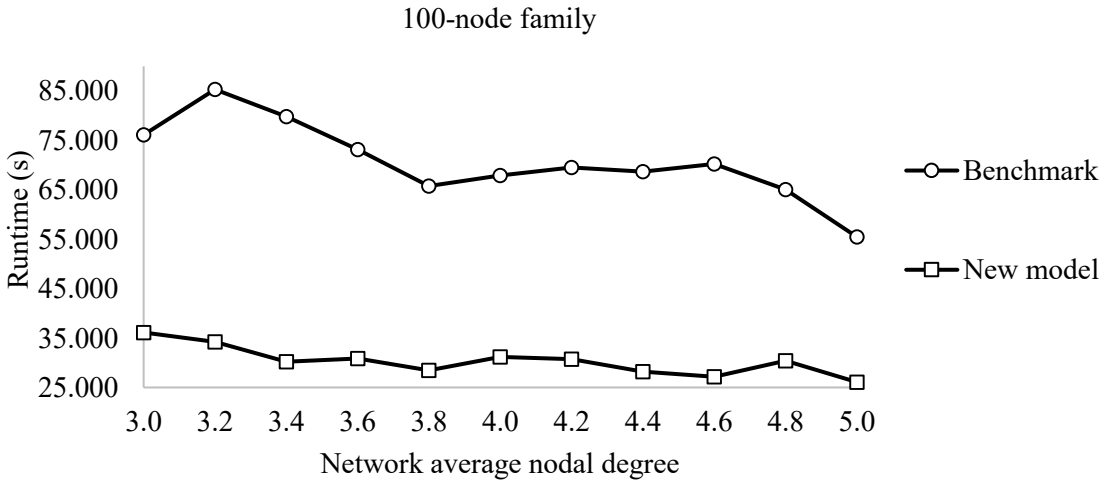


Figure 7.26 – Runtime of the 100-node network family

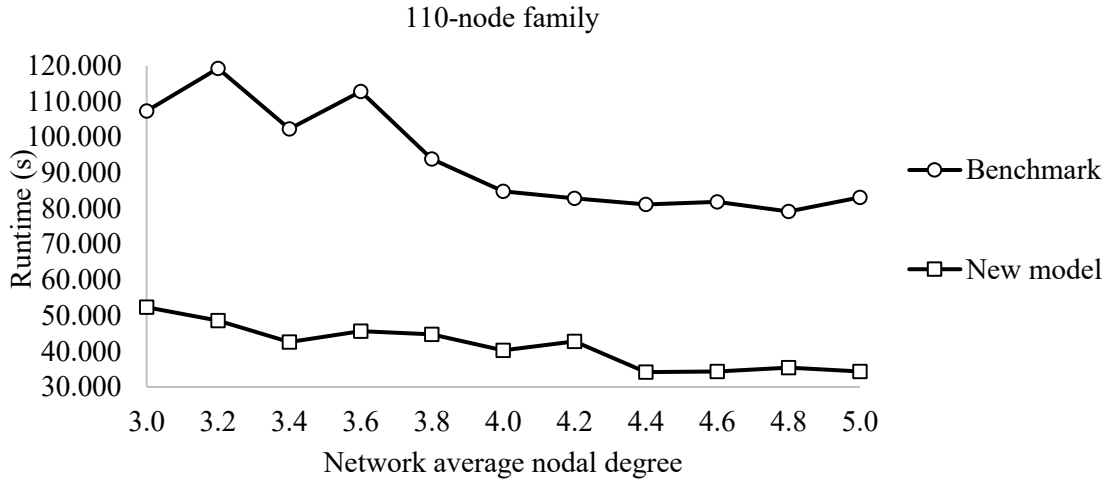


Figure 7.27 – Runtime of the 110-node network family

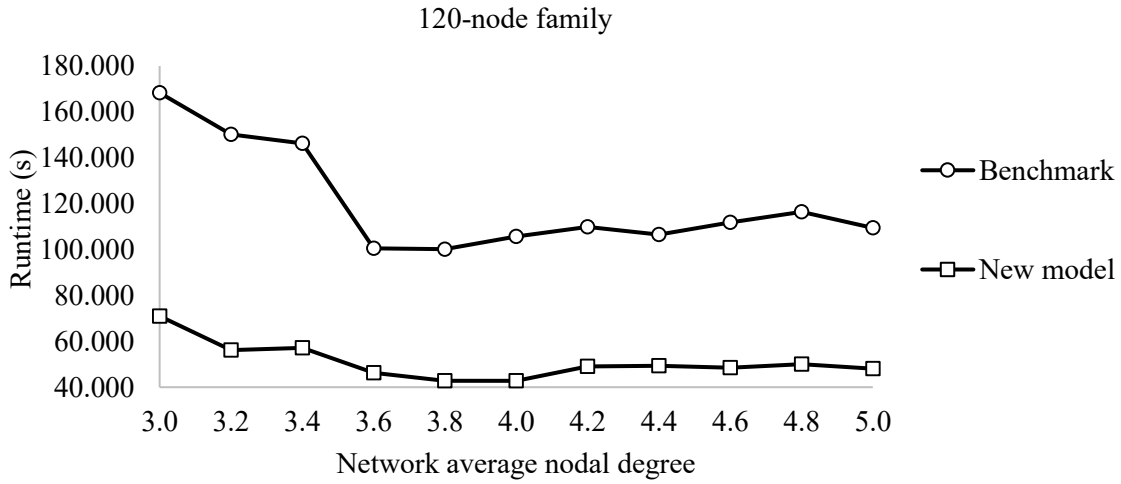


Figure 7.28 – Runtime of the 120-node network family

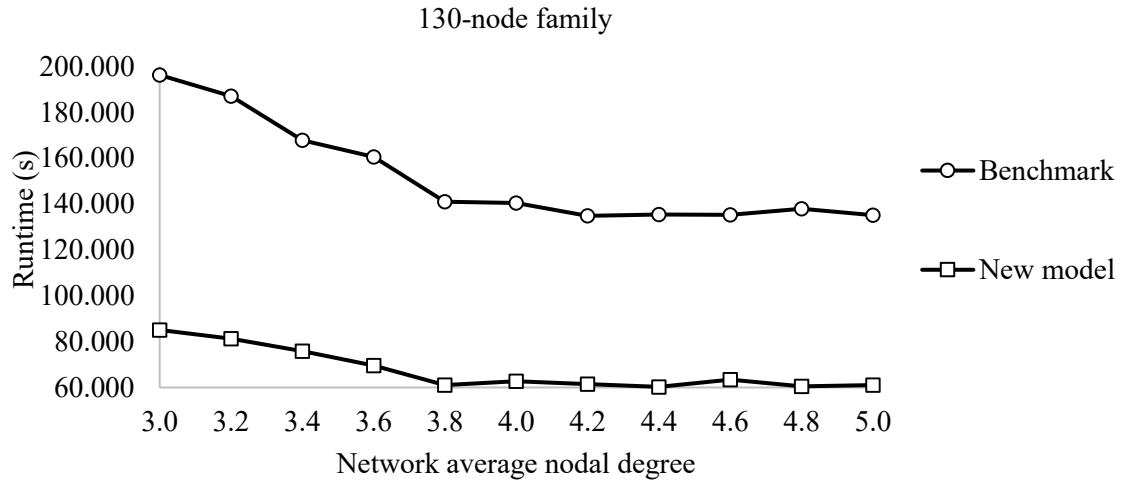


Figure 7.29 – Runtime of the 130-node network family

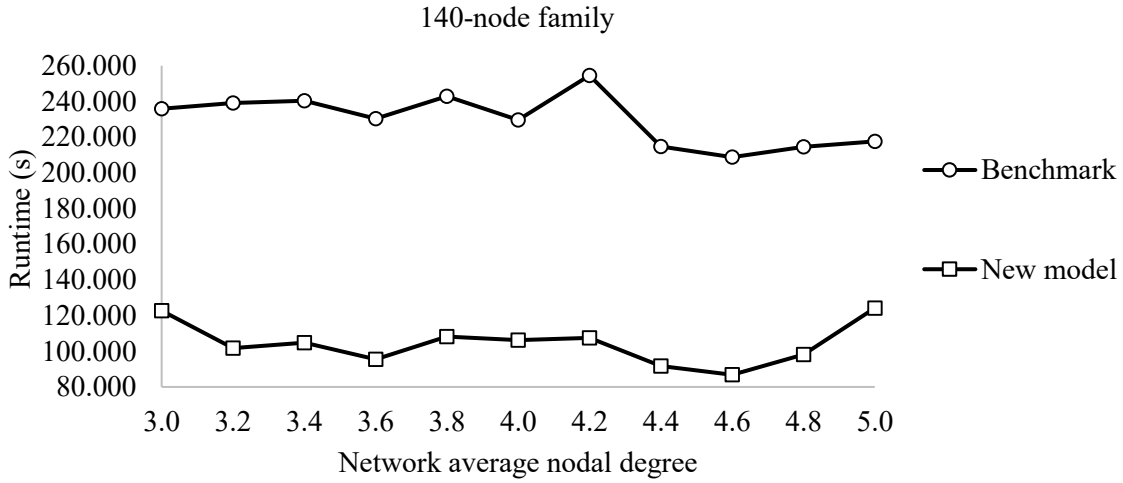


Figure 7.30 – Runtime of the 140-node network family

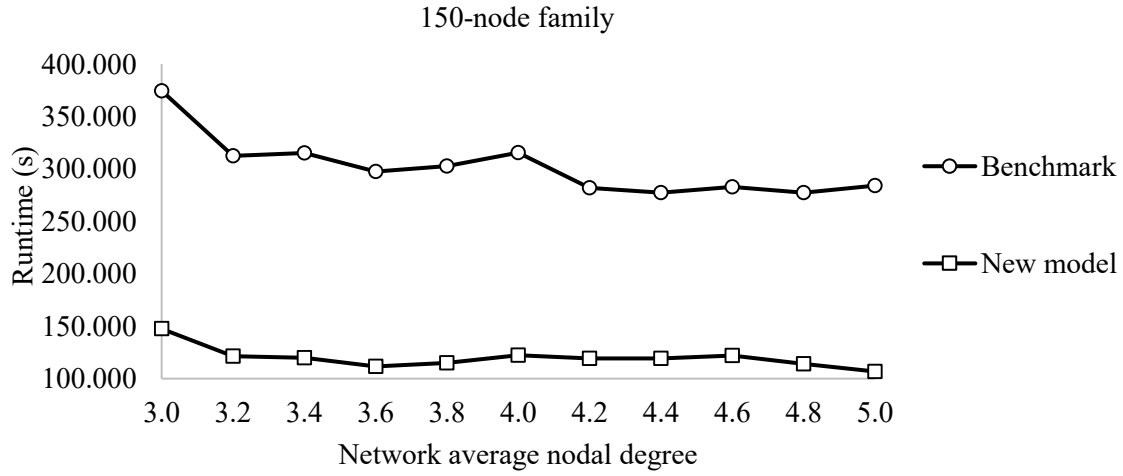


Figure 7.31 – Runtime of the 150-node network family

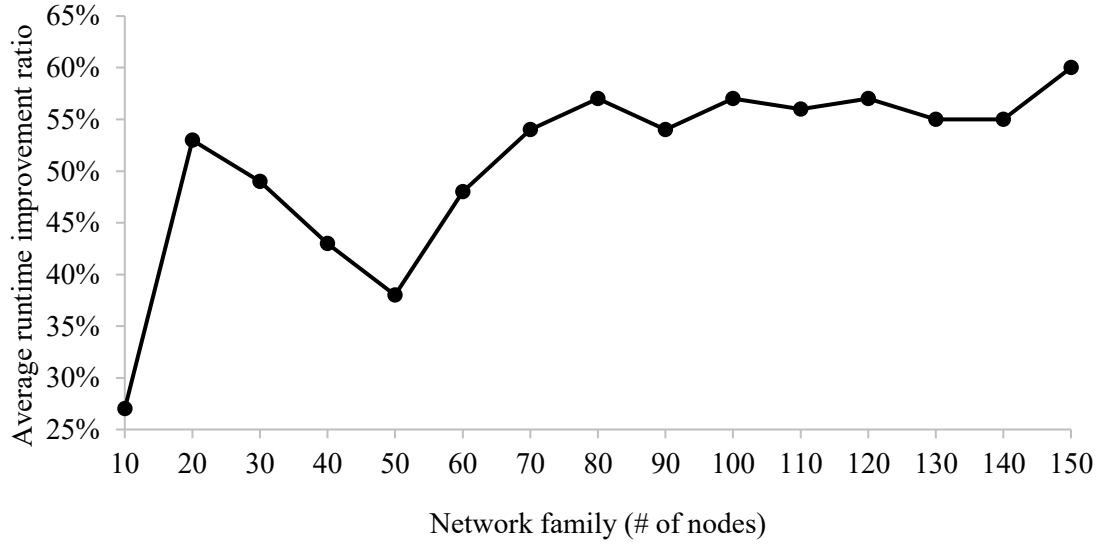


Figure 7.32 – Average runtime improvement ratio with regard to network scale

7.5.2 AVAILABILITY ANALYSIS

Our analysis of network availability for the benchmark multi-flow SBPP model and our new multi-flow SBPP model is shown in Figure 7.33 through Figure 7.47. Each panel of the figure corresponds to a single network family, where each data point represents the overall availability of the member of the family with the average nodal degree indicated on the x-axis for the benchmark multi-flow SBPP model and the new multi-flow SBPP model. First, we note that availability of networks arising from both models are between three nines and four nines, which meets the criteria presented in [117]. (i.e., both design models produce networks whose availabilities are generally acceptable). The availability of a network designed using the new multi-flow SBPP model is approximately 1.0×10^{-5} smaller (in absolute terms) on average compared to the availability of a network designed using the benchmark multi-flow SBPP model. However, we observe that as network scale grows larger, the difference between the two models becomes smaller, and in some cases, the new model has a larger availability. In order to find out how different the availability can be as network scale grows, we define *availability difference* as simply the availability of the network design arising from the new model minus the availability of the network design arising from the benchmark model. The values of availability difference are shown in Figure 7.48. Each point represents the availability difference for the network designs arising from the two models for each of the 165 test case networks. The x-axis represents an arbitrary "scale" of the network test case. We start with the leftmost data point corresponding to the sparsest member of the 10-node network family, then each subsequent data point corresponds to the next more highly connected member of the family, until we reach the most richly connected member of that family, after which the next data point

corresponds to the sparsest member of the next larger network family, and so on. It is observed that the availability difference generally increases from negative to positive as network scale grows.

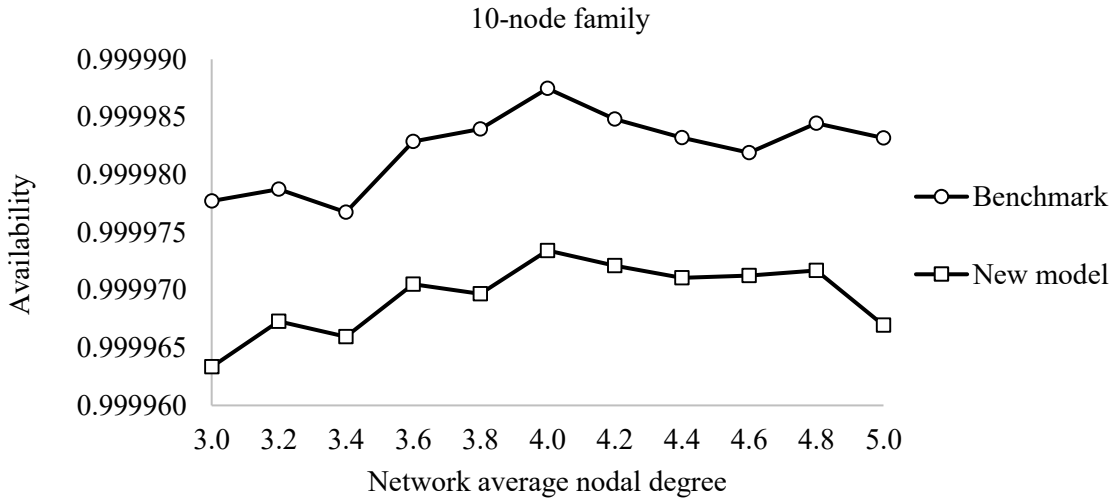


Figure 7.33 – Availability analysis of the 10-node network family

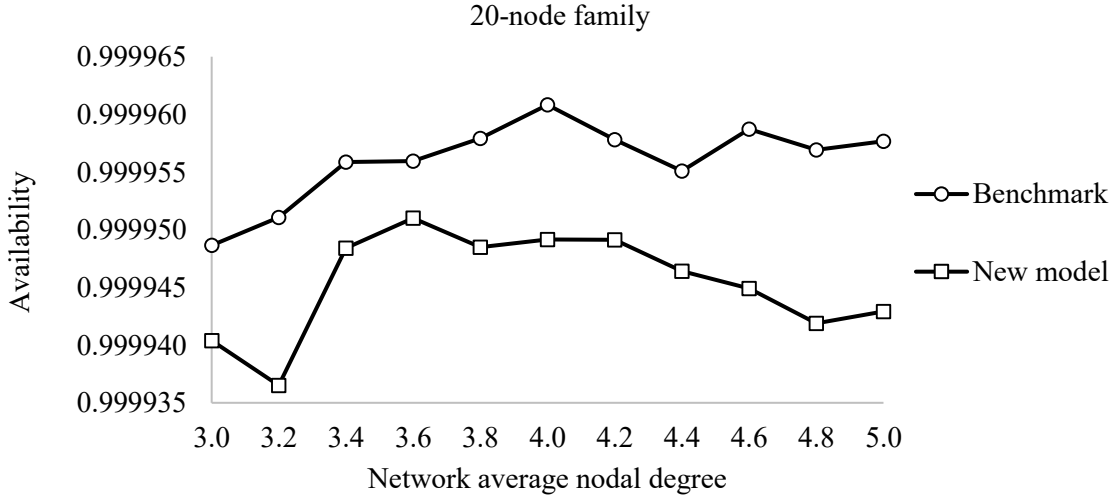


Figure 7.34 – Availability analysis of the 20-node network family

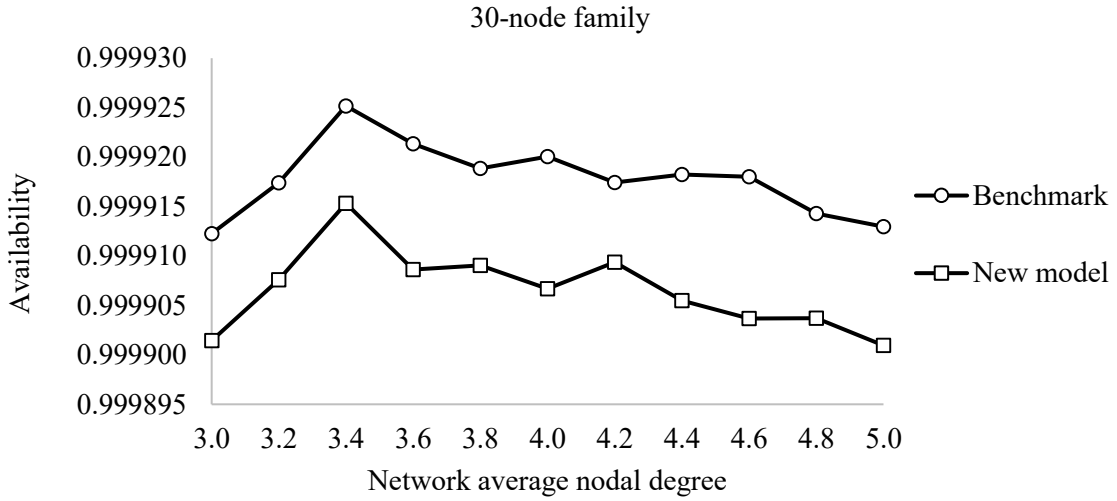


Figure 7.35 – Availability analysis of the 30-node network family

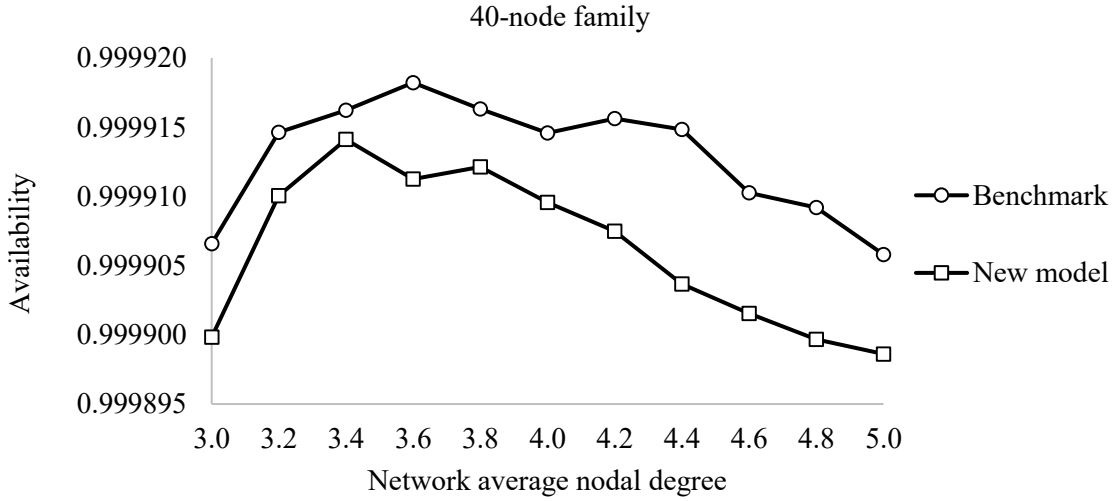


Figure 7.36 – Availability analysis of the 40-node network family

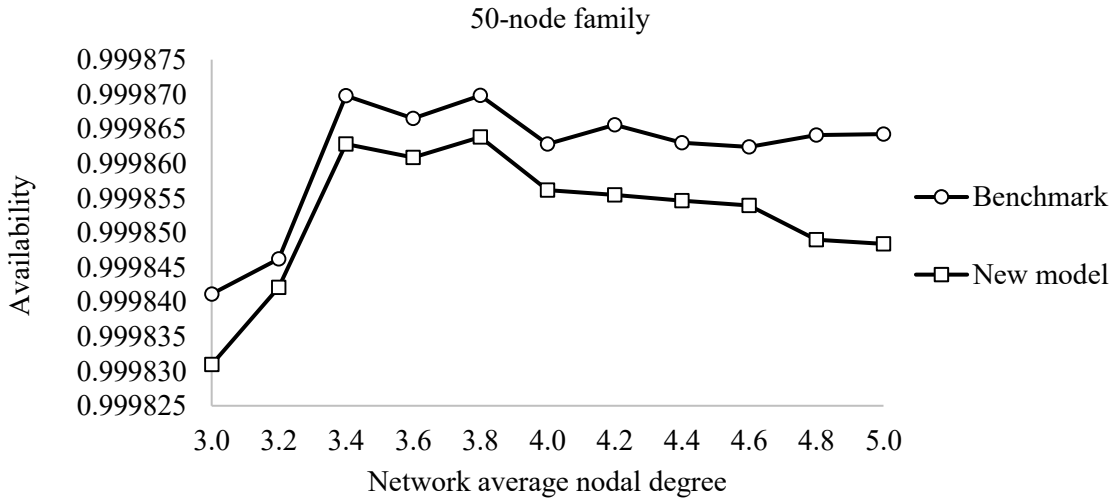


Figure 7.37 – Availability analysis of the 50-node network family

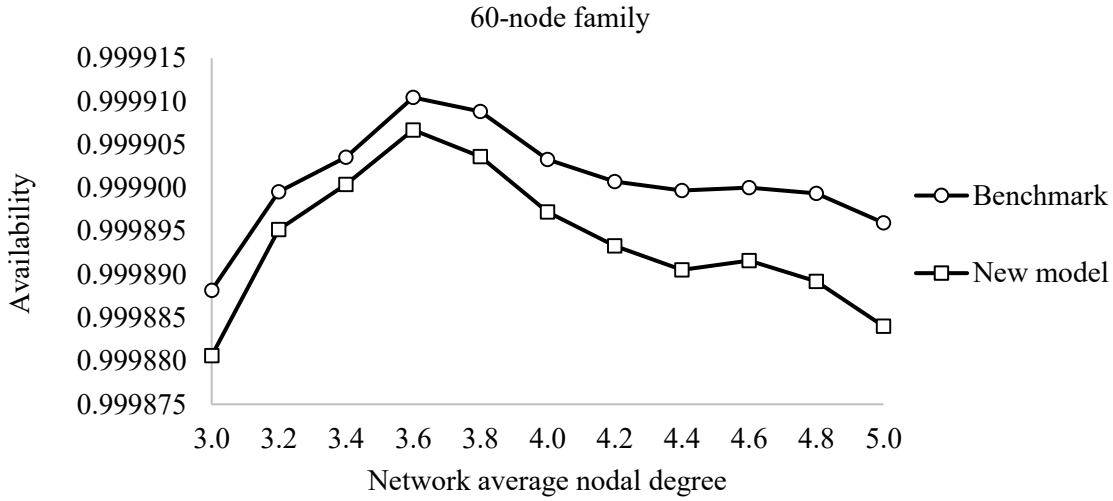


Figure 7.38 – Availability analysis of the 60-node network family

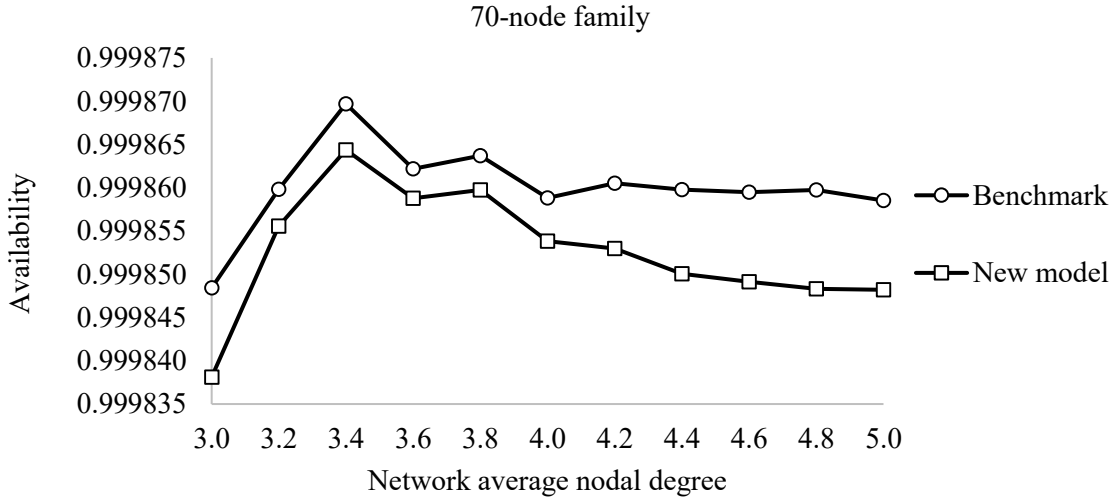


Figure 7.39 – Availability analysis of the 70-node network family

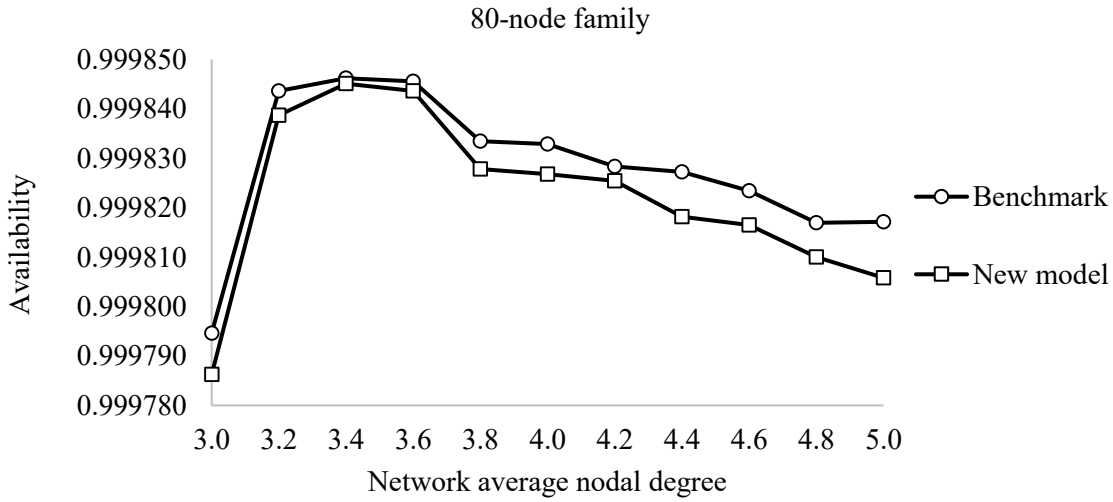


Figure 7.40 – Availability analysis of the 80-node network family

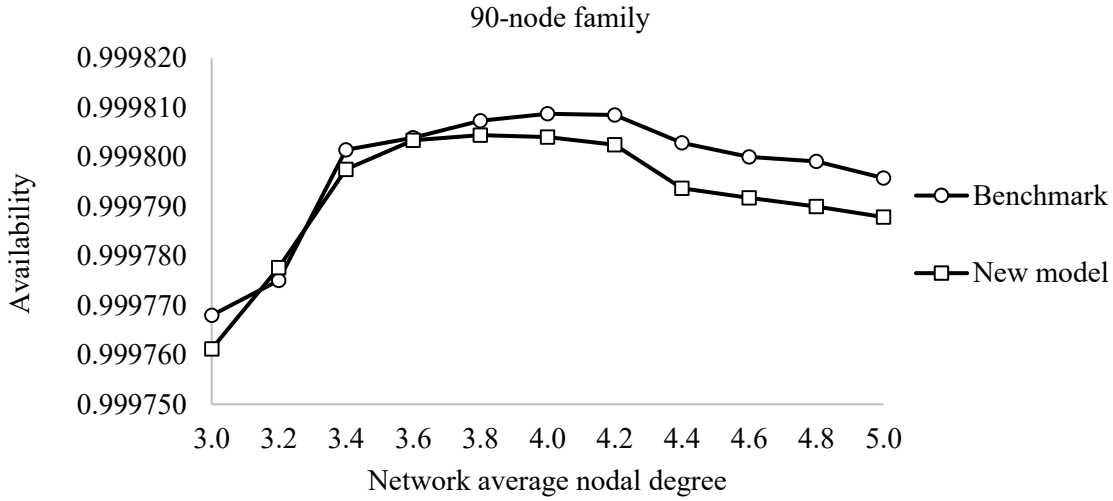


Figure 7.41 – Availability analysis of the 90-node network family

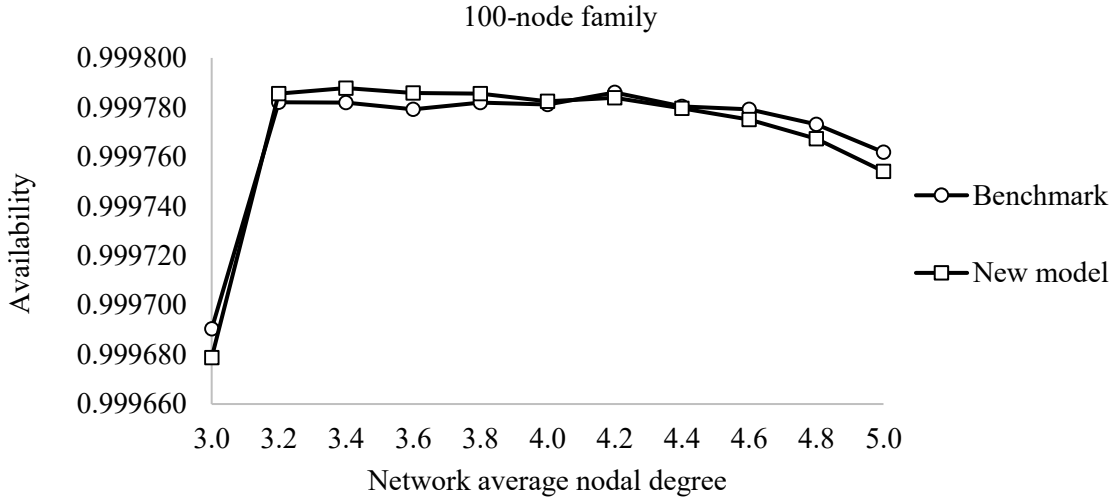


Figure 7.42 – Availability analysis of the 100-node network family

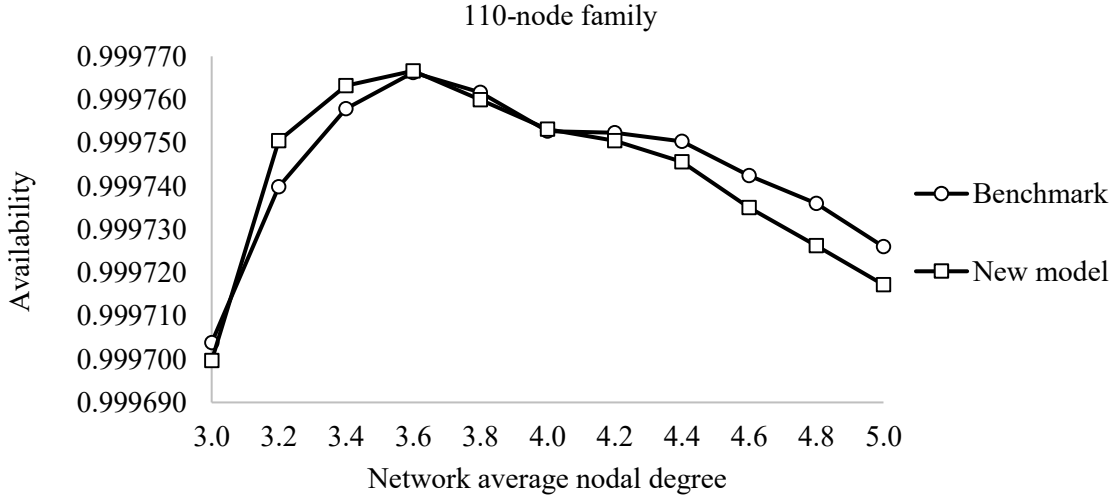


Figure 7.43 – Availability analysis of the 110-node network family

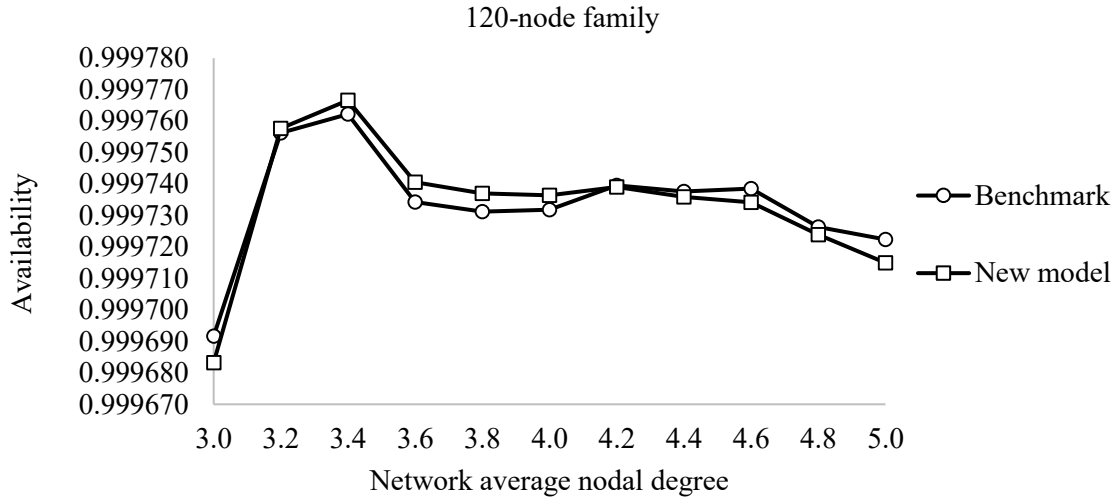


Figure 7.44 – Availability analysis of the 120-node network family

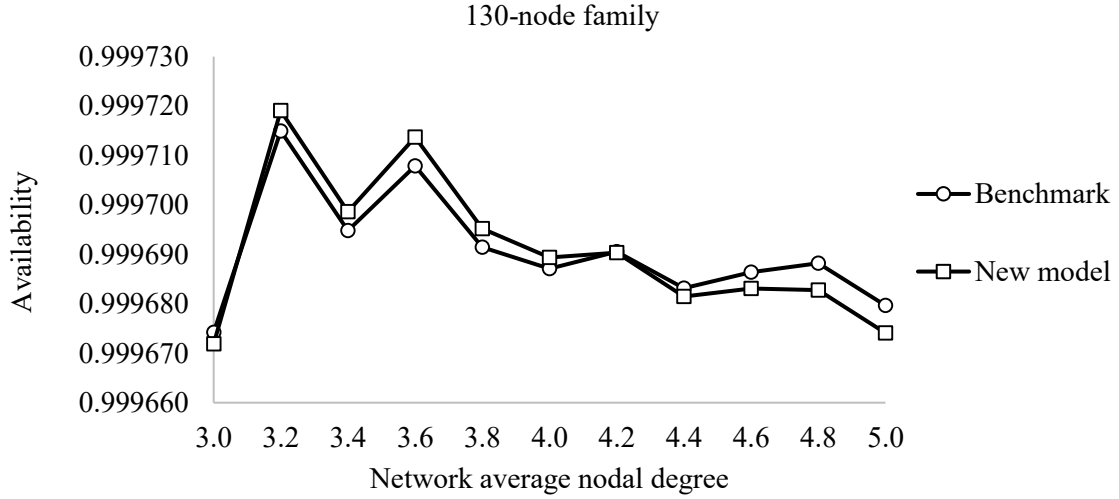


Figure 7.45 – Availability analysis of the 130-node network family

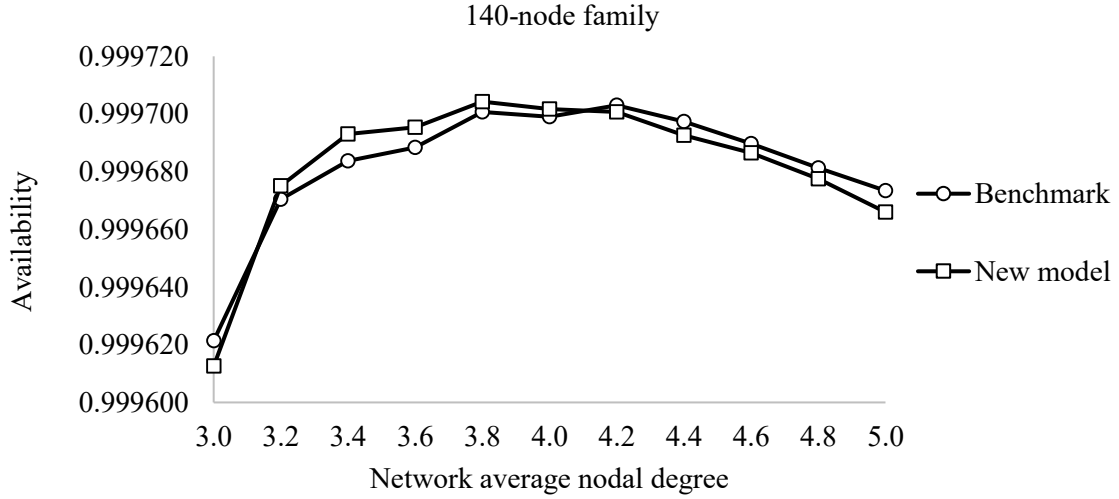


Figure 7.46 – Availability analysis of the 140-node network family

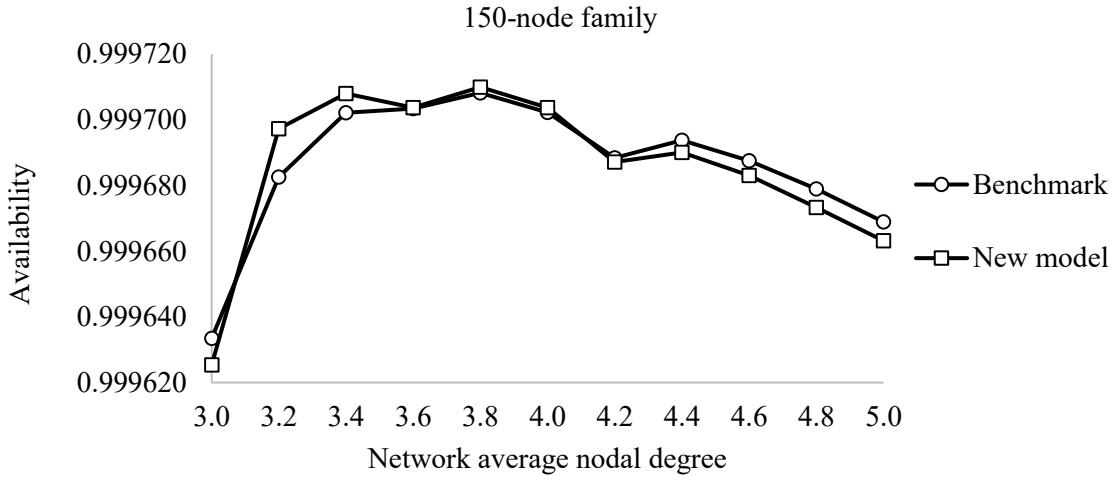


Figure 7.47 – Availability analysis of the 150-node network family

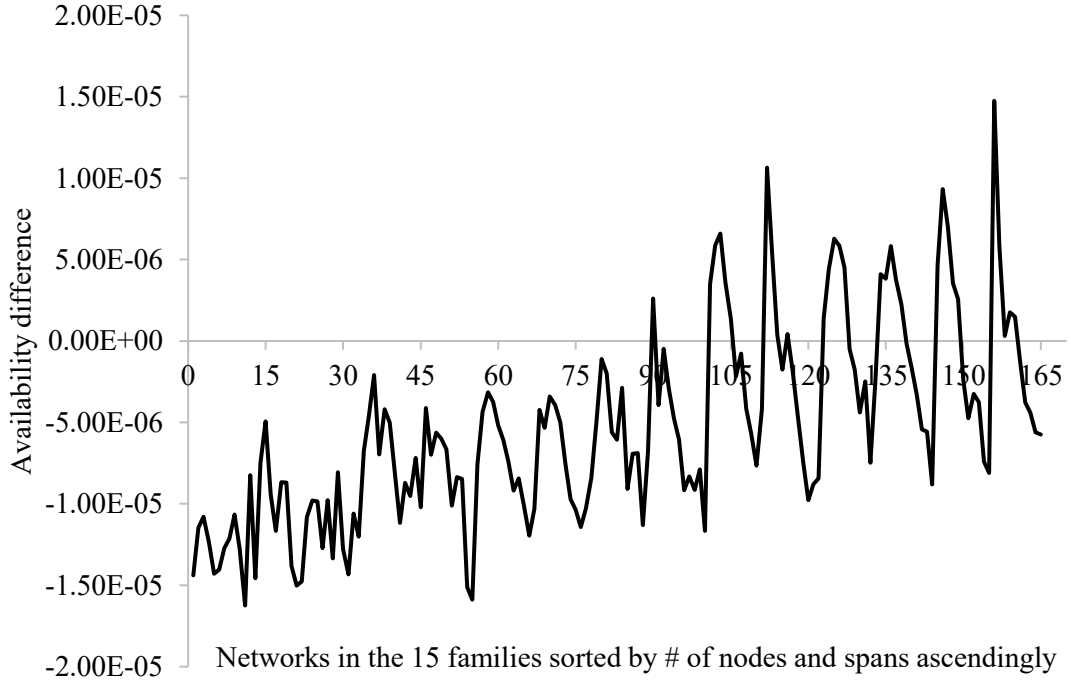


Figure 7.48 – Average availability difference with respect to network scale

From the above analysis, we can observe the general trend of how network availability reacts with increasing network connectivity and scale. Interestingly, network availability appears to be higher in networks with intermediate connectivity, with this trend becoming more apparent in large-scale networks (i.e., those with a greater number of nodes). It is not surprising to us that networks with low connectivity will have relatively

lower availability, as we expect that that arises from the poor diversity of backup routes by which two working routes can be restored following two simultaneous failures (or more generally, two failures overlapping in time). As network connectivity increases, the greater diversity of routes permits routing that will generally result in a greater ability to provide backup routing in the event of two failures. The decrease in availability as network connectivity increases further is somewhat more difficult to explain, as one would expect a more richly connected network to permit much more diverse routing such that two simultaneous failures are less likely to result in outage. And this would be the case if the network design approach produced backup routing assignments that specifically went out of their way to accommodate survivability in the event of two failures. However, the network designs considered only single-failure protection, and because the objective function emphasized capacity efficiency, working and backup routes did not to take as much advantage of the overall network route diversity as they could have. As a result, there are relatively more dual-failure scenarios that cause outage. We believe that as network connectivity increases, there are two major counteracting factors that will affect overall network availability: (1) higher connectivity generates to a greater diversity of working and backup routes and shorter routes, which results in fewer dual-failure scenarios interacting in a manner that can result in outage, and (2) higher connectivity permits better sharing of spare capacity, leading to more backup routes to be struck by a specified dual-failure scenario. Our hypothesis is that when network connectivity is still relatively low, the first factor dominates when connectivity increases, but as we move to higher and higher connectivity, the second factor starts to dominate.

To test our hypothesis, we selected three individual networks from the 10-node

family, with 15, 20, and 25 spans, respectively. The detailed topologies of the three networks are shown in Figure 7.49, where the nodes and spans of each network have been labeled explicitly (in the prior figures, we omitted those labels). As shown in the figure, the 20-span network consists of all spans in the 15-span network plus spans S9, S13, S16, S18, and S23, and the 25-span network consists of the spans in the 20-span network plus spans S4, S5, S15, S19, and S21. The demands are a full mesh of O-D pairs, as previously described, and these demands are identical for all three networks.

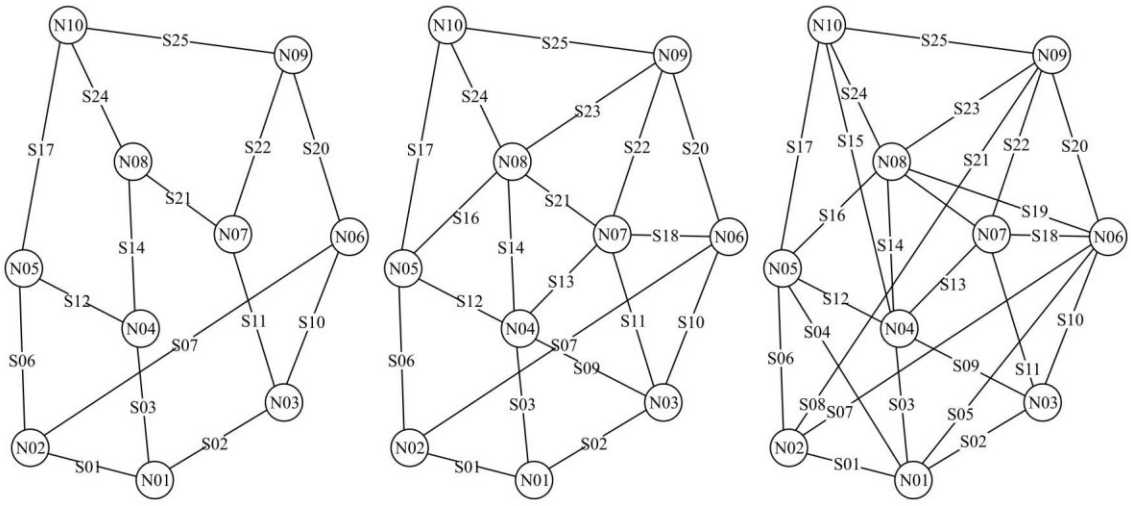


Figure 7.49 – Topologies of the 10-node 15-span network, 10-node 20-span network, and 10-node 25-span network

The capacity design results of the three networks are summarized in Table 7.1, while the detailed routing is shown in Table 7.2. We can see that as the connectivity of the network increases, demands are increasingly split onto multiple working and backup routes, which allows more and more demands to utilize shorter working and backup routes. More specifically, we note that moving from the 15-span network to the 20-span network permitted a greater number of demands to use shorter routes than moving from the 20-span network to the 25-span network, suggesting that this factor may be stronger at lower connectivity.

The spare capacities allocated to each span in the three networks are shown in Table 7.3. It is evident that as network connectivity increases, the amount of spare capacity of a span decrease in general, which reduces the capability of the network to route failed working routes in the event of a dual-failure scenario. Furthermore, the decrease in spare capacity is greater when moving from the 20-span network to the 25-span network than moving from the 15-span network to the 20-span network, suggesting that this factor may be stronger at higher connectivity.

Table 7.1– Summary on capacity design results of networks three 10-node test case networks

	15-span network	20-span network	25-span network
Fraction of demand pairs using multiple working routes	9/45	11/45	16/45
Fraction of demand pairs using multiple backup routes	9/45	11/45	16/45
Fraction of demand pairs using shorter routes than in the next lower connectivity network	N/A	32/45	19/45

Table 7.2– Spare capacities in the 10-node 15-span network, 10-node 20-span network, and 10-node 25-span network

	Spare capacity		
Spans	15-span	20-span	25-span
S01	11	5	9
S02	17	14	2
S03	18	19	9
S04	-	-	3
S05	-	-	1
S06	15	11	2
S07	14	11	5
S08	-	-	10
S09	-	9	9
S10	11	5	2
S11	15	5	4
S12	9	9	8
S13	-	13	10
S14	19	14	9
S15	-	-	8
S16	-	9	7

S17	12	8	8
S18	-	10	4
S19	-	-	9
S20	9	11	8
S21	19	10	1
S22	19	10	10
S23	-	10	13
S24	15	13	10
S25	19	11	4
Sum	222	207	165
Average	14.8	10.35	6.6

Table 7.3 – Capacity design results in the 10-node 15-span network, 10-node 20-span network, and 10-node 25-span network

Demands		10-node 15-span network				10-node 20-span network					10-node 25-span network				
		Working routes		Backup routes		Working routes		Backup routes		S?	Working routes		Backup routes		S?
O-D	Q	Routes	F	Routes	F	Routes	F	Routes	F		Routes	F	Routes	F	
N01-N02	2	S01	2	S03-S12-S06	2	S01	2	S03-S12-S06	2	N	S01	2	S04-S06	2	Y
N01-N03	9	S02	9	S01-S07-S10	9	S02	9	S03-S09	9	Y	S02	9	S03-S09	9	N
N01-N04	4	S03	4	S01-S06-S12	4	S03	4	S02-S09	4	Y	S03	3	S04-S12	3	N
											S04-S12	1	S03	1	
N01-N05	1	S01-S06	1	S03-S12	1	S03-S12	1	S01-S06	1	N	S04	1	S03-S12	1	Y
N01-N06	3	S01-S07	3	S02-S10	3	S02-S10	3	S01-S07	3		S05	2	S02-S10	2	Y
											S02-S10	1	S05	1	
N01-N07	10	S02-S11	10	S03-S14-S21	10	S03-S13	4	S02-S11	4	Y	S03-S13	1	S05-S18	1	N
						S02-S11	6	S03-S13	6		S05-S18	9	S03-S13	9	
N01-N08	1	S03-S14	1	S02-S11-S21	1	S03-S14	1	S02-S11-S21	1	N	S04-S16	1	S03-S14	1	Y
N01-N09	6	S02-S10-S20	6	S03-S14-S21-S22	6	S03-S13-S22	2	S02-S10-S20	2	Y	S05-S20	6	S01-S08	6	Y
						S02-S11-S22	4	S03-S14-S23	4						
N01-N10	6	S03-S12-S17	6	S02-S11-S22-S25	6	S01-S06-S17	3	S03-S14-S24	3	Y	S04-S17	3	S03-S15	3	Y
						S03-S12-S17	3	S02-S10-S20-S25	3		S04-S16-S24	3	S03-S15	3	
N02-N03	5	S07-S10	3	S01-S02	3	S01-S02	5	S07-S10	5	N	S01-S02	2	S07-S10	2	N
		S01-S02	2	S07-S10	2						S07-S10	2	S01-S02	2	
											S06-S12-S09	1	S07-S10	1	
N02-N04	7	S01-S03	5	S06-S12	5	S01-S03	7	S06-S12	7	N	S06-S12	7	S01-S03	7	N
		S06-S12	2	S01-S03	2										
N02-N05	2	S06	2	S01-S03-S12	2	S06	2	S01-S03-S12	2	N	S06	2	S01-S04	2	Y
N02-N06	8	S07	8	S01-S02-S10	8	S07	5	S01-S02-S10	5	N	S07	8	S08-S20	8	Y
						S01-S02-S10	3	S07	3						
N02-N07	9	S07-S10-S11	6	S06-S17-S25-S22	6	S07-S18	9	S06-S12-S13	9	Y	S07-S18	2	S08-S22	2	Y
		S01-S02-S11	3	S07-S20-S22	3						S06-S16-S21	4	S07-S18	4	
											S01-S03-S13	3	S07-S18	3	
N02-N08	10	S06-S12-S14	4	S07-S20-S22-S21	4	S06-S16	10	S07-S18-S21	10	Y	S06-S16	10	S08-S23	10	Y
		S01-S03-S14	4	S06-S17-S24	4										
		S06-S17-S24	2	S01-S03-S14	2										
N02-N09	2	S07-S20	2	S06-S17-S25	2	S07-S20	2	S06-S17-S25	2	N	S08	2	S07-S20	2	Y
N02-N10	1	S06-S17	1	S07-S20-S25	1	S06-S17	1	S07-S20-S25	1	N	S08-S25	1	S06-S17	1	Y
N03-N04	2	S02-S03	2	S11-S21-S14	2	S09	2	S11-S13	2	Y	S09	2	S11-S13	2	N
N03-N05	1	S02-S01-S06	1	S11-S21-S14-S12	1	S11-S21-S16	1	S09-S12	1	Y	S09-S12	1	S02-S04	1	Y
N03-N06	4	S10	4	S11-S22-S20	4	S10	4	S11-S18	4	Y	S10	4	S11-S18	4	N
N03-N07	3	S11	3	S10-S20-S22	3	S11	3	S09-S13	3	Y	S11	3	S09-S13	3	N
N03-N08	8	S11-S21	8	S02-S03-S14	8	S11-S21	5	S09-S14	5	Y	S11-S21	6	S09-S14	6	N
						S09-S14	3	S11-S21	3		S10-S19	1	S09-S14	1	
											S09-S14	1	S11-S21	1	
N03-N09	2	S11-S22	2	S10-S20	2	S11-S22	2	S10-S20	2	N	S11-S22	2	S10-S20	2	N
N03-N10	1	S10-S20-S25	1	S11-S21-S24	1	S09-S12-S17	1	S10-S20-S25	1	Y	S09-S15	1	S11-S22-S25	1	Y
N04-N05	9	S12	9	S03-S01-S06	9	S12	9	S14-S16	9	Y	S14-S16	1	S12	1	Y
											S12	8	S15-S17	8	
N04-N06	1	S03-S01-S07	1	S14-S21-S22-S20	1	S13-S18	1	S09-S10	1	Y	S09-S10	1	S14-S19	1	N
N04-N07	9	S14-S21	9	S03-S02-S11	9	S13	9	S14-S21	9	Y	S14-S21	2	S13	2	N
											S09-S11	7	S13	7	
N04-N08	5	S14	5	S12-S17-S24	5	S14	5	S12-S16	5	Y	S14	5	S15-S24	5	N
N04-N09	9	S12-S17-S25	1	S14-S21-S22	1	S13-S22	5	S14-S23	5	Y	S13-S22	9	S14-S23	9	N
		S14-S24-S25	5	S03-S02-S11-S22	5	S14-S23	4	S13-S22	4						
		S12-S06-S07-S20	3	S14-S21-S22	3	S01	2	S03-S12-S06							
N04-N10	10	S12-S17	10	S14-S24	10	S12-S17	5	S14-S24	5	N	S14-S24	2	S15	2	Y
						S12-S16-S24	3	S13-S22-S25	3		S15	8	S12-S17	8	
						S13-S22-S25	2	S12-S17	2						
N05-N06	4	S06-S07	4	S17-S25-S20	4	S12-S13-S18	2	S06-S07	2	Y	S16-S19	1	S04-S05	1	Y
						S12-S09-S10	2	S06-S07	2		S12-S13-S18	3	S16-S19	3	
N05-N07	7	S17-S24-S21	5	S06-S07-S20-S22	5	S16-S21	4	S12-S13	4	Y	S12-S13	1	S16-S21	1	N
		S12-S03-S02-S11	2	S17-S25-S22	2	S12-S14-S21	3	S17-S25-S22	3		S16-S21	6	S12-S13	6	
N05-N08	4	S12-S14	3	S17-S24	3	S16	4	S12-S14	4	Y	S16	4	S17-S24	4	N
		S17-S24	1	S12-S14	1										
N05-N09	3	S17-S25	3	S06-S07-S20	3	S16-S23	2	S17-S25	2	Y	S17-S25	3	S16-S23	3	N
						S12-S13-S22	1	S17-S25	1						
N05-N10	5	S17	5	S12-S14-S24	5	S17	5	S16-S24	5	Y	S17	5	S12-S15	5	N
N06-N07	1	S10-S11	1	S20-S22	1	S18	1	S20-S22	1	Y	S18	1	S20-S22	1	N
N06-N08	10	S20-S25-S24	1	S10-S11-S21	1	S18-S21	10	S20-S23	10	Y	S19	4	S20-S23	4	Y
		S20-S22-S21	9	S07-S06-S12-S14	9						S18-S21	6	S19	6	
N06-N09	10	S20	10	S10-S11-S22	10	S20	10	S18-S22	10	Y	S20	9	S19-S23	9	N
											S18-S22	1	S20	1	
N06-N10	4	S20-S25	4	S07-S06-S17	4	S20-S25	4	S07-S06-S17	4	N	S19-S24	4	S20-S25	4	Y

Table 7.3– Capacity design results in the 10-node 15-span network, 10-node 20-span network, and 10-node 25-span network (cont.)

Demands		(10-node, 15-span)				(10-node, 20-span)					(10-node, 25-span)				
		Working routes		Backup routes		Working routes		Backup routes		S?	Working routes		Backup routes		S?
O-D	Q	Routes	F	Routes	F	Routes	F	Routes	F		Routes	F	Routes	F	
N07-N08	10	S21	10	S22-S25-S24	10	S21	9	S13-S14	9	Y	S21	10	S22-S23	10	N
						S13-S14	1	S21	1		S22	1	S21-S23	1	
N07-N09	7	S22	7	S21-S24-S25	7	S22	7	S18-S20	7	Y	S18-S20	6	S22	6	N
N07-N10	7	S22-S25	3	S21-S24	3	S21-S24	7	S22-S25	7	N	S22-S25	5	S13-S15	5	N
		S21-S24	4	S22-S25	4						S21-S24	2	S13-S15	2	
N08-N09	8	S21-S22	5	S24-S25	5	S23	8	S24-S25	8	Y	S23	4	S24-S25	4	N
		S24-S25	3	S21-S22	3						S21-S22	3	S23	3	
											S19-S20	1	S23	1	
N08-N10	8	S24	8	S14-S12-S17	8	S24	8	S16-S17	8	Y	S24	7	S16-S17	7	N
											S16-S17	1	S24	1	
N09-N10	10	S25	10	S22-S21-S24	10	S25	10	S23-S24	10	Y	S25	10	S23-S24	10	N
Note	The demands for the three networks are fixed with the same O-D pairs (column "O-D") and demands quantity (column "Q"). Both working routes and backup routes are shown with routes (columns "Routes") and flow values (columns "F"). Meanwhile, for networks (10-node, 20-span) and (10-node, 25-span), there is a column "S", which denotes whether the working or backup routes are shorter compared with the previous network. Note that if there are more than one working or backup route for a specified demand, as long as one of the routes is shorter, we assume "S" is a Yes (i.e., "Y" as denoted in the table).														

We can also investigate further. As discussed previously, network availability is a function of the values of wf_{aff} and wf_{lost}. Using the 10-node network family described above as a test case, we can investigate how these two characteristics change with network connectivity. While each dual-failure scenario will produce their own of wf_{aff} and wf_{lost} values, for simplicity, we will examine only the average values over all dual-failure scenarios, with the results shown in Figure 7.50. Each data point in the figure represents the average of the wf_{aff} values ("Average affected flow") and the average of the wf_{lost} ("Average lost flow) values in the 10-node test case network of the indicated connectivity, with the lost flow associated with the left y-axis and the affected flow associated with the right y-axis. As network connectivity increases, the working traffic affected by a dual-failure scenario increases quite steadily, while there is a corresponding decrease in the amount of working traffic that is failed but not restored. The result is an overall general decrease in the ratio of lost-to-affected working traffic, thereby generally driving availability down.

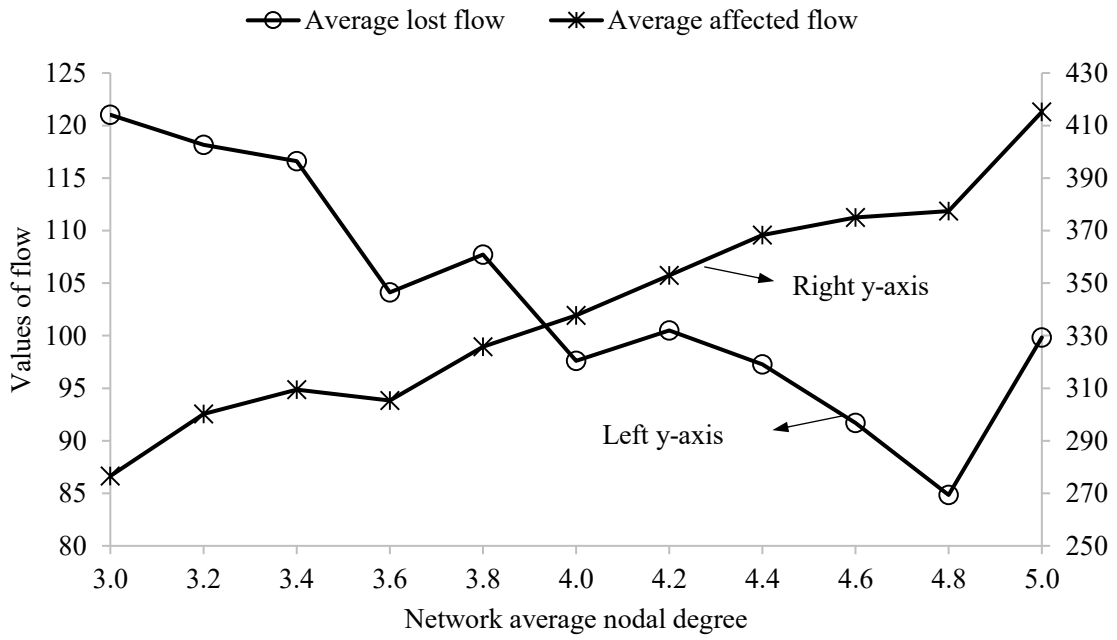


Figure 7.50 – Average values of lost flow and affected flow in 10-node network family

7.6 CONCLUSIONS

We have proposed a new multi-flow SBPP ILP design model and developed an algorithm to analyze network overall availability for multi-flow SBPP networks. Our key findings are as follows:

(1) The runtime of the new multi-flow SBPP model is 51% faster on average than the runtime of the benchmark model, with larger speed improvements for higher connectivity networks.

(2) Both models generally meet published network availability standards. Our new model results in a slight decrease in availability (1.0×10-5 in absolute terms on average in the 165 networks tested). However, the new model provides better availability than the benchmark model in the larger networks with higher connectivity, topping out at approximately 1.5×10-5 better in one of the 150-node networks.

(3) Network availability tends to be highest for networks of moderate connectivity, and there is no obvious difference between the two models in this regard.

(4) Network availability decreases with increasing network scale (i.e., number of nodes), and there is no obvious difference between the two models in this regard.

CHAPTER 8 AVAILABILITY OPTIMIZATION AND IMPACT OF SPARE CAPACITY ON NETWORK AVAILABILITY FOR SHARED BACKUP PATH PROTECTION NETWORKS[4]

8.1 MOTIVATIONS AND GOALS

The previous chapter proposes an approach to calculate network availability for SBPP networks. However, the proposed algorithm cannot guarantee an optimal network availability. In addition, the relationship between network availability and extra spare capacity for multi-flow SBPP has not been involved in the literature. Therefore, in this chapter, we seek to achieve the following goals:

(1) Develop an algorithm that determines the backup routing required in dual-failure scenarios to maximize SBPP network availability, given an existing capacity plan.

(2) Determine how increases in backup capacity within an SBPP network will impact its availability.

[4] This chapter is adapted from our journal paper: W. Wang, J. Doucette, "Availability Optimization and Spare Capacity Impact Analysis for Shared Backup Path Protection Networks," *Journal of Optical Communications and Networking (JOCN)*, in review; first submitted on July 26 2017; revised and resubmitted on November 17 2017.

8.2 AVAILABILITY OPTIMIZATION ALGORITHM

8.2.1 NOTATIONS

The following symbols are utilized in this section.

Sets:

S is the set of all spans in the network.

P is the set of all the working routes.

B^p is the set of all backup routes of the specified primary route p.

Parameters:

s_k is an integer parameter, representing the amount of backup capacity designed for the specified span k.

wf^p is an integer parameter, representing the amount of working flow designed for the primary route p.

Variables:

$bf^{b,p}$ is an integer variable, representing the amount of backup flows on the specified backup route b for restoration of primary route p.

8.2.2 AVAILABILITY OPTIMIZATION ALGORITHM

As stated above, our first goal is to develop an algorithm to maximize a network's availability. We assume an existing network design that is at least fully single-failure survivable. As such, we already have complete working lightpath routing as well as a

corresponding set of eligible backup routes and the specific backup lightpath routing in use for single failures. More specifically, this includes the set of all the spans in the network (i.e., the set S), the set of all working routes (i.e., the set P), the set of all eligible backup routes that can be used by each working route p (i.e., the set $B^p, \forall\, p \in P$), the number of working lightpaths on each working route (i.e., $wf^p, \forall p \in P$), the number of backup lightpaths on each backup route (i.e., $bf^{b,p}, \forall b \in B^p, p \in P$), and the amount of backup capacity on each span (i.e., $s_k, \forall k \in S$).

For convenience, we obtain this initial network design by solving the optimal single-failure SBPP design model from [77] (i.e., it is the minimum capacity possible for full single-failure survivability). However, any pre-existing network design will suffice, irrespective of whether the design is a capacity-efficient one or not. We will use this to our advantage later when we test the impact of providing extra backup capacity to the network.

As discussed earlier, calculating an SBPP network's availability (or more specifically, what we call dual-failure availability [25]) is difficult, with no closed form solution. As such, there is no straightforward method for incorporating an availability calculation in the objective function of an SBPP network design ILP model. Consequently, determining the maximum availability of an SBPP network requires some form of heuristic or algorithmic approach. We illustrate our approach in the form of the pseudocode in Figure 8.1. It is an iterative approach, which iterates over each dual-failure scenario $(i,j), \forall i,j \in S | i \neq j$, in order to calculate the value of $R_2(i,j), \forall i,j \in S | i \neq j$.

Recall from Chapter 4 that $wf_{\text{aff}}(i,j)$ represents the number of working lightpaths affected by dual failure of spans i and j, and that $wf_{\text{lost}}(i,j)$ indicates the number of

working lightpaths lost in that scenario. At the beginning of each iteration, we initialize $wf_{\text{aff}}(i,j)$ and $wf_{\text{lost}}(i,j)$ to zero in row 04, and set about to calculate those two quantities. As we assumed that we start with a network that is fully single-failure survivable, the protection response for the first failure i is known from the pre-existing design. Accordingly, for each selected working route p that is affected by failure i, we calculate the variable $wf_{\text{aff}}(i,j)$ by adding the number of working lightpaths routed on p in row 06, and then recalculate the available backup capacity on its backup routes by removing the occupied backup capacity in rows 07 through 10. Next, in rows 11 through 14, we cycle through each backup route b of the primary route p to check whether it is affected by the second failure j. If b is affected by j, it cannot be exploited to restore the working route p. That is, the working lightpaths restored by this backup route b is lost. Thus, the value of $bf^{b,p}$ is added up to the value of $wf_{\text{lost}}(i,j)$ in row 12. Once all of the j-affected backup routes in B^p are considered, we will go to the next working route p until all the i-affected working routes are considered. The subsequent step in rows 16 through 18 is to deal with the working routes that are affected by the second failure j excluding those that have been affected by the first failure i.

Next in row 19, we need to determine how many working lightpaths crossing span j but not i are lost. Unlike the response when span i fails, the order in which the working lightpaths are considered when span j fails will influence the calculation of network availability, given that their backup lightpaths will compete for the available backup capacity (since there is generally not enough for restoring this second failure). If we want to investigate the impact of the increased spare capacity on network availability, it is a must to keep the assignment of the previous spare capacity (i.e., the spare capacity excluding the

newly increased) exactly the same in both the previous availability analysis process (i.e., before spare capacity increase) and the new availability analysis process (i.e., after spare capacity increase). It is not scalable in terms of memory and runtime to store all the assignment schemes of capacity for all the working routes, backup routes, working capacity, and backup capacity under the second failure in all dual-failure scenarios. To solve this issue, we develop a custom ILP model (described in the next subsection) to minimize the total lost lightpaths wf_{lost} on the working routes affected only by the second failure in a specified dual-failure scenario. In this regard, the ILP model saves us the trouble of considering the sequence of affected working routes and other assignment details as well. The minimal total lost working lightpaths is then added to the variable of $wf_{\text{lost}}(i,j)$ in row 20, and $R_2(i,j)$ is calculated as per Eq. (4-9) in row 21. The same procedure is repeated until all the dual-failure scenarios are covered. Finally, we can calculate network availability as per Eq. (4-10) and (4-11) in row 23.

```
01    Begin
02    Input data: S,P,B^p,wf^p,bf^{b,p}, and s_k for all p∈P, k∈S
03    For dual-failure(i,j) for all i,j∈S and i≠j
04          Initialize wf_aff(i,j)=wf_lost(i,j)=0
05          For each working route p affected by failure i
06                Add its working flow to wf_aff(i,j)
07                Update available spare capacity for spans on p
08                For each backup route b of p
09                      Update spare capacity for spans
10                End for
11                For backup route b of p affected by failure j
12                      Add its backup flow to wf_lost(i,j)
13                      Update spare capacity on span j
14                End for
15          End for
16          For working route p affected by failure j but not i
17                Add its working flow to wf_aff(i,j)
18          End for
19          Optimize total lost flow caused only by failure j
20          Calculate total lost flow under dual-failure (i,j)
21          Calculate restorability under dual-failure (i,j)
22    End for
23    Calculate network availability
24    End
```

Figure 8.1 – Pseudocode for availability analysis algorithm for SBPP networks

8.3 MINIMIZING LOST WORKING LIGHTPATHS VIA ILP

While the restoration response to the first failure is known (it was a part of the inputs to the algorithm) and therefore easy to deal with in the algorithm, the restoration response to the second failure is not known, and could take many different configurations. As

described above, we utilize a custom ILP model to determine the specific response to the second failure (i.e., the routing of the backup lightpaths) that will minimize the working lightpaths lost due to the dual failure scenario in question.

8.3.1 ILP NOTATIONS

The ILP model makes use of the following sets, parameters, and variables.

Sets:

P' is the set of all primary routes affected by the second failure but not affected by the first failure.

B^p is the set of all backup routes for working lightpath p.

S^b is the set of all spans on backup route b.

CB^k is the set of all backup routes that traverse span k.

S' is the set of all spans on all the backup routes in $CB^k \cap B^p$.

Parameters:

wf^p is the number of working lightpaths on primary route p for the failed span in question (the second failure).

s_k is an integer parameter that represents the amount of backup capacity on span k.

Variables:

$wf_{\text{lost}}^p \geq 0$ is the number of lost working lightpaths on the specified primary route p due to the second failure in a dual-failure scenario.

$bf^{b,p} \geq 0$ is the number of backup lightpaths on backup route b for restoration of primary route p. This is defined the same here as it is in the algorithm above, but p here refers only to the second of the dual failures.

8.3.2 ILP FORMULATION

The ILP model is comprised of equations (8-1) through (8-4), below.

Minimize:

$$wf_{\text{lost}} = \sum_{p \in P'} wf_{\text{lost}}^{p} \tag{8-1}$$

Subject to:

$$wf_{\text{lost}}^{p} \geq wf^{p} - \sum_{b \in B^p} bf^{b,p'} \qquad \forall p \in P' \tag{8-2}$$

$$bf^{b,p'} \leq s_k \qquad \forall p \in P',\ b \in B^p,\ k \in S^b \tag{8-3}$$

$$\sum_{p \in P'} \sum_{b \in CB^k \cap B^p} bf^{b,p'} \leq s_k \qquad \forall k \in S' \tag{8-4}$$

As described above, the ILP model seeks to find the backup lightpath routing in response to a second failure such that the availability is maximized. This is effectively accomplished by maximizing the value of $R_2(i,j)$ as per Eq. (4-10), which in turn is accomplished by minimizing $wf_{\text{lost}}(i,j)$ as per Eq. (4-9). The ILP functions within each dual failure (i,j) independently, so for clarity and convenience, we omit the subscript (i,j) of each relevant variable in the formulation. The objective function of this ILP model is expressed in Eq. (8-1), where each wf_{lost}^{p} represents the amount of lost working lightpaths on working route p under the current dual-failure scenario.

If the sum of the available backup lightpaths on all backup routes available to restore lightpaths on working route p is no less than the number of working lightpaths originally assigned to it, then the lost working lightpaths can be accommodated fully. That is, the number of lost working lightpaths on primary route p, due to the second failure in the current dual-failure scenario, is equal to zero. Otherwise, the number of lost working lightpaths on primary route p is the difference between its working lightpaths and the sum of all its backup lightpaths. Equation (8-2) ensures that the number of lost working lightpaths on a specified working route p is no less than the difference between its working lightpaths and the sum of all its backup lightpaths (i.e., $wf^p - \sum_{b \in B^p} bf^{b,p'}$).

Eq. (8-3) and (8-4) ensure that the backup lightpaths selected do not exceed the spare capacity available on all spans they cross, either individually for all backup lightpaths in response to the second failure only, as in Eq. (8-3), or concurrently with all other backup lightpaths in use by the first failure, as in Eq. (8-4).

8.4 EXPERIMENTS

8.4.1 EXPERIMENTAL NETWORKS AND SETUP

We test our approach on a total of 165 test case networks comprised of 15 network families with 11 networks in each family. Please refer to Chapter 5 for details of experimental networks and experimental setup. Note that Gurobi is called through Python in each loop wherever an ILP model is involved.

8.4.2 INTEGRATION OF GUROBI AND PYTHON

The Gurobi manual, which can be found on Gurobi official website [101], has covered the approach to call Gurobi within Python. Overall, two ways can be used to achieve this: one is to deploy the Gurobi Python Interface through Gurobi Interactive Shell, the other is to apply Gurobi within the existing Python environment (in which case, you need to install Gurobi module into Python). In this chapter, we adopt the second approach.

After setup of the programming environment, the next step is to implement the ILP formulation following the grammar specified in the manual. It should be noted that all Gurobi Python applications always start with importing Gurobi functions and classes into Python, using the sentence [101]:

```
From gurobipy import *
```

Meanwhile, Gurobi offers various methods to simplify the programming process. For example, the method `Model()` is used to create a new optimization model, and the method `addVar()` is for adding variables to the model. Additionally, arithmetic operators and comparison operations are overloaded in Python in order to ease the process of building objective functions and constraints. The implementation of our optimization model in Python is more complicated, considering Gurobi needed to be called in each iteration of dual-failure scenario (i, j).

8.4.3 VALIDATION OF PROPOSED ALGORITHM

In evaluating the proposed SBPP network availability optimization algorithm, we use as a prior SBPP network availability algorithm already in the literature [119] as a

benchmark. That prior algorithm did not seek to optimize availability, rather, it simply provided a passive availability calculation. The present algorithm herein, however, specifically determines the response to a second failure so as to maximize the resultant network availability. The data in Figure 8.2 shows the results from solving our new algorithm from this section with the benchmark. Each data point in the figure represents the SBPP network availability calculated by the new algorithm or the benchmark, as indicated, for the various test case networks increasing in scale as we move to the right on the x-axis. More specifically, the leftmost data point corresponds to the sparsest member in the 10-node network family, each subsequent data point corresponds to the next more richly connected member in that family, and up to the 10-node master network, after which the next data point corresponds to the sparsest member of the 15-node network family, then its sequentially more richly connected members, and so on. Figure 8.3 shows similar data, but only for the master networks.

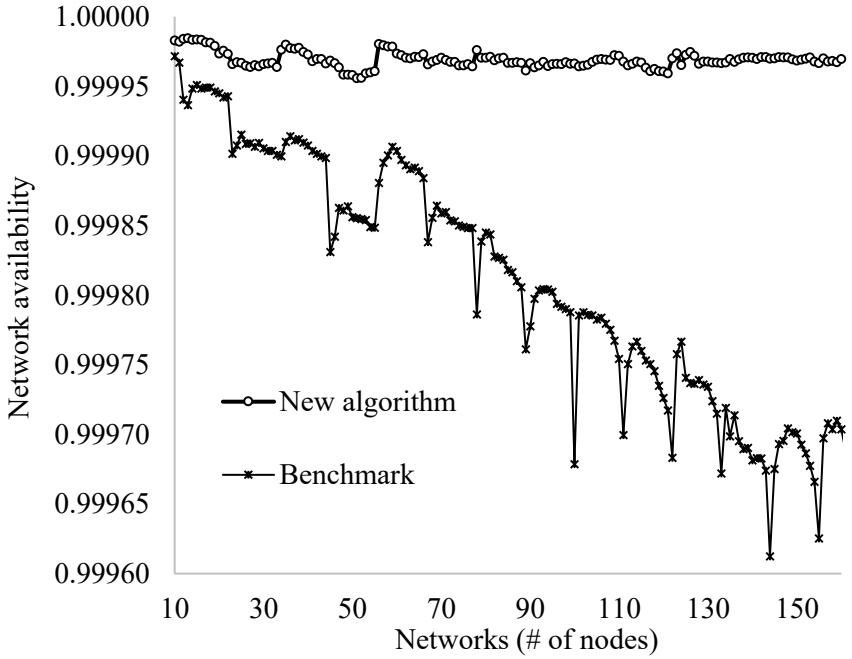


Figure 8.2 – Maximized network availability via the new algorithm for all 165 networks

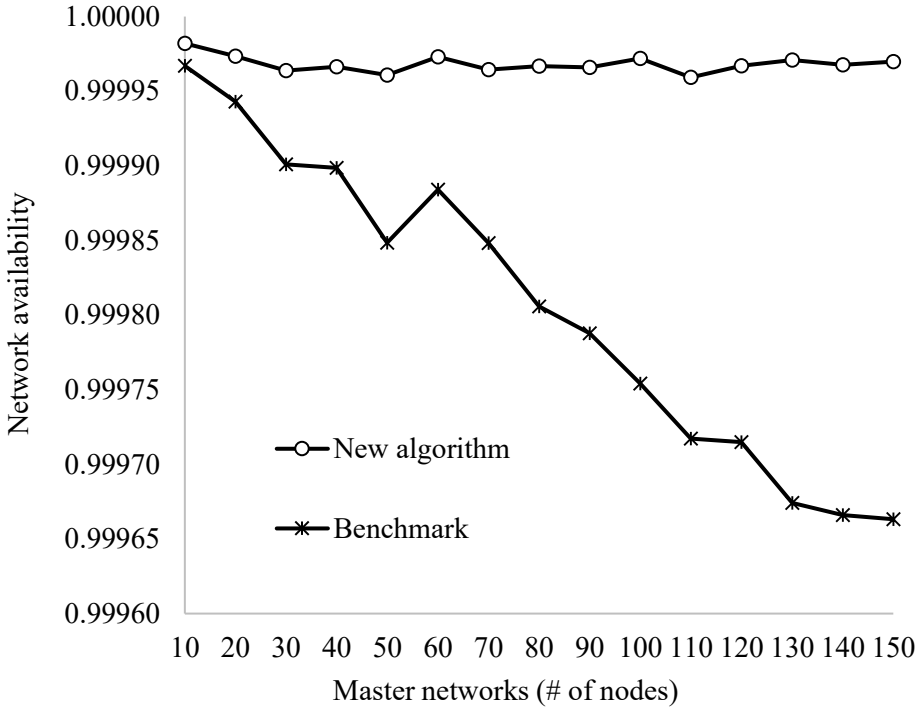


Figure 8.3 – Maximized network availability via the new algorithm for master networks only

First, it is clear that the network availability corresponding to our new algorithm are higher than those corresponding to the benchmark algorithm for each test case network. This is reasonable since our new algorithm seeks the optimal network availability. In contrast, the benchmark algorithm is purely passive in nature; when assessing the restoration of a second failure, the failed working routes in a specified dual-failure scenario is selected and restored randomly without regard for optimum availability. As a consequence, there is no guarantee that the selection of backup lightpaths is a particularly good one, let alone an optimum one, and so the network availability that arises is not necessarily optimal.

We can observe from the figure that optimal network availability is not only much higher when the proposed algorithm is applied, but is also relatively stable with increasing network scale for the new algorithm. In contrast availability that arises when the

benchmark algorithm is applied decreases when network scale grows. This reveals that network scale has little impact on optimal network availability using our new approach; the fact that optimal network availability tends to be stable with increasing network scale is the result of two factors that work in opposite directions in terms of the impact on optimal network availability. On one hand, higher network scale means more nodes and spans are involved, which can generate a greater diversity of available backup routes for a specified O-D pair and results in higher optimal network availability. On the other hand, higher network scale represents more demands and more requirements for available spare capacity, which leads to higher sharing of spare capacity and lower optimal network capacity. These two factors interact and their effects towards optimal network availability offset each other. As such, we end up with relatively stable optimal network availability when increasing network scales.

8.4.4 Complexity and Solution Time

Problem complexity and scalability is of general concern with new ILP models and algorithms. The ILP model developed in Section 8.3 has the numbers of variable and constraints as shown in Table 8.1, and the solution time of the algorithm is as shown in Figure 8.4.

Table 8.1– Numbers of Variables and Constraints in Availability Optimization ILP Model for Master Networks

Nodes	Spans	Instances	Mean Variables per Instance	Mean Constraints per Instance
10	25	600	9	18
20	50	2,450	23	60
30	75	5,550	39	112
40	100	9,900	53	164
50	125	15,500	71	233
60	150	22,350	94	330
70	175	30,450	117	430
80	200	39,800	130	469
90	225	50,400	177	740
100	250	62,250	194	803
110	275	75,350	201	779
120	300	89,700	241	1,013
130	325	105,300	278	1,254
140	350	122,150	336	1,664
150	375	140,250	361	1,789

Note that the numbers in Table 8.1 represent the average numbers of variables and constraints in each instance of the ILP for the master network indicated. The ILP is called once per dual-failure scenario (i.e., twice per span pair, since failure order matters), which is shown in the "Instances" column in the table. So for instance, for the 100-node master network (with 250 spans), the algorithm would need to call the ILP $|S| \times (|S| - 1) =$ 62,250 time, with an average of 194 variables and 803 constraints in the ILP each time it is called. The precise numbers of variables and constraints will vary somewhat from instance to instance because the specific backup routes impacted by the dual failure in question will differ from one dual failure to the next, and similarly for the other variables and for the constraints.

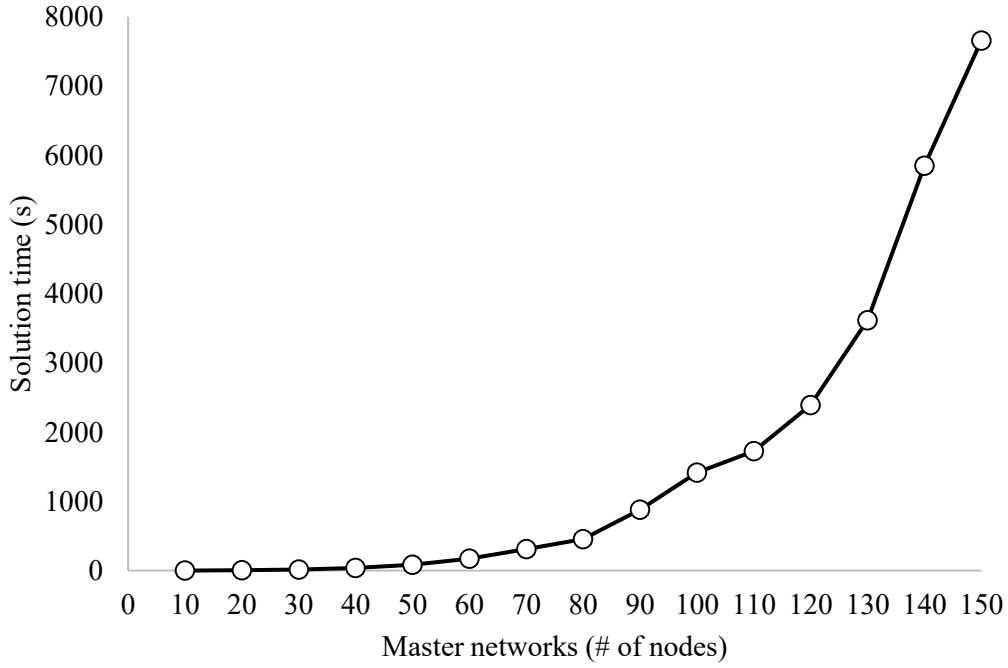


Figure 8.4 – Solution time for master networks

While a part of the scalability picture is painted by this data, solution time may be of more interest to some readers, as this will more directly address the model's performance directly. Figure 8.4 shows the total solution time of the algorithm on the master networks (i.e., the test case networks with average nodal degree of 5.0). As expected, solution time increases in an exponential-like manner with the increasing number of nodes in the network. Using the *Least Squares approach* [120], the best-fit third order polynomial approximation for solution time in seconds is shown in Eq. (8-5), where n is the number of nodes in the master network; the R^2 value is 0.9926.

$$t = 0.007n^3 - 0.9993n^2 + 46.609n - 538.86 \quad (8\text{-}5)$$

8.4.5 Impact of Spare Capacity on Network Availability

We now investigate how increasing spare capacity influences network availability.

Rather than provide the algorithm with the minimum-cost SBPP network designs that are fully single-failure survivable, we provide it with network designs that have a slightly greater amount of additional capacity. We produce these network designs by first using the minimum-capacity SBPP network design model from [77] and then increasing the spare capacity on each span by a specified percentage, rounding up to integer values. The experimental results are shown in Figure 8.5, where the x-axis represents the percentage increase in spare capacity, and the y-axis represents the associated network availability. The pattern is visually indistinguishable for test case networks with all connectivities we tested, so we show data for only the networks with average nodal degree of 4.0.

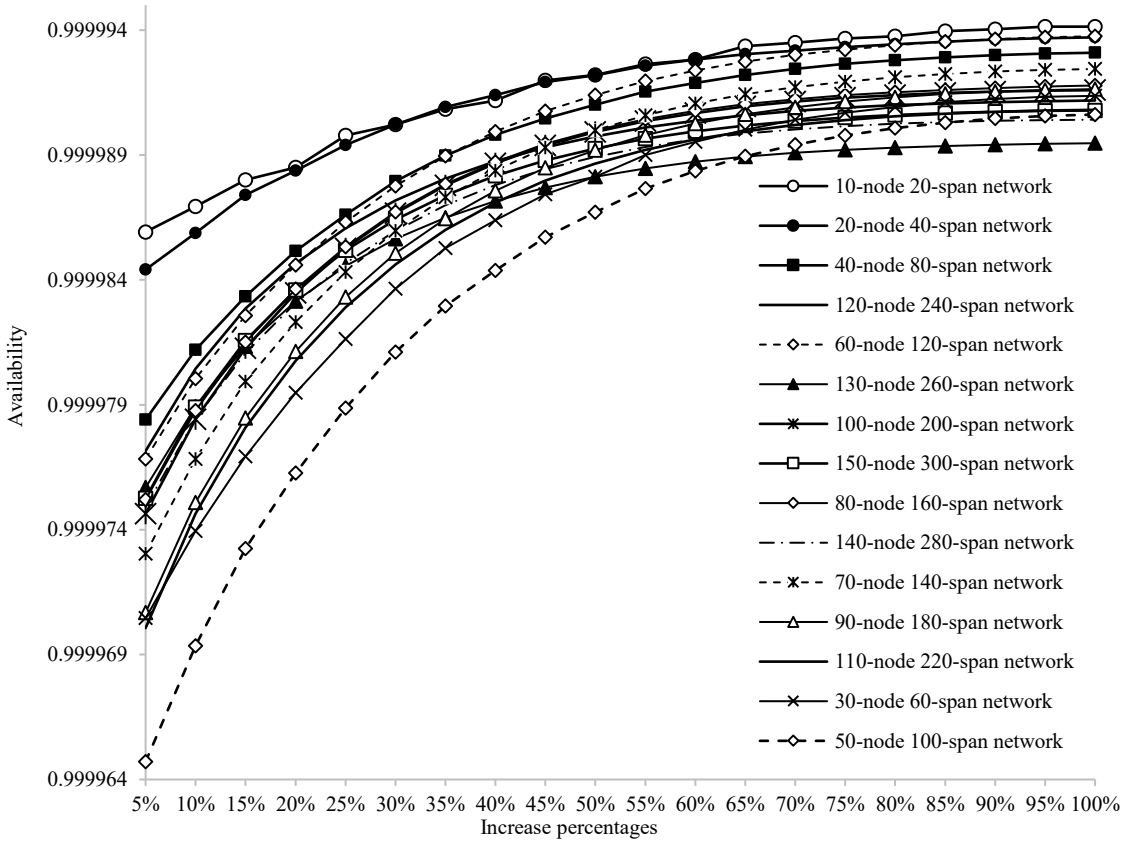


Figure 8.5 – Maximized network availability for test case networks with average nodal degree of 4.0, provided with additional spare capacity increases beyond the min-cost single-failure survivable design.

As the figure shows, all 15 test case networks reveal the same pattern on network availability as we increase network spare capacity by the percentages from 5% to 100%. Evidently, as network spare capacity increases, network availability increases accordingly. However, the declining slope of each line suggests that the availability improvements decline as the amount of spare capacity continues to increase. 100% dual-failure restorability is achievable, however, this will cost a disproportionate amount of additional spare capacity.

This data can be summarized more succinctly by calculating the average availability increment at each incremental percentage spare capacity increase (i.e., the average improvement in availability when increasing spare capacity from one specified percentage to the next higher percentage in 5% intervals), as in Figure 8.6. To be more precise, each data point represents the average amount (over all test case networks with average nodal degree of 4.0) that availability improves when increasing spare capacity in the network by an additional 5% to the total percentage increase indicated on the x-axis.

From Figure 8.6, it is evident that the average improvement in availability is high when the network is provided with small amounts of additional spare capacity (i.e., relative to that of the minimum cost design), but these improvements drop quickly as the spare capacity increases continue. For instance, if we increase the total amount of spare capacity by 5%, network availability can be enhanced by 5.0×10^{-6}, but if we increase spare capacity from 15% above the minimum capacity to 20% above the minimum capacity, we can only get an additional 2.0×10^{-6} increase. Our goal is to seek a balance point between availability increment and total cost, i.e., we want to achieve a desirable availability improvement at

an acceptable cost. There is no standard for such a balance point, but we observe an interesting behavior from the figure; when the percentage increase in spare capacity is greater than 40%, the average availability increment is lower by an order of magnitude than the previous 35% capacity increase. Therefore, we can suggest that in the test cases herein, a 35% increase in spare capacity provides a good tradeoff between capacity cost and availability improvement.

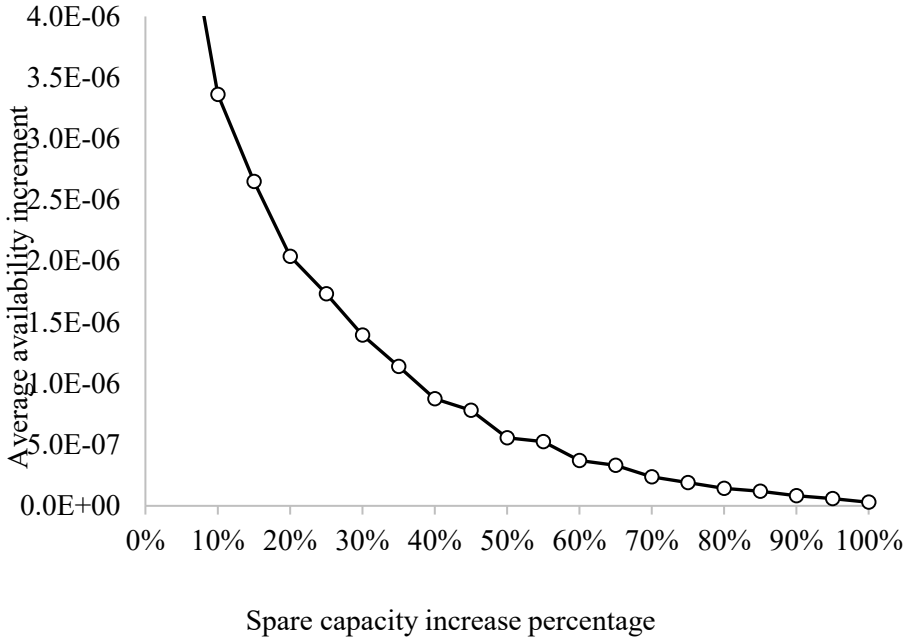


Figure 8.6 – Improvement in maximized network availability via the new algorithm for test case networks with average nodal degree of 4.0, provided with additional spare capacity increases beyond the minimum cost single-failure survivable design.

8.5 CONCLUSIONS

The SBPP survivable mechanism has received some attention in the literature in recent years, but there is limited prior work related to its relationship to network availability. Previous approaches rely on random selection of failed working routes when

dealing with dual-failures, which leads to sub-optimal (with respect to availability) designs. The algorithm proposed herein is designed to achieve an optimal SBPP network availability design. We demonstrate its scalability by applying it to test case networks as large as 150 nodes and 375 spans. We also show that as additional spare capacity is placed in a network (above and beyond that needed to provide single-failure survivability), overall network-wide availability initially improves considerably, but those improvements diminish as spare capacity increases continue. In the test case networks herein, increasing spare capacity by 35% balances capacity increases and availability improvements.

The approach we developed specifically addresses dual failures, as it is those scenarios that contribute the majority of a network's unavailability. However, higher order failures certainly occur, and so the reader may wish to extend our approach for such failures. In order to extend our algorithm to triple-failure scenarios, one further layer of iteration would be added (i.e., the appropriate for loop would be added within the loop on lines 11 through 14 of Figure 8.1), and the algorithm would need to cycle through all triple-failure scenarios (i.e., all span triplets). Although extending the algorithm would be fairly straightforward, we expect scalability to be an issue, particularly with larger networks, as the numbers of variables and constraints per instance of the ILP will increase (a great number of backup routes will be impacted by a triple-failure scenario) and the number of instances of the ILP increases from $|\mathrm{S}| \times (|S| - 1)$, to $|\mathrm{S}| \times (|S| - 1)(|S| - 2)$.

CHAPTER 9 DESIGN AND AVAILABILITY OPTIMIZATION OF PATH-RESTORABLE NETWORKS

9.1 MOTIVATIONS AND GOALS

In path restoration, end-to-end restoration routes for a failed lightpath will be specific to the location of the failed span along the lightpath, so path restoration is also called failure-dependent path protection (FDPP) [18]. In addition, path restoration typically also allows stub-release, where the surviving portions of the failed lightpath are released and the associated working capacity along the route is made available to use as spare capacity for restoration of any of the simultaneously failed lightpaths (in general, a failed span might carry lightpaths between a number of different end-to-end node pairs) [24]. As compared to span restoration, the replacement backup routes are distributed throughout a much wider range [24]. Path restoration is guaranteed to be at least as efficient as SBPP, and it receives a lot of attention as well [18].

Prior work addressing path-restorable network availability is limited, which we seek to address herein. More precisely, we will (1) propose and develop an algorithm to obtain the optimal network dual-failure availability for path-restorable networks, and (2) attempt to determine the nature of the relationship between network dual-failure availability and spare capacity.

9.2 DESIGN OF PATH-RESTORATION NETWORKS

9.2.1 NOTATION

The path-restorable network capacity allocation ILP model is a well-understood design model, which we will reproduce here from [18] for completeness and for ease of understanding of a new but related ILP network design model we will propose and develop later. The notation used in the model is defined as follows:

Sets:

S is the set of spans.

D is the set of demands.

P^r is the set of candidate primary routes for demand r.

B_i^r is the set of candidate backup routes for demand r under single failure i.

Parameters:

C_k is the cost of placing one unit of capacity on span k.

d^r is the quantity of demand r.

$\zeta_k^{p,r}$ equals 1 if the working route p of demand r crosses over span k, and equals 0 otherwise.

$\delta_k^{b,r}$ equals 1 if the backup route b of demand r crosses over span k, and equals 0 otherwise.

Variables:

w_k is the amount of working capacity placed on span k.

s_k is the amount of backup capacity placed on span k.

wf_p^r is the amount of working flow on working route p of demand r.

$bf_i^{b,p,r}$ is the amount of backup flow on backup route b for the restoration of working route p of demand r under single failure i.

$s_{k,i}^0$ is the units of spare capacity assigned to span k under single failure i due to stub-release.

9.2.2 ILP FORMULATION

Similarly to span restoration and SBPP networks, our capacity design model for path restoration networks is also based on ILP and the objective remains to produce a minimal capacity design. For research purpose, a practical way is to assume a static demand matrix [55]. Again, we adopt a JCA method. The basic approach is to assign both working and restoration flow to their respective set of eligible routes over the network. The objective function is:

Minimize:

$$Cost = \sum_{k \in S} C_k (w_k + s_k) \tag{9-1}$$

The constraints are:

$$\sum_{p \in P^r} wf_p^r = d^r \qquad \forall r \in D \tag{9-2}$$

$$w_k = \sum_{r \in D} \sum_{p \in P^r} \zeta_k^{p,r} \cdot wf_p^r \qquad \forall k \in S \tag{9-3}$$

$$\sum_{b \in B_i^r} bf_i^{b,p,r} \geq \zeta_i^{p,r} \cdot wf_p^r \qquad \forall i \in S, \forall r \in D, \forall p \in P^r \tag{9-4}$$

$$s_k \geq \sum_{r \in D} \sum_{b \in B_i^r} \delta_k^{b,r} \cdot bf_i^{b,r} - s_{k,i}^0 \qquad \forall i, k \in S | i \neq k \tag{9-5}$$

$$s_{k,i}^{0} = \sum_{r \in D} \sum_{p \in P^r} \zeta_k^{p,r} \cdot \zeta_i^{p,r} \cdot wf_p^r \qquad \forall i,k \in S | i \neq k \qquad (9\text{-}6)$$

The objective function in (9-1) seeks to minimize the total cost of working and spare capacity in the network. The constraints defined in (9-2) through (9-6) aim to deploy both working and spare capacity throughout the network in order to satisfy the required demands as well as guarantee full single-failure restorability. The constraints set in (9-2) guarantees that each demand r is satisfied by providing enough working lightpaths from all the candidate working routes. Constraints in (9-3) then calculate the number of working lightpaths on each span to determine each span's working capacity. Note that wavelength continuity is not modelled here, so although we use the term "lightpath", we assume opaque nodes, where full wavelength conversion is allowed (which is still the norm in many core transport networks). The set of constraints in (9-4) ensures that sufficient restoration paths are formed between any pair of end nodes to accommodate all lightpaths impacted by failure of span i. In much the same manner as in (9-3) for working lightpaths, the constraints in (9-4) calculate the maximum number of restored lightpaths simultaneously crossing each span in order to ensure sufficient spare capacity on each span. The constraints set in (9-6) calculates the stub-released capacity on each span k in the event of failure of span i, which are utilized on the right-hand side of the constraints in (9-5).

9.3 AVAILABILITY OPTIMIZATION ALGORITHM

As discussed, the ILP model above will produce a minimum-cost path-restorable network design that is fully survivable in the event of any single span failure. Although prior work has shown that such networks will have some inherent dual-failure restorability [105], there is no guarantee of any specific level of dual-failure restorability, and it most

certainly will fall short of full dual-failure restorability. As a consequence, there will inevitably be some level of unavailability in such a network arising from dual-failure scenarios, which have been shown to be the source of most unavailability in a network [26]. Precisely how much unavailability is still an open question.

In order to answer this question, we develop an algorithm to obtain optimal network dual-failure availability in a path-restorable network. Here, we define dual-failure availability as the availability of the network if we consider only outages arising from dual failures. We assert that this is a reasonable approximation of network availability where single-failure survivability is assured and where higher order failures (i.e., those involving simultaneous failure of three or more spans) are so exceedingly rare that their contribution to a network's unavailability is many orders of magnitude smaller than the contribution from dual-failure scenarios [119].

The input of this algorithm, illustrated in Figure 9.1, is essentially the design data from the path-restorable network capacity allocation model above, including the set of all the spans in the network (denoted by S), the set of all the eligible working routes (denoted by P), the set of all eligible backup routes (denoted by $B^p, \forall\, p \in P$), the amount of backup capacity allocated to each span (denoted by $s_k, \forall k \in S$), the number of working lightpaths on each eligible working route (denoted by $wf^p, \forall p \in P$), the number of backup lightpaths on each eligible backup route (denoted by $bf^{b,p}, \forall b \in B^p, p \in P$), and the vector that indicates whether a specified working route p traverses span k (denoted by $\zeta^{k,p}, \forall k \in S, p \in P$).

In the first step, we cycle through all the dual-failure scenarios (i.e., dual-failure $(i,j), \forall i,j \in S | i \neq j$) to obtain the dual-failure restorability for each dual-failure scenario (i.e., $R_2(i,j), \forall i,j \in S | i \neq j$). Recall that $wf_{\text{aff}}(i,j)$ and $wf_{\text{lost}}(i,j)$ represent the number of working lightpaths affected and lost due to this dual-failure, respectively. Apparently, at the beginning of each iteration, both of their values are initialized at zero. We then examine each working route that is affected by the first failure i, add its working lightpaths to the value of $wf_{\text{aff}}(i,j)$, and update the spare capacity for the spans on the backup routes of working route p. Meanwhile, for the surviving lightpaths on working route p, we update the spare capacity for the corresponding spans by allowing stub-release. In the following step, we examine each backup route b of the primary route p to check whether this backup route is affected by the second failure j. If it is affected by the second failure j, the value of $wf_{\text{lost}}(i,j)$ will be updated by adding up the backup lightpaths on this examined backup route. The procedure is continued until all the working routes affected by the first failure, i, are considered. Subsequently, we examine all the working routes that are affected only by the second failure, j, i.e., we will exclude those that were already affected by the first failure, i. All such working routes are denoted by the set of P' as indicated in the figure. Likewise, we add all the working lightpaths to the value of $wf_{\text{aff}}(i,j)$ to update the count of affected working lightpaths. The number of lost working lightpaths is difficult to determine, because the input design results considered only single-failure scenarios, and so we do not have any specific restoration strategy for the second failure. For this reason, we develop an ILP model to do this for us.

For now, we will simply state that the ILP seeks to minimize the wflost in the network, but will describe it in full in the following section. After solving the ILP model, we

calculate the value of $R_2(i,j)$ for the current dual-failure scenario. Once all the dual-failure scenarios have been addressed, we calculate A_2^p for each working route. Finally, the network dual-failure availability A_2 is obtained via the relevant equations as in (4-10) and (4-11) from Chapter 4.

```
01    Begin
02    Input S, P, B^p, s_k, wf^p, bf^{b,p}, and ζ^{k,p} for k∈S, p∈P, b∈ B^p
03    For dual-failure(i,j), i,j∈S and i≠j
04          Initialize wf_aff(i,j)=wf_lost(i,j)=0
05          For each working route p affected by failure i
06                Add its working flow to wf_aff(i,j)
07                Update spare capacity for span i
08                For each backup route b of p
09                      Update spare capacity for spans
10                End for
11                For each span on p except span i
12                      Reuse its working capacity
13                End for
14                For backup route b of p affected by failure j
15                      Add its backup flow to wf_lost(i,j)
16                      Update spare capacity on span j
17                End for
18          End for
19          For working route p affected by failure j but not i
20                Add its working flow to wf_aff(i,j)
21          End for
22          Minimize total lost flow caused only by failure j
23          Calculate total lost flow under dual-failure (i,j)
24          Calculate restorability under dual-failure (i,j)
25    End for
26    Calculate network availability
27    End
```

Figure 9.1 – Pseudocode of availability optimization algorithm for path restoration

9.4 TOTAL LOST WORKING FLOW OPTIMIZATION

Since the first failure of any dual-failure scenario is fully restored, the dual-failure availability is essentially determined by the amount of restoration realized by the second failure (though there will often be some loss of the first failure's restoration lightpaths in the event that some of them crossed the second failed span). Accordingly, in order to optimize dual-failure availability, we can minimize the total number of lost working lightpaths on the working routes that are affected by the second failure, excluding those already affected by the first failure. Since failed working routes will compete for available spare capacity during the ensuing restoration process, the restoration sequence of the failed working routes important to consider; in order to obtain the optimal dual-failure availability, we cannot restore the failed working routes randomly. As a consequence, we needed to develop a new ILP model to determine the optimal sequence for the restoration process, as stated above in the discussion of the algorithm in Figure 9.1. The assumptions we apply in this optimization process include:

(1) The restoration of the first failure is completed before this optimization. That is, we only consider the second failure in this process. Since only one failure is involved, we can remove the failure subscript j of the second failure from related sets, parameters, and variables in the model, although the optimization process is designed for a specified dual-failure scenario. We will still specify the second failure using the parameter j where applicable. In this case, j is a parameter rather than an index variable.

(2) After obtaining input data from the design process, the optimization process is independent from the design process. Therefore, we redefine symbols in this process. In

some cases, even though we use the same symbol as the design process, we may refer to different meanings.

(3) Once the second failure occurs, the pre-defined backup routes, which are designed originally for the restoration of the first failure, are activated to restore the failed working flow caused by the second failure. It should be noted that the amount of backup flow and the backup capacity on each span are not known for the restoration of the second failure, which is different from the restoration of the first failure. We will get these values by maximizing the network availability in order to make the best of available backup capacity already assigned to each span.

9.4.1 ANNOTATIONS

The new ILP model utilizes the following new notation.

Sets:

P' is the set of all working routes that only affected by the second failure in a specified dual-failure scenario.

B^p is the set of all backup routes of the working route p.

AB is the set of all backup routes for the restoration of all working routes specified in the set of P'.

S^b is the set of all spans on the backup route b.

S' is the set of all spans on all the backup routes in AB.

CB^k the concurrent backup routes, i.e., it is the set of all backup routes that traverse span k simultaneously.

Parameters:

i is the parameter denoting the first failure in a specified dual failure scenario.

j is the parameter denoting the second failure in a specified dual failure scenario.

wf^p is the amount of working flow on the working route p.

s_k is the amount of backup capacity on the span k obtained from the design.

$s_k{}'$ is the amount of backup capacity on the span k considering stub-release due to the second failure.

$\zeta^{k,p}$ is a binary parameter, being 1 if the working route p crosses span k and 0 otherwise.

Variables:

wf_{lost}^p is an integer variable, representing the amount of lost working flow on the working route p caused by the second failure in a specified dual-failure scenario.

$bf^{b,p}$ is an integer variable, representing the amount of backup flows on the backup route b for restoration of the working route p under the second failure in a specified dual-failure scenario.

9.4.2 ILP FORMULATION

The objective is to minimize the total lost working flow on the working routes that are affected by the second failure excluding those affected by the first failure in a specified dual failure scenario.

Minimize:

$$wf_{lost} = \sum_{p \in P'} wf_{lost}^{p} \tag{9-7}$$

The constraints are:

$$wf_{lost}^{p} \geq 0 \qquad \forall p \in P' \tag{9-8}$$

$$wf_{lost}^{p} \geq wf^{p} - \sum_{b \in B^{p} | i \notin S^{b}, j \notin S^{b}} bf_{b}^{p\,\prime} \qquad \forall p \in P' \tag{9-9}$$

$$bf_{b}^{p} \leq s_{k}', \qquad \forall p \in P', b \in B^{p} | i \notin S^{b}, j \notin S^{b}, k \in S^{b} \tag{9-10}$$

$$\sum_{p \in P'} \sum_{b \in CB^{k} \cap B^{p} | i \notin S^{b}, j \notin S^{b}} bf_{b}^{p} \leq s_{k}' \qquad \forall k \in S' \tag{9-11}$$

$$s_{k}' = s_{k} + \sum_{p \in P'} \zeta^{k,p} \cdot \zeta_{j}^{p} \cdot wf^{p} \qquad \forall k \in S' | k \neq j \tag{9-12}$$

The objective function in (9-7) seeks to minimize the total lost working lightpaths on the working routes that are affected by the second failure, excluding those affected by the first failure in a specified dual-failure scenario.

For a given working route p, the total backup flow that can be used to restore it once the second failure occurs is the sum of the available backup flow on all its backup routes. This sum is either less than the failed working flow, as per equation (9-8), or no less than the failed working flow, as per equation (9-9). If the sum is less than the failed working

flow, then the lost working flow on the working route is the difference between the failed working flow and the sum; while if the sum is no less than the failed working flow, then the lost working flow is zero, i.e., there is no lost working flow.

The constraints in (9-10) ensure that the maximum backup flow on a specified backup route b cannot exceed the minimum actual spare capacity of the spans on it. We guarantee this condition by forcing the backup flow to be less than the actual spare capacity of each span on it.

Regarding the constraints in equation (9-11), first recall that the set P' represents all the primary routes that are only affected by the second failure in a specified dual-failure scenario. In order to restore the working flow interrupted by the second failure, all the backup routes, which are defined in the set AB, of all the primary routes defined in P' are activated simultaneously. For any span k in the set S', if all the backup routes in any subset of the set AB traverse span k, we call them concurrent backup routes with respect to span k, and denote this subset with CB^k. The sum of backup flow on the concurrent backup routes cannot surpass the amount of available backup capacity on span k as shown in equation (9-11).

The constraints in (9-12) accounts for the stub-release effect. As a result of stub-release, the amount of actual spare capacity on a specified span k varies under different failure scenarios. More specifically, if the failed working route does not cross over span k, its available spare capacity is the same as obtained from the design process, while if the failed working route traverses span k and span k is not the failed span, then the working capacity on span k can be released and used as the backup capacity.

9.4.3 INTEGRATION OF GUROBI AND PYTHON

The method of calling Gurobi within Python is the same as sated in Section 7.3.3. We will not discuss it here to avoid repetitiveness. Please refer to Section 7.3.3 for details if needed.

9.5 EXPERIMENTS

9.5.1 EXPERIMENTAL NETWORKS AND SETUP

In this section, we seek to investigate (1) how network availability of span-restorable networks responses as network average nodal degree increases, (2) how spare capacity increases influences span-restorable network's availability. From previous studies, we observe that patterns among different network families are similar, so in order to save time, we use 30-node family, 50-node family, 60-node family, and 70-node family, instead of all the network families. We use the experimental setup as indicated in Section 3.5.3 as well and call Gurobi through Python in each loop wherever an ILP model is involved.

9.5.2 IMPACT OF NETWORK AVERAGE NODAL DEGREE ON NETWORK AVAILABILITY

Figure 9.2 through Figure 9.5 show the changing trend of network availability with increasing network average nodal degree for 30-node family, 50-node family, 60-node family, and 70-node family, respectively. Each data point represents the dual-failure availability of the network of the indicated average nodal degree in the indicated network family, as calculated by our algorithm described above.

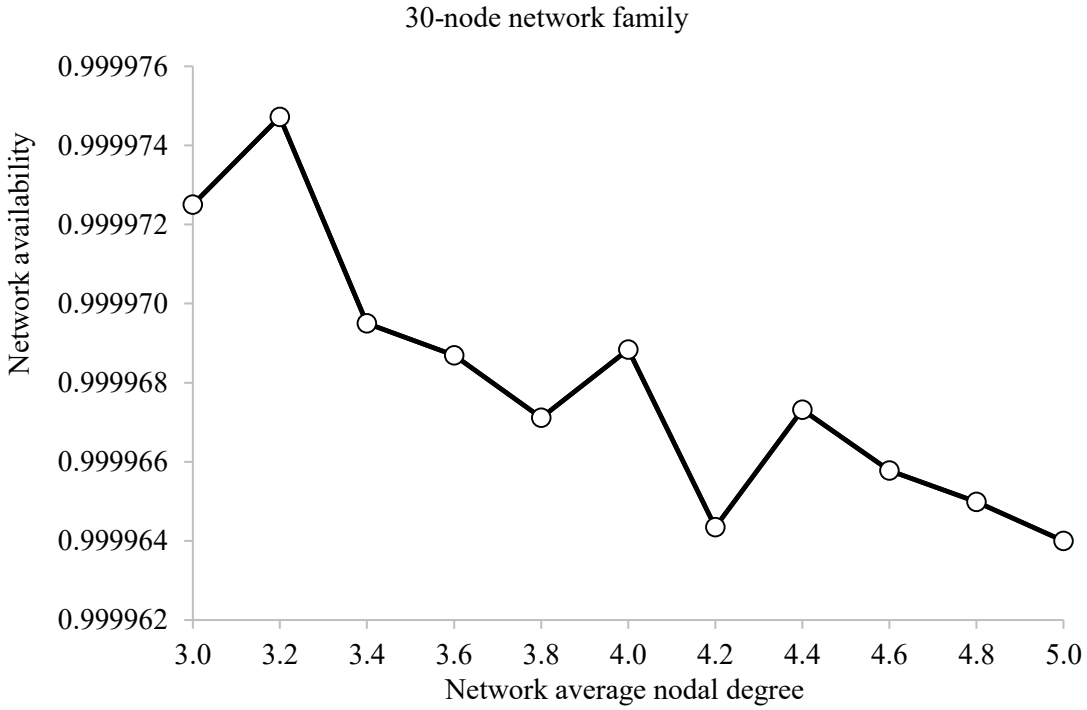


Figure 9.2 – Network availability for 30-node network family

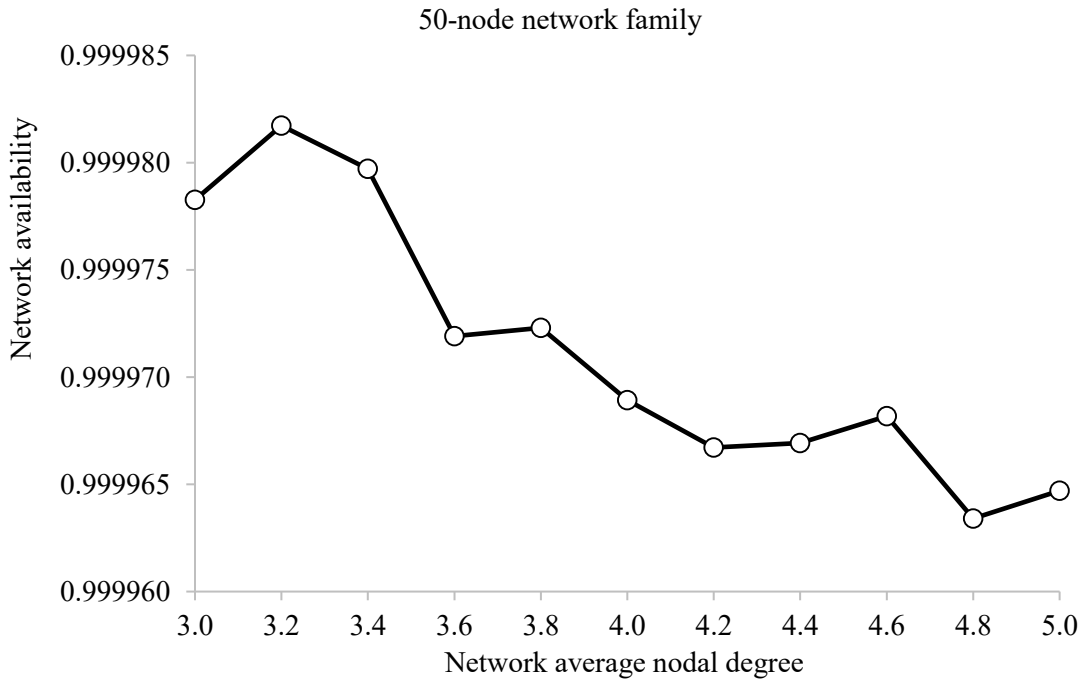


Figure 9.3 – Network availability for 50-node network family

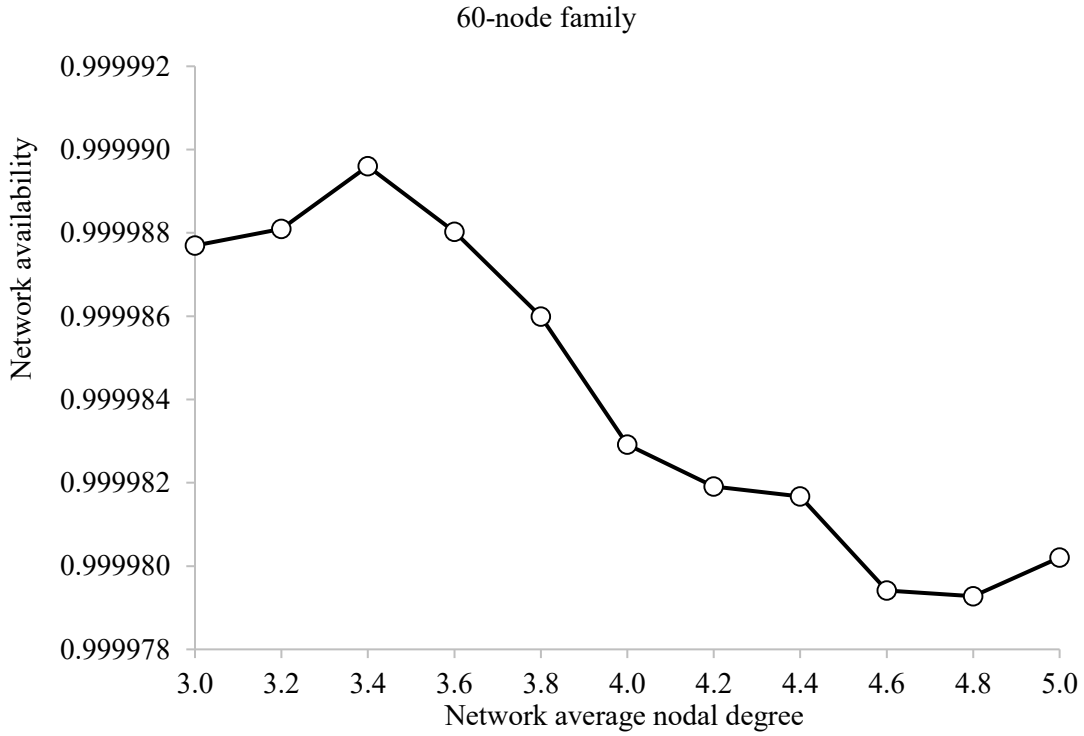


Figure 9.4 – Network availability for 60-node network family

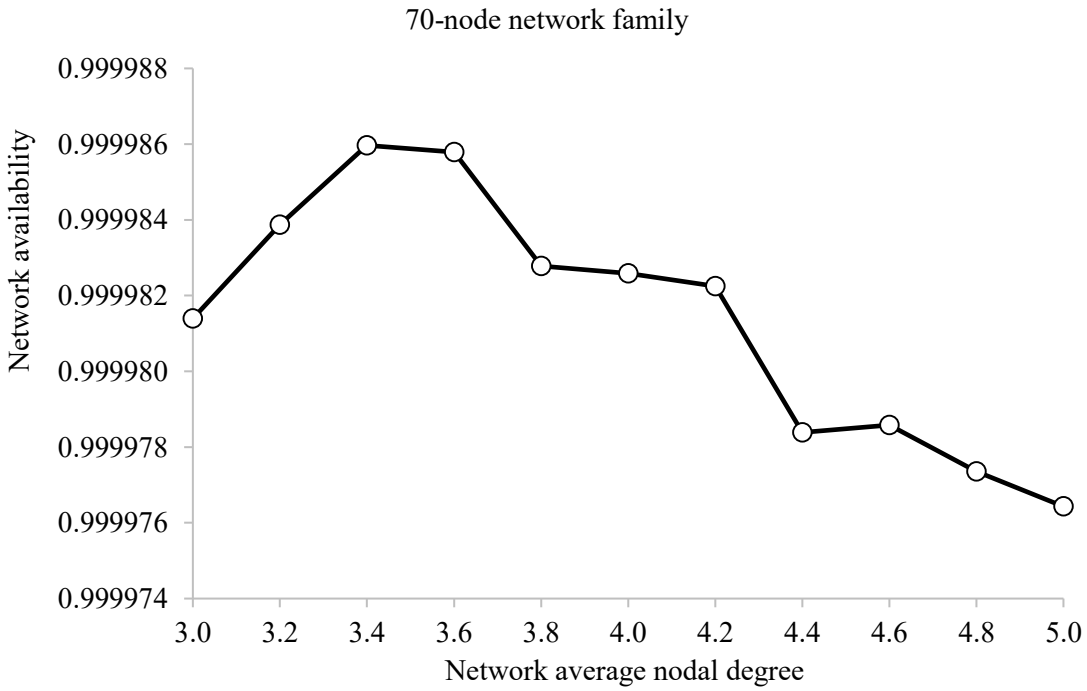


Figure 9.5 – Network availability for 70-node network family

From these figures, we observe counter intuitively that although network availability initially increases when moving to higher network connectivity, it then generally decreases with further increases in connectivity, though not monotonically. One may expect that as the network connectivity increases, creating a greater diversity of eligible routes for both working and restoration routing, availability would only ever increase. However, recall that availability has been calculated for networks that have been designed to be of minimum cost, which can have an unpredictable effect on the subsequent availability. More precisely, the networks were designed for minimum cost, not maximum availability.

Two factors appear to work together to drive the observations we've noted. On one hand, a higher connectivity produces a greater diversity of shorter eligible routes, which in turn reduces the total number of dual-failure scenarios. In this regard, the likelihood of network outage becomes smaller, increasing availability. But on the other hand, shorter working routes will lead to a much smaller amount of spare capacity in the network, as well as a smaller amount of reused working capacity during the stub-release process. This has a negative effect in terms of network availability. In most cases in our test case networks, the impact of the second factor prevails over the first when moving to the next more richly connected network, thereby resulting in a decrease in dual-failure availability.

9.5.3 Impact of Spare Capacity on Network Availability

Also of interest is how network availability will respond to increases of the total amount of spare capacity placed on the network. Each data point in Figure 9.6 represents the dual-failure availability of the network with the indicated spare capacity increase. Here, the initial network design is determined via the path-restorable network capacity allocation

ILP model above, and then the spare capacity on each span is increased by the percentage specified (and rounded up to the nearest integer). We provide data for the test case networks with average nodal degree of 4.0 (i.e., the 60-span member from the 30-node network family, the 100-span member from the 50-node network family, the 120-span member from the 60-node network family, and the 140-span member from the 70-node network family), and provide spare capacity increases in 5% increments up to 100%. We can observe that network availability generally improves with additional spare capacity, but improvements become smaller as the spare capacity increases become larger. The declining slope of each curve suggests that the network availability improvements have a downward trend with the continuously increase of spare capacity. Inexpensive improvements in availability can be had, but become increasingly costly. And although 100% dual-failure restorability might be achievable, this will cost a disproportionate amount of additional spare capacity.

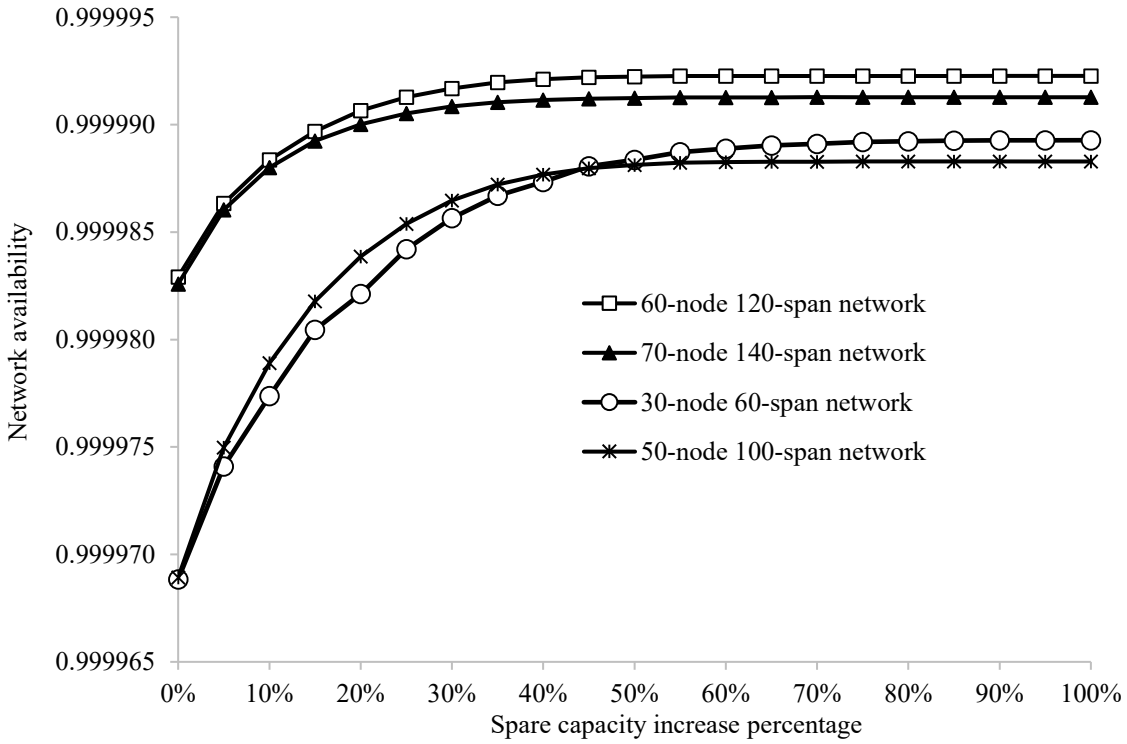


Figure 9.6 – Network availability for experimental networks

If we wish to look strictly at the improvements in availability, we can plot those improvements a well, as shown in Figure 9.7 through Figure 9.10. As these figures illustrate, the availability increment declines quickly with initial increases in spare capacity, and levels off near zero as the spare capacity increases continue, particularly in the larger network.

For example, if the total amount of spare capacity is increased by 5%, network availability is improved by 5.2×10^{-6} in the 30-node 60-span network and 6.1×10^{-6} in the 50-node 100-span network. However, as spare capacity continues to increase, say, when moving from 35% to 40% above the optimal single-failure design, we can only get an additional 6.3×10^{-7} increase for the 30-node 60-span network and a 7.5×10^{-7} increase for the 50-node 100-span network. In order to achieve a desirable availability improvement at an acceptable cost, it is important to seek some kind of balance point. While we could easily suggest some target, there is no clear inflection point observed, and each operator will have their own interpretation of the tradeoff between improving availability and increasing cost.

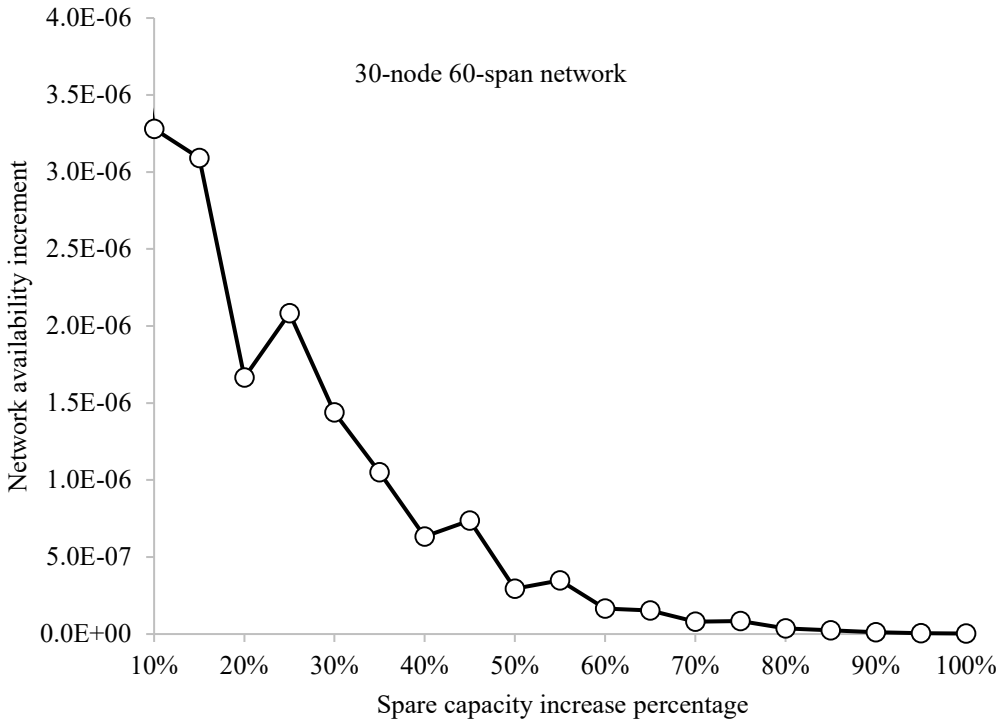


Figure 9.7 – Network availability increment for 30-node 60-span network

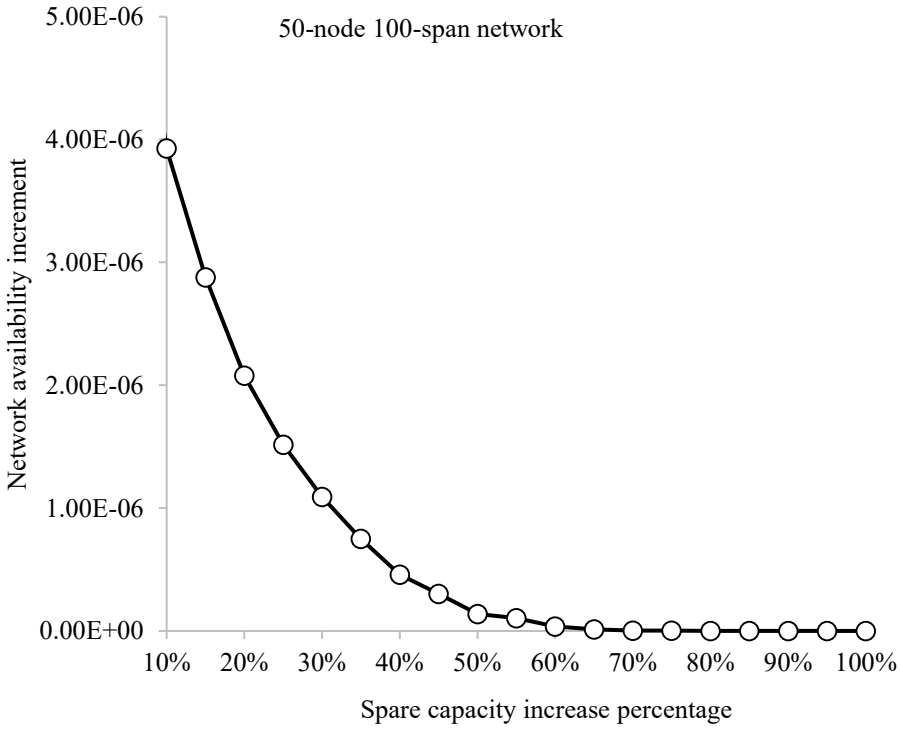


Figure 9.8 – Network availability increment for 50-node 100-span network

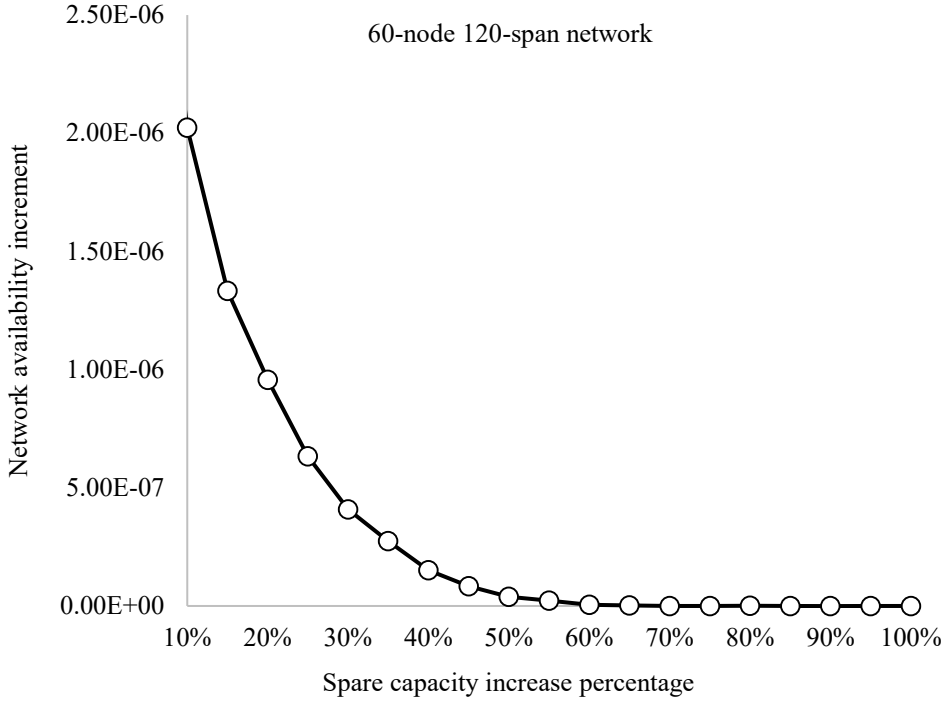


Figure 9.9 – Network availability increment for 60-node 120-span network

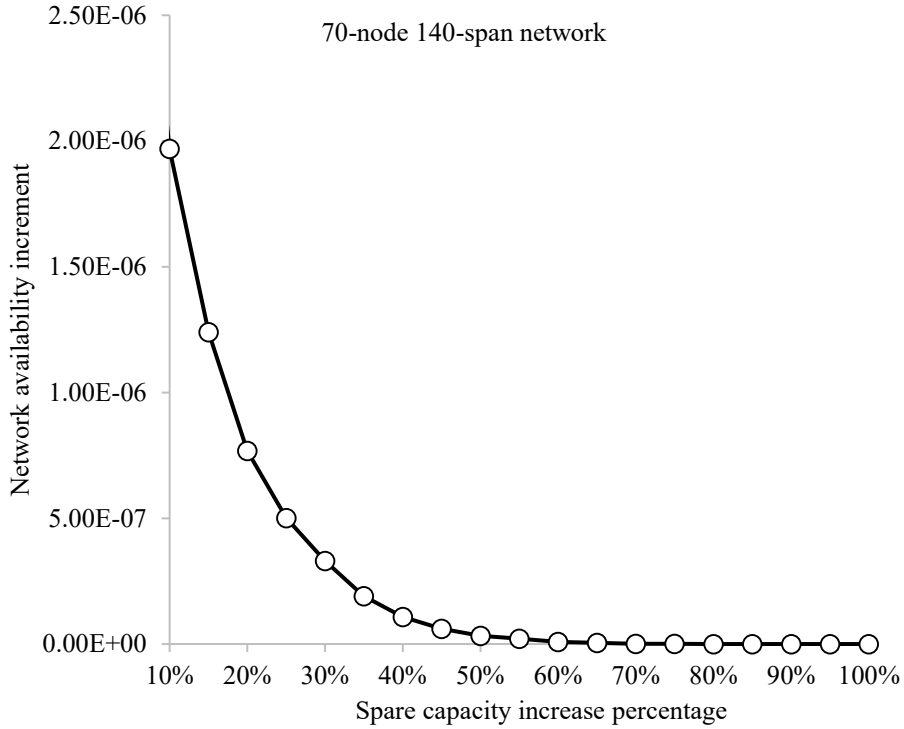


Figure 9.10 – Network availability increment for 70-node 140-span network

9.6 CONCLUSIONS

We have proposed and developed a new algorithmic approach and constituent ILP model to calculate dual-failure availability in path-restorable networks. Experiments show that counter intuitively, network availability generally decreases as network connectivity increases. Furthermore, substantial improvements in network availability can be obtained by adding small amounts of additional spare capacity in a network.

CHAPTER 10 PERFORMANCE COMPARISON OF VARIOUS MESH NETWORKS

From Section 6 through Section 9, we have investigated various survivability mechanisms individually. In this section, we will combine the above algorithms and compare the performance of these mechanisms in terms of network availability.

10.1 MOTIVATIONS AND GOALS

As stated in the previous sections, the study regarding network availability optimization is very few, and to the best of our knowledge, there is no comparison of network optimal availability among various survivability mechanism. In order to fill this research gap, we seek to compare network optimal availability for span restorable networks, path restorable networks, and SBPP networks.

10.2 METHODOLOGY

Although the optimal network availability for these three types of networks have been investigated in the previous sections, respectively, the results from previous sections cannot be used directly for comparison, due to two reasons.

(1) Experimental assumptions are not exactly the same. Specifically, we assume the physical failure rate is the same for each span in span restorable networks, but we utilize span-specific failure rate for each span in path restorable and SBPP networks. In this section in order to keep consistency, we use span-specific failure rate for each span for all types of networks.

(2) Experimental networks are not exactly the same. In span restorable networks, we use both existing networks and the newly created networks for experiments, while in path restorable and SBPP networks, we only use the newly created networks for experiments.

(3) we focus on how network availability indicators behave with increasing non-restored working capacity in the network for span restorable networks, and we do not investigate how optimal network availability responses in terms of network average nodal degree, as we do for path restorable and SBPP networks. As such, we will design experiments to obtain the same type of data for span restorable networks such that we can compare them with path restorable and SBPP networks.

The basic theory behind each algorithm is that we minimize the total lost working flow to maximize the value of network overall dual-failure availability under the condition that the network has been designed for full single-failure restorability already. Although the detailed procedures for span restorable, path restorable, and SBPP networks are different, the general procedure of the algorithm can be summarized in Figure 10.1. More specifically, for starters, we create network family (i.e., creation of *.top files and *.dem files for each network in the created network families) with the help of Inkscape and Python programs. Next, we build ILP design model for networks with a specified survivability mechanism using AMPL program (i.e., the buildup of AMPL *.mod file). Then the input data file (i.e., the *.dat file) is prepared through programming with Python program. After that, the ILP design model is solved within AMPL by calling Gurobi as the solver. In the next step, the total lost working flow under dual-failure scenarios is minimized by using AMPL (for span restorable networks) or Gurobi, and Python (for path restorable and SBPP networks). As such, dual-failure availability is maximized and we end up with optimal

dual-failure availability for the entire network. Results are analyzed and comparisons are made among different types of networks.

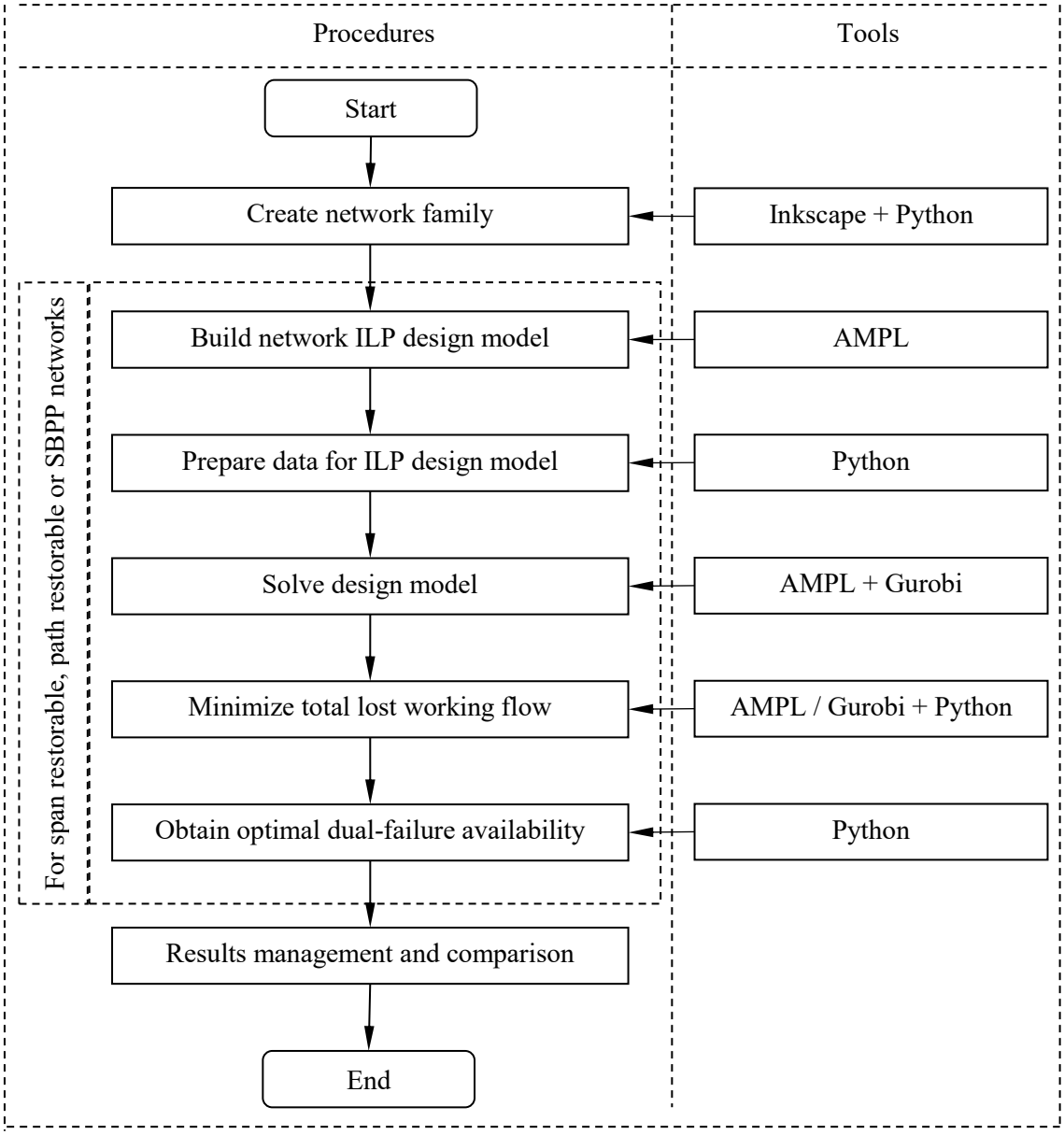


Figure 10.1 – Flowchart of methodology

10.3 EXPERIMENTS

We employ 30-node, 40-node, 50-node, and 60-node network families for experiments. See Chapter 5 for details regarding experimental networks and setup. The results for each network family are shown in Figure 10.2 through Figure 10.5.

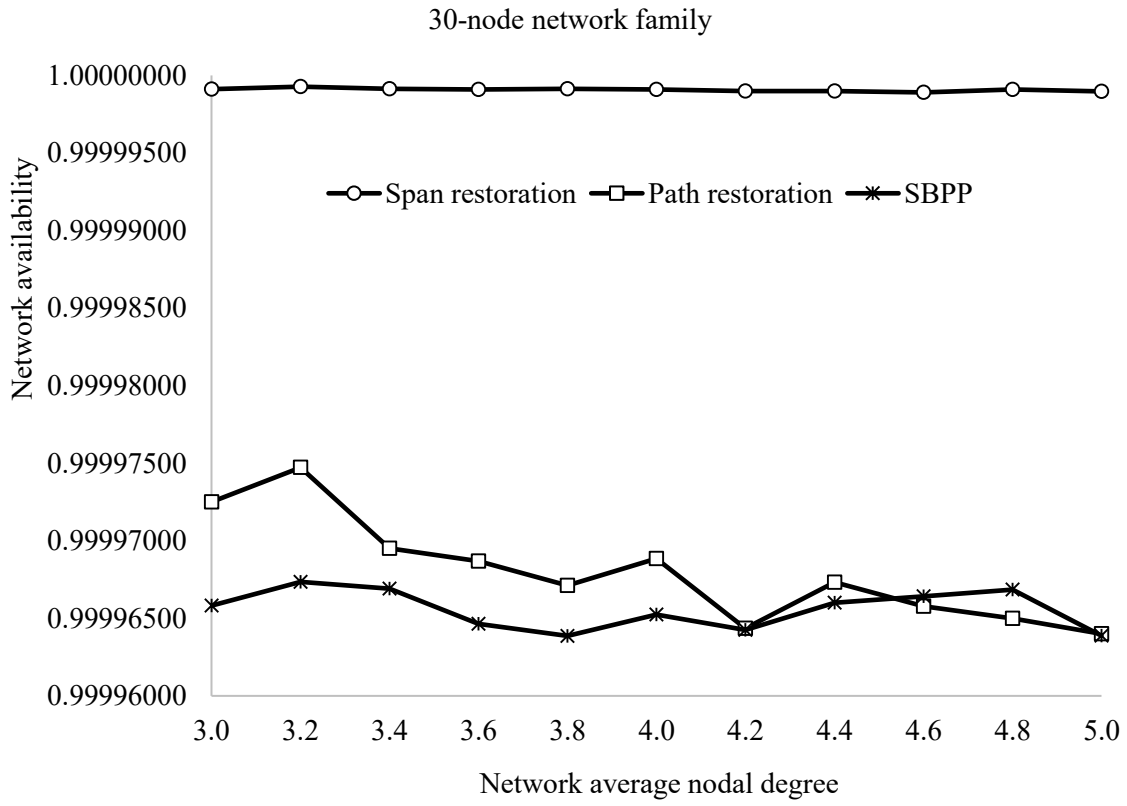


Figure 10.2 – Network availability for 30-node network family

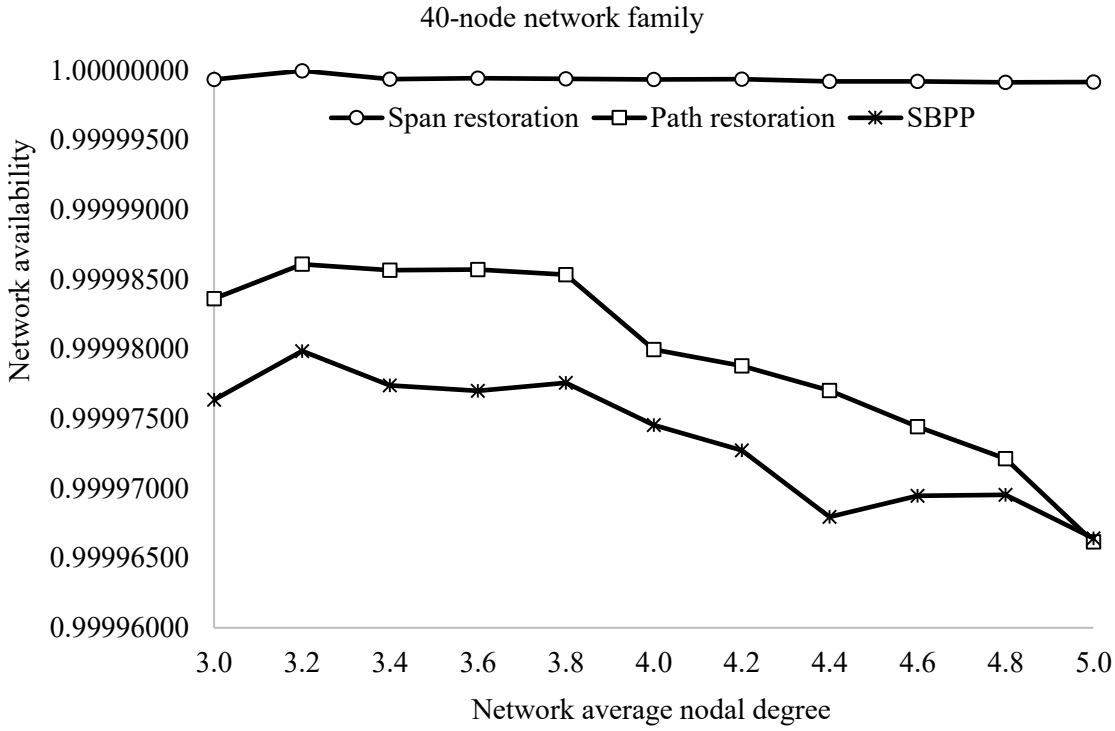


Figure 10.3 – Network availability for 40-node network family

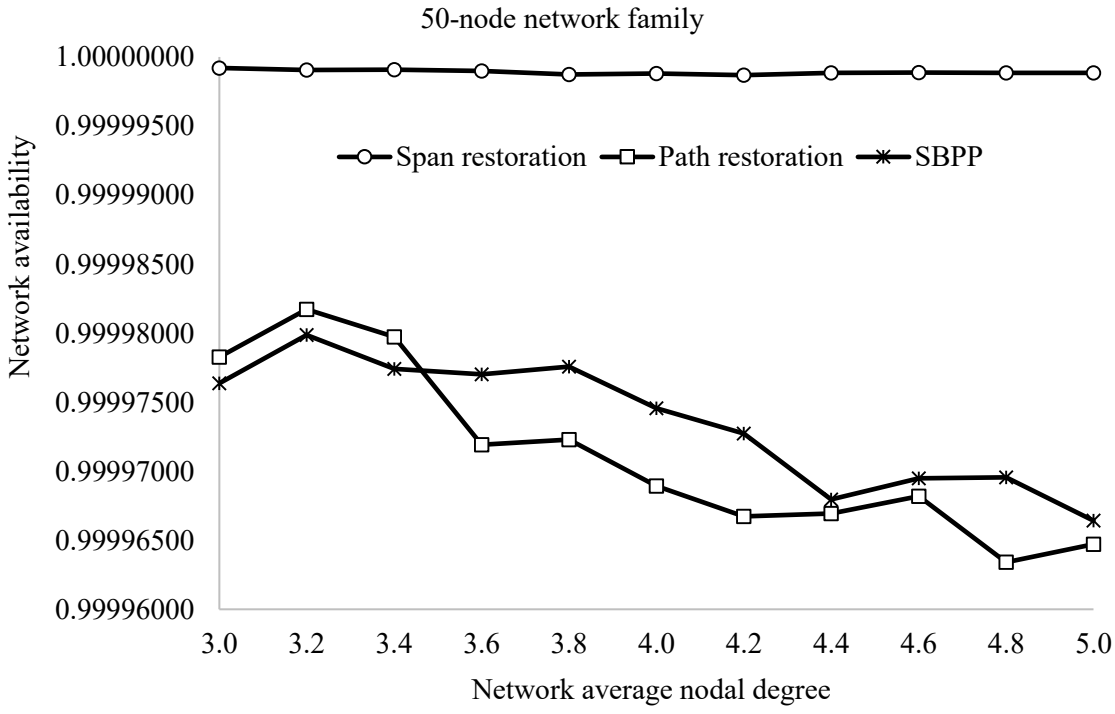


Figure 10.4 – Network availability for 50-node network family

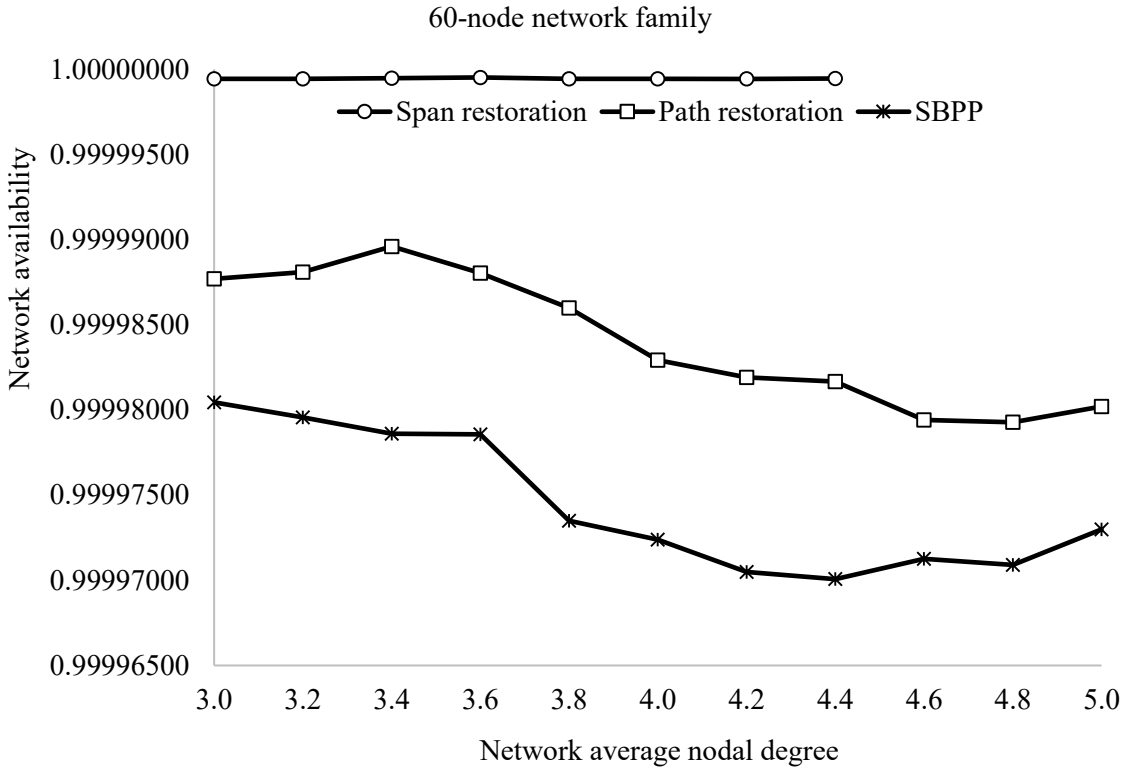


Figure 10.5 – Network availability for 60-node network family

Three observations from the figures are: (1) span-restorable networks have higher overall availability compared to path-restorable networks and SBPP networks in terms of the number of nines, (2) path-restorable networks and SBPP networks have similar overall network availability in terms of the number of nines, and (3) as network average nodal degree increases, the values of network overall availability of path-restorable networks and SBPP networks have similar changing trend. The third observation has been discussed thoroughly in the previous chapters. With regard to the first two observations, our hypothesis is that under the design of full single-failure restorability, span-restorable networks have the largest amount of spare capacity, while path-restorable networks and SBPP networks have a smaller but similar amount of spare capacity. As a result, span-restorable networks tend to have largest overall network availability, whereas path-

restorable networks and SBPP networks tend to have smaller and similar overall network availability generally.

In order to investigate the trend of overall network availability under the three survivability mechanisms on the same figure, we plot network availability for path-restorable and SBPP networks on the left axis and span-restorable networks on the right axis, as shown in Figure 10.6 through Figure 10.9.

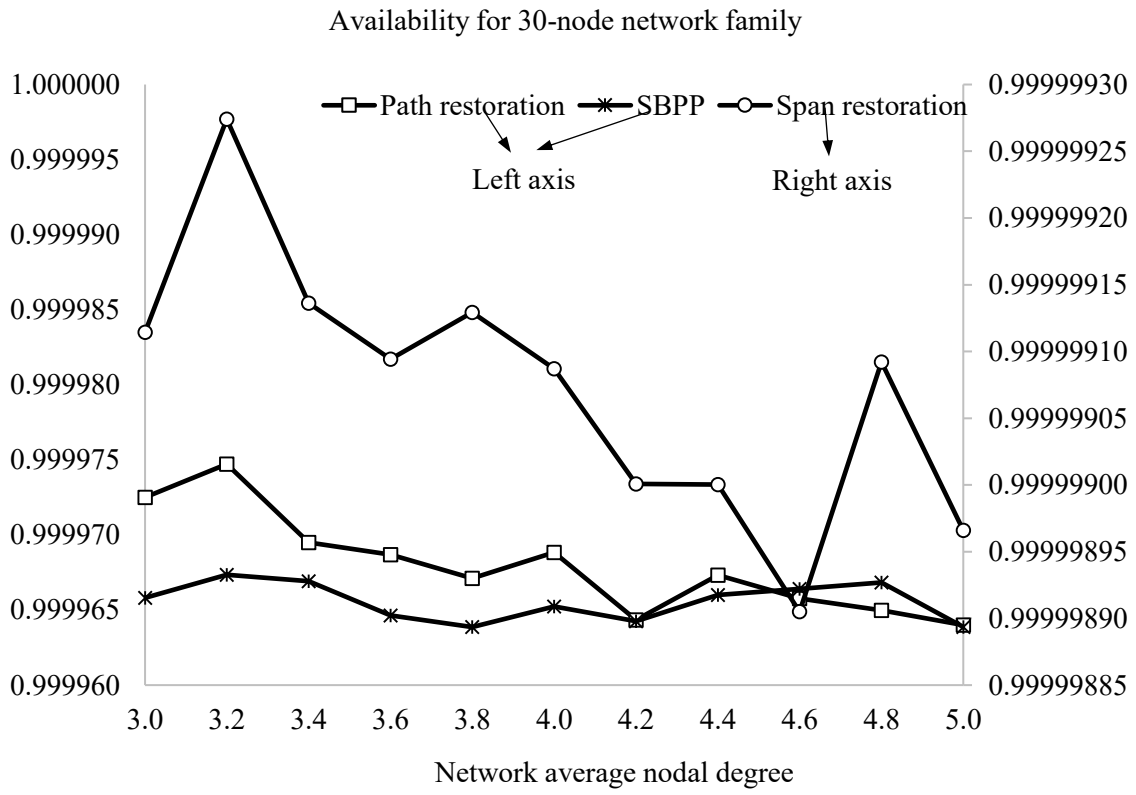


Figure 10.6 – Network availability for 30-node network family with two y-axes

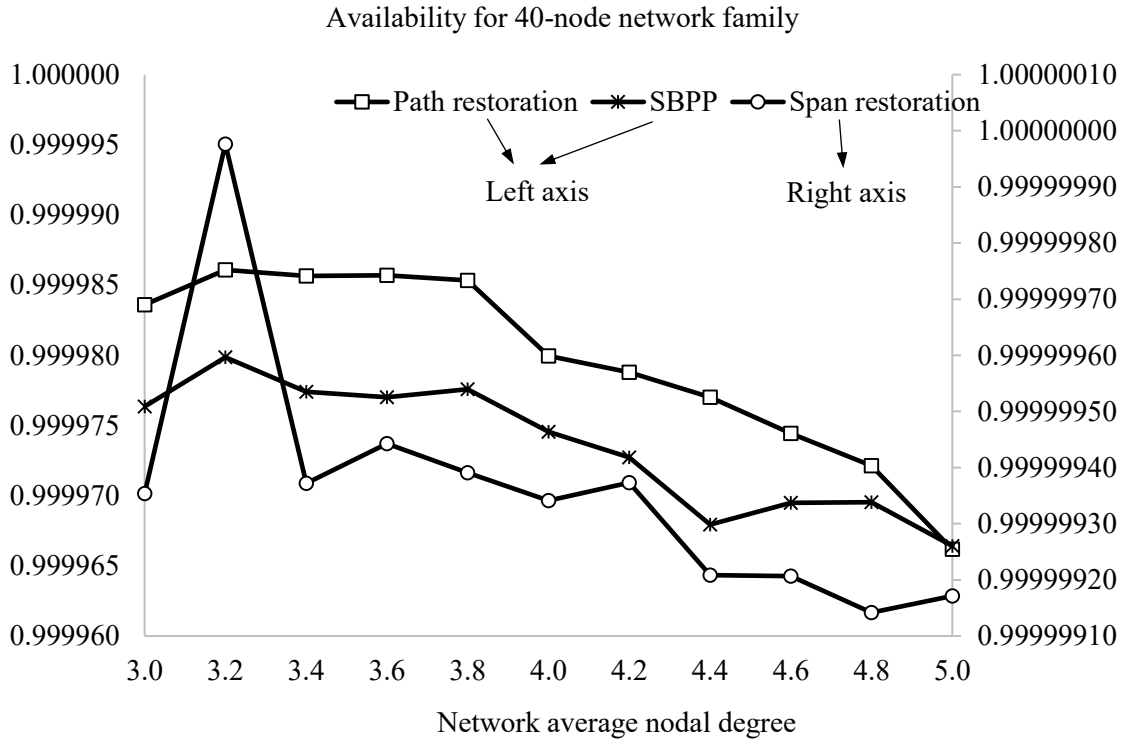


Figure 10.7 – Network availability for 40-node network family with two y-axes

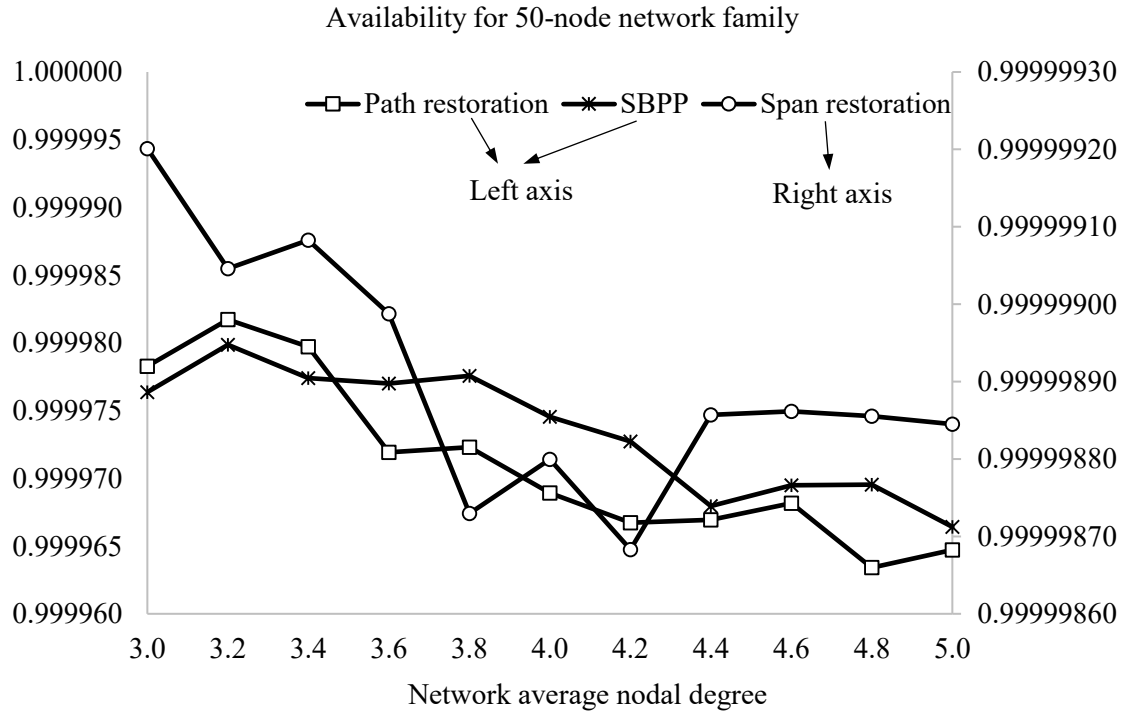


Figure 10.8 – Network availability for 50-node network family with two y-axes

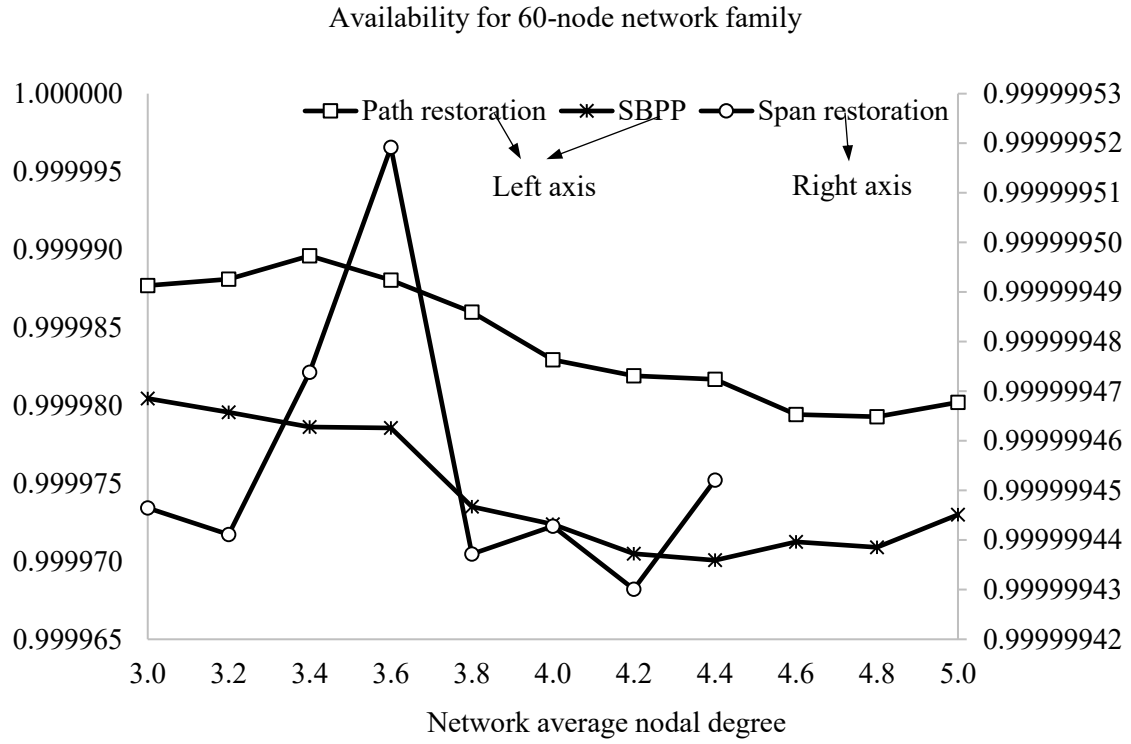


Figure 10.9 – Network availability for 60-node network family with two y-axes

It reveals from the figures that overall network availability under the three types of survivability mechanisms has similar pattern with increasing network availability, i.e., overall network availability tends to increase at the very beginning and then decreases later on. The reasons have been discussed and explained in the previous chapters, so we will not go with details here.

In order to prove our hypothesis regarding Observation (3), we use 30-node 45-span network to perform a case study as follows.

For span restoration and SBPP, the total spare capacity listed in Table 10.1 is the actual available spare capacity on each span to protect network from dual failures. However, for path-restorable networks, hidden spare capacity exists due to stub-release. As previously stated in Chapter 9, stub-release is allowed in both network design and

availability optimization processes. Accordingly, an amount of working capacity on failed working routes is reused as spare capacity in these two processes. The amount of hidden spare capacity due to stub-release under single-failure scenario S01 in design process is listed in Table 10.2 and the full set of values under all single-failure scenarios is listed in Appendix A.1.

Table 10.1 – Spare capacity for 30-node 45-span network

spans	Amount of spare capacity			spans	Amount of spare capacity		
	SR	PR	SBPP		SR	PR	SBPP
S01	81	73	107	S24	53	78	130
S02	47	52	83	S25	177	45	79
S03	47	11	38	S26	299	119	93
S04	152	172	136	S27	291	103	104
S05	144	23	56	S28	253	100	88
S06	112	0	34	S29	255	17	40
S07	16	52	75	S30	302	46	80
S08	88	143	224	S31	257	64	72
S09	144	51	79	S32	160	38	74
S10	474	122	171	S33	174	74	87
S11	330	133	186	S34	291	104	91
S12	427	57	126	S35	85	19	62
S13	458	69	102	S36	291	81	84
S14	296	24	122	S37	81	14	31
S15	85	119	163	S38	79	24	67
S16	474	222	187	S39	85	134	140
S17	410	155	160	S40	291	107	71
S18	474	213	156	S41	119	32	45
S19	296	140	186	S42	255	21	52
S20	246	9	27	S43	255	44	72
S21	50	15	75	S44	79	97	153
S22	476	95	106	S45	334	68	142
S23	476	164	144				
Total					10269	3543	4600

Table 10.2 – Hidden spare capacity due to stub-release for 30-node 45-span network in design process under single failure S01

Single failure	Span	Stub-release
S01	S02	99
S01	S04	119
S01	S07	45
S01	S08	94
S01	S14	38
S01	S15	41
S01	S17	22
S01	S20	30
S01	S24	11
S01	S28	13
S01	S29	4
S01	S30	4
S01	S36	6
S01	S39	15
S01	S41	3
S01	S44	4

Each row in the table represents the amount of working capacity that is reused as spare capacity on the specified span when the span S01 fails. For example, the first row indicates that when single-failure S01 occurs, 99 units of working capacity are assigned to span S02 as its spare capacity. As such, the value of spare capacity for span S02 is increased by 99. From Appendix A.1 we observe that the total value of stub-release in design process is 34920. However, it should be noted that the total amount of spare capacity is not increased by 34920 in design process, because we only consider one single failure each time. Here, we use the average amount of spare capacity due to stub-release as the total hidden spare capacity in design process, which is 776 (34920/45). Apart from design process, availability optimization process allows stub-release as well. The units of spare capacity due to stub-release in availability optimization process under dual-failure scenario

(S01, S02) is listed in Table 10.3, and the full set of values under all dual-failure scenarios is listed in Appendix A.2.

Table 10.3 – Hidden spare capacity due to stub-release for 30-node 45-span network in availability optimization process under dual-failure (S01, S02)

First failure	Second failure	Span	Stub-release
S01	S02	S03	2
S01	S02	S04	75
S01	S02	S06	7
S01	S02	S07	99
S01	S02	S08	57
S01	S02	S10	2
S01	S02	S11	2
S01	S02	S14	22
S01	S02	S15	27
S01	S02	S16	2
S01	S02	S17	66
S01	S02	S19	2
S01	S02	S20	16
S01	S02	S22	9
S01	S02	S23	9
S01	S02	S24	42
S01	S02	S26	9
S01	S02	S28	2
S01	S02	S29	12
S01	S02	S30	20
S01	S02	S33	9
S01	S02	S37	3
S01	S02	S39	10
S01	S02	S41	3
S01	S02	S43	2

Each row in Table 10.3 represents the amount of original working capacity that is reused as spare capacity on the specified span under the specified dual-failure scenario (i.e., dual-failure (S01, S02)). Take the first row as an example, it reveals that 2 units of the original working capacity on failed working routes have been reused as spare capacity on span S03 upon arrival of dual-failure (S01, S02). Calculated from Appendix A.2, the sum of hidden spare capacity due to stub-release is 1497939, and the average stub-release over all dual-failure scenarios is 757 (i.e., 1497939/ (44×45)).

As a result, the total amount of available spare capacity for network 30-node 45-span with path restoration is the sum of: (1) the amount of spare capacity directly from design process (i.e., 3543), (2) the amount of average hidden spare capacity due to stub-release from design process (i.e., 776), and (3) the amount of hidden spare capacity due to stub-release from optimization process (i.e., 757). As such, the total amount of this available spare capacity is 5076. Recall from Table 10.1 that the total amount of available spare capacity with span restoration and SBPP is 10269 and 4600, respectively. Since 10269 is much larger than 5076 and 4600, it is reasonable that span-restorable networks have much higher overall availability than path-restorable and SBPP networks, and that path-restorable and SBPP networks have similar level of overall availability with overall availability of path-restorable networks being slightly larger than SBPP networks on average.

CHAPTER 11 CLOSING DISCUSSION

11.1 SUMMARY OF THESIS

The major objective of this thesis is to provide network researchers with sufficient fundamentals regarding network designs and availability analysis, and to present algorithms for network availability optimization under span restoration, path restoration, and SBPP survivability mechanisms.

We opened this thesis in Chapter 1 with introduction to motivation and goals, and thesis outline. Chapter 2 presents mathematical basics and tools, including graph theory, searching algorithms, linear programming, and programming and solving tools. We documented background in this thesis in Chapter 3, including network classifications, transport networks, mesh network survivability, and related work. In Chapter 4, we presented network availability basics, which includes unavailability of spans, span-oriented mesh networks, and path-oriented networks. In Chapter 5, we presented experimental networks and setup including concepts of network family, topologies of master networks, assumptions, and experimental setup.

In Chapter 6, we investigated the issues of current availability analysis methods for span-restorable mesh networks, and proposed a new algorithm in order to obtain optimal overall network availability for span-restorable mesh networks. We provided thorough analyses of the existing availability analysis methods (i.e., R_2, NWC_2, and SPU_2) and pointed out why they cannot evaluate network overall optimal availability exactly and correctly. Meanwhile, we proposed *SDU* as a new expression for evaluating network

overall availability. Based on this new indicator, we came up with a framework for comparing the current and new availability analysis methods. Two major findings were (1) *SDU* is a more accurate expression of network overall availability, and (2) there is a linear relationship between R_2, SPU_2, and NWC_2, but there is no fixed relationship between SDU and NWC_2.

In Chapter 7, we developed a new ILP design model and an algorithm for analyzing network overall availability for large-scale SBPP networks. We analyzed traditional single-flow and multi-flow ILP design models thoroughly, based on which, we built a new multi-flow ILP design model. The new ILP design model specifies available backup routes for each single working route and is more concise in format, compared to the traditional multi-flow ILP model. The availability analysis algorithm is based on random selection of failed working routes when dealing with the second failure. Key findings of node were (1) the new multi-flow ILP model is 51% faster on average than the traditional multi-flow ILP model, with larger speed improvements for higher connected networks, (2) the new multi-flow ILP model leads to a slight decrease in network overall availability, but provides better overall availability for higher connected networks, and (3) moderate connected networks tend to have higher overall availability and larger scale networks (in terms of number of nodes) tend to have lower overall availability for both new and traditional multi-flow ILP models.

Chapter 8 seeks to optimize overall availability for SBPP mesh networks and investigate relationship between overall availability and spare capacity usage on top of that used for achieving full single-failure restorability. The entire availability optimization algorithm considers failure orders and deals with the two failed failures in order. The core

of the algorithm is an ILP model which is utilized to minimize the total lost flow under each dual-failure scenario. To implement the algorithm, Gurobi is called by Python where an ILP is involved. The availability optimization algorithm is validated with the benchmark algorithm as proposed in Chapter 6. In order to find out the efficient way of increasing network spare capacity in terms of improving overall availability, EIM, GAM, and SSM methods are proposed. Key findings in this chapter included (1) SSM method is the most efficient way to increase network overall availability, and (2) increasing network spare capacity is able to improve network overall availability but the increase extend becomes smaller with more spare capacity is added to the network.

In Chapter 9, we developed an algorithm to optimize network availability for large-scale path-restorable networks and investigate how network average nodal degree and spare capacity increase influence network overall availability for such networks. The basic theory is similar to that for SBPP networks as discussed in Chapter 8, but the details are different. Apart from the ILP model differences between SBPP and path-restorable networks, the major difference for evaluating overall availability is that the latter has a stub-release feature. The working capacity related to this stub-release feature was reused as backup capacity where applicable. This feature added difficulty to the availability optimization of path-restorable networks in comparison with SBPP networks. Key findings in this chapter included (1) as network average nodal degree grows, network overall availability of path-restorable networks has an increase trend at the very beginning and then drops gradually with slight fluctuations, and (2) as we increase network spare capacity, network overall availability increases accordingly, but the increase speed slows with more spare capacity is invested.

Chapter 10 compares performances of networks designed with span restoration, path restoration, and SBPP survivability mechanisms, in terms of network overall availability. Although the algorithm of network availability optimization for these networks has been covered in the previous chapters already, it is not always the key point of these chapters and the settings are not always the same for these three types of networks. As such, in order to compare the performance of these three types of networks, Chapter 10 focuses on network overall availability optimization of these networks by using the same settings and set of experimental networks. In this chapter, we first summarized the methodology for comparison and then run a series of experiments to obtain the performance results. A case study was applied to explain the experimental results. The key points of the findings were (1) span-restorable networks have the highest overall availability among these three types of networks, (2) path-restorable networks have a slight advantage over SBPP networks on average, and (3) the trend of network overall availability is similar for these three types of networks, i.e., the overall availability has a slight increase at the very beginning and then decreases with small fluctuations later on as network average nodal degree climbs up.

11.1.1 MAIN CONTRIBUTIONS

This thesis includes six main contributions:

(1) Availability optimization for span-restorable networks

- analyzed issues of existing availability analysis methods
- proposed a new method to evaluate network optimal availability
- compared performances of existing and new network availability analysis methods

(2) ILP model design for SBPP survivable networks

- performed theoretical analyses of traditional ILP models including traditional single-flow ILP model and traditional multi-flow ILP model
- built new multi-flow ILP model
- compared traditional and new multi-flow ILP models in terms of normalized total cost
- performed analyses of normalized total cost for new and traditional multi-flow models under varying network connectivity

(3) Availability analysis for SBPP survivable networks

- proposed an algorithm to evaluate network overall availability
- compared network overall availability for networks designed with traditional and new ILP design models
- performed analyses of network overall availability under varying network connectivity
- performed analyses of network overall availability under varying network scales

(4) Availability optimization for SBPP networks

- built an ILP model to minimize network total lost flow due to the second failure in a specified dual-failure scenario
- proposed an algorithm to optimize network overall availability
- performed analyses of network overall availability under varying network connectivity
- designed three methods to investigate network spare capacity

- investigated the relationship between network overall availability and network spare capacity increase

(5) Availability optimization for path-restorable networks

- built an ILP model to minimize network total lost flow due to the second failure in a specified dual-failure scenario
- proposed an algorithm to optimize network overall availability
- performed analyses of network overall availability under varying network connectivity
- investigated the relationship between network overall availability and spare capacity increase

(6) Availability performance comparison of span-restorable, path-restorable, and SBPP networks

- presented an availability performance comparison algorithm
- compared network overall availability of the three types of networks
- performed analyses under varying network connectivity
- analyzed reasons behind availability performances

11.2 FUTURE WORK

The work presented in this thesis focuses on the dual-failure scenarios. However, the methodology behind this is not limited to dual-failure scenarios. The algorithm behind each type of networks (i.e., span-restorable networks, path-restorable networks, and SBPP networks) can be extended to multiple failures as well by relevant modifications.

Take the SBPP networks as an example. Please refer to Figure 7.1 to recall the availability analysis algorithm for SBPP networks. Basically, what we do is we loop through each dual-failure scenario to examine the first span failure and the second span failure in order. First, we examine the working routes affected by the first span failure, and then we check the backup routes for each failed working route to see whether they are affected by the second span failure. Later, we examine the working routes affected by the second span failure excluding those already affected by the first span failure. We do this for all the dual failure scenarios and then we are able to obtain network availability eventually.

In the event of triple failures, in order to deal with the third span failure, we can add one more step after Major sub-step 2. In this new step, we remove the exhausted spare capacity on each span before we move on to restore the third failed span. By doing so, we are able to obtain the actual available spare capacity on each span for the restoration of the third failed span. Similar to the way we dealt with the second failed span, we calculate the available backup flow on each backup route for each failed working route based on the updated available spare capacity on each span. After all the triple-failure scenarios are considered, network availability is calcualted eventually.

11.3 PUBLICATIONS OF PH.D. WORK ASSOCIATED WITH THESIS

Apart from the contributions mentioned above, other Ph.D. work associated with this thesis mainly included two peer-reviewed conference papers and three journal papers (one published and two submitted).

1. W. Wang, J. Doucette, “Dual-Failure Availability Analysis of Span-Restorable Mesh Networks,” *Journal of Network and Systems Management*, vol. 24, no. 3, pp. 534–556, July 2016.
2. W. Wang, J. Doucette, “On the Indicators of Service Availability in Span-Restorable Networks,” *7th International Workshop on Reliable Networks Design and Modeling (RNDM 2015)*, pp. 21-26, Munich, Germany, October 2015.
3. W. Wang, J. Doucette, "Dual-Failure Availability Analysis for Multi-Flow Shared Backup Path Protected Mesh Networks," *8th International Workshop on Resilient Networks Design and Modeling (RNDM 2016)*, pp. 127-133, Halmstad, Sweden, October 2016.
4. W. Wang, J. Doucette, “Optimized Design and Availability Analysis of Large-Scale Shared Backup Path Protected Networks,” *Telecommunication Systems (TELS)*, https://doi.org/10.1007/s11235-017-0392-2.
5. W. Wang, J. Doucette, “Availability Optimization and Spare Capacity Impact Analysis for Shared Backup Path Protection Networks,” *Journal of Optical Communications and Networking (JOCN)*, in review; first submitted on July 26 2017; revised and resubmitted on November 17 2017.

11.4 REPORTS OF PH.D. WORK NOT ASSOCIATED WITH THESIS

Because I transferred from another research group to my current research group at the beginning of my third year of my Ph.D. study, I have also made some contributions at

the first two years. These contributions included four technical reports, two oral presentations, and one poster.

1. Wenjing Wang, Yongsheng Ma, "Simulation of Force for Circular Saw Blades Based on MATLAB," *technical report*, University of Alberta, Edmonton, Alberta, Canada, June 18, 2014.
2. Wenjing Wang, Yongsheng Ma, "Geometric Modeling and Cutting Force Simulation for Circular Saw Blades," *Oral presentation*, University of Alberta, Edmonton, Alberta, Canada, April 23, 2014.
3. Wenjing Wang, Yongsheng Ma, "Feature-Based Modeling for Circular Saw Blades," *Poster*, University of Alberta, Edmonton, Alberta, Canada, April 23, 2014.
4. Wenjing Wang, Yongsheng Ma, "Optimization of Slotted Liner Manufacturing," *Oral presentation*, University of Alberta, Edmonton, Alberta, Canada, July 31, 2013.
5. Wenjing Wang, Jonhansel Ng, Yongsheng Ma, "Manufacturing Processes of Slotted Liners Used In SAGD Process," *technical report*, University of Alberta, Edmonton, Alberta, Canada, August 2013.
6. Jonhansel Ng, Wenjing Wang, Yongsheng Ma, "Steam Control Application in SAGD Process," *technical report*, University of Alberta, Edmonton, Alberta, Canada, August 2013.
7. Jonhansel Ng, Wenjing Wang, Yongsheng Ma, "Produce Development in SAGD Process," *technical report*, University of Alberta, Edmonton, Alberta, Canada, August 2013.

Reference

[1] J. Chugh, "Resilience, Survivability and Availability in WDM Optical Mesh Network," *Computing for Sustainable Global Development* (INDIACom), New Delhi, India, pp. 222-227, March 2015.

[2] J. Doucette, W. D. Grover, "Influence of Modularity and Economy-Of-Scale Effects on Design of Mesh-Restorable DWDM Networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1912-1923, 2000.

[3] M. Noor-E-Alam, A. Zaky Kasem, J. Doucette, "ILP Model and Relaxation-Based Decomposition Approach for Incremental Topology Optimization in *P*-Cycle Networks," *Journal of Computer Networks and Communications*, pp. 1-10, 2012.

[4] A. K. Pradhan, K. Das, T. De, "Multicast Traffic Grooming with Survivability in WDM Mesh Networks," *International Conference on Signal Processing and Integrated Networks (SPIN 2015)*, pp. 1020-1025, Noida, India, February 2015.

[5] A. Bhattacharya, M. Agarwal, S. Tabassum, M. Chatterjeee, "Resource Aware Traffic Grooming with Shared Protection at Connection in WDM Mesh Networks," *International Conference on Advances in Computing, Communications and Informatics (ICACCI 2015)*, pp. 110-115, Kochi, India, August 2015.

[6] H. Alazemi, S. Sebbah, M. Nurujjaman, "Fast and Efficient Network Protection Method Using Path Pre-Cross-Connected Trails," *Journal of Optical Communications and Networking*, vol. 5, no. 12, pp. 1343-1352, 2013.

[7] D. A. A. Mello, D. A. Schupke, M. Scheffel, H. Waldman, "Availability Maps for Connections in WDM Optical Networks," *5th International Workshop on Design of Reliable Communication Networks (DRCN 2005)*, pp. 77-84, Island of Ischia, Naples, Italy, October 2005.

[8] M. A. H. Clouqueur, *Availability of Service in Mesh-Restorable Transport Networks*, University of Alberta Ph.D. Thesis, Edmonton, AB, Canada, 2004.

[9] D. A. Schupke, "Guaranteeing Service Availability in Optical Network Design," *2007 ITG Symposium on Photonic Networks*, pp. 1-3, Leipzig, Germany, May 2007.

[10] F. Dikbiyik, B. Mukherjee, M. Tornatore, "Adaptive Time- and Location-Aware Routing in Telecom Mesh Networks," *Networks, IET*, vol. 2, no. 1, pp. 19-29, 2013.

[11] B. Todd, J. Doucette, "Use of Network Families in Survivable Network Design and Optimization," *2008 IEEE International Conference on Communications (ICC 2008)*, pp. 151-157, Beijing, China, May 2008.

[12] J. Doucette, W. D. Grover, "Shared-Risk Logical Span Groups in Span-Restorable Optical Networks: Analysis and Capacity Planning Model," *Photonic Network Communications*, vol. 9, no. 1, pp. 35-53, 2005.

[13] W. Fawaz, F. Martignon, K. Chen, G. Pujolle, "A Novel Protection Scheme for Quality of Service Aware WDM Networks," *2005 IEEE International Conference on Communications (ICC 2005)*, vol. 3, pp. 1720-1725, Seoul, Korean, May 2005.

[14] C. J. Bastos-Filho, R. C. Freitas, D. A. Chaves, R. C. Silva, M. L. Freire, H. A. Pereira, J. F. Martins-Filho, "An Adaptive Path Restoration Algorithm Based on Power Series Routing for All-Optical Networks," *15th International Conference on Transparent Optical Networks* (ICTON 2013), pp. 1-4, Cartagena, Spain, June 2013.

[15] M. Wang, M. Furdek, L. Wosinska, P. Monti, "Wavelength Overprovisioning Strategies for Enhanced Optical Path Restoration," *18th International Conference on Transparent Optical Networks* (ICTON 2016), pp. 1-5, Trento, Italy, July 2016.

[16] G. Conte, M. Listanti, M. Settembre, R. Sabella, "Strategy for Protection and Restoration Of Optical Paths in WDM Backbone Networks for Next-Generation Internet Infrastructures," *Journal of lightwave technology*, vol. 20, no. 8, pp. 1264-1276, 2002.

[17] L. Song, B. Mukherjee, "On the Study of Multiple Backups and Primary-Backup Link Sharing for Dynamic Service Provisioning in Survivable WDM Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 6, pp. 84-91, 2008.
[18] J. Doucette, *Advances on Design and Analysis of Mesh-Restorable Networks*, University of Alberta Ph.D. Thesis, Edmonton, Alberta, Canada, 2005.
[19] M. Herzberg, S. J. Bye, A. Utano, "The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 775-784, December 1995.
[20] R. R. Iraschko, M. H. MacGregor, W. D. Grover, "Optimal Capacity Placement for Path Restoration in STM or ATM Mesh-Survivable Networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, pp. 325-336, June 1998.
[21] A. M. C. A. Koster and A. Zymolka, "Demand-wise shared protection for meshed optical networks," *Fourth International Workshop on Design of Reliable Communication Networks* (DRCN 2003), pp. 85–92, 2003.
[22] S. Sengupta, R. Ramamurthy, "Capacity Efficient Distributed Routing of Mesh-Restored Lightpaths in Optical Networks," *IEEE Global Telecommunications Conference* (GlobeCom 2001), San Antonio, TX, pp. 2129-2133, November 2001.
[23] D. Stamatelakis, W.D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks Using Pre-configured Cycles ("p-cycles")," *IEEE Transactions on Communications*, vol.48, no.8, pp. 1262-1265, August 2000.
[24] W. D. Grover, *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
[25] W. Wang, J. Doucette, "Dual-Failure Availability Analysis of Span-Restorable Mesh Networks," *Journal of Network and Systems Management*, vol. 24, no. 3, pp. 534–556, July 2016.
[26] M. Clouqueur, W. D. Grover, "Availability Analysis of Span-Restorable Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 810-821, May 2002.
[27] A. Nafarieh, S. C. Sivakumar, W. Phillips, W. Robertson, "Memory-Aware SLA-Based Mechanism for Shared-Mesh WDM Networks," *3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops* (ICUMT 2011), pp. 1-8, Budapest, Hungary, October 2011.
[28] L. Song, B. Mukherjee, "New Approaches for Dynamic Routing with Availability Guarantee for Differentiated Services in Survivable Mesh Networks: The Roles of Primary-Backup Link Sharing and Multiple Backup Paths," *Global Telecommunications Conference*, pp. 1-5, San Francisco, CA, USA, November-December 2006.
[29] M. Clouqueur, W. D. Grover, "Mesh-Restorable Networks with Complete Dual Failure Restorability and with Selectively Enhanced Dual-Failure Restorability Properties," *The Convergence of Information Technologies and Communications* (ITCom 2002), pp. 1-12, Boston, MA, USA, July-August 2002.
[30] D. A. Schupke, W. D. Grover, M. Clouqueur, "Strategies for Enhanced Dual Failure Restorability with Static or Reconfigurable *p*-Cycle Networks," *IEEE International Conference on Communications*, vol. 3, pp. 1628-1633, Paris, France, June 2004.
[31] W. Wang, J. Doucette, "On The Indicators of Service Availability in Span-Restorable Networks," *7th International Workshop on Reliable Networks Design and Modeling* (RNDM 2015), pp. 21-26, Munich, Germany, October 2015.
[32] S. Venkatesan, M. Patel, N. Mittal, "A Distributed Algorithm for Path Restoration in Circuit Switched Communication Networks," *24th IEEE Symposium on Reliable Distributed Systems (SRDS 2005)*, pp. 226-235, Orlando, FL, USA, October 2005.
[33] H. T. Mouftah, P. H. Ho, *Optical Networks*, pp. 149-210. *Springer*, USA, 2003.

[34] Y. Xiong, L. G. Mason, "Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks," *IEEE/ACM Transactions on networking*, vol. 7, no. 1, pp. 98-110, 1999.
[35] Y. Liu, D. Tipper, K. Vajanapoom, "Spare Capacity Allocation in Two-Layer Networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 5, pp. 974-986, 2007.
[36] Y. Liu, D. Tipper, "Spare Capacity Allocation for Non-Linear Link Cost And Failure-Dependent Path Restoration," 3rd International Workshop on *Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungary, October 2001.
[37] Y. Liu, "Spare Capacity Allocation: Model, Analysis And Algorithm," Doctoral dissertation, University of Pittsburgh, 2001.
[38] V. Y. Liu, D. Tipper, "Spare Capacity Allocation Using Shared Backup Path Protection For Dual Link Failures," *Computer Communications*, vol. 36, no.6, pp. 666-677, 2013.
[39] P. H. Ho, H. T. Mouftah, "Spare Capacity Allocation for WDM Mesh Networks with Partial Wavelength Conversion Capacity," *2003 High Performance Switching and Routing (HPSR 2003)*. pp. 195-199, Torino, Italy, June 2003.
[40] S. Chen, S. Cheng, B. Chen, J. Chen, "An Efficient Spare Capacity Allocation Strategy for ATM Survivable Networks," *1996 Global Telecommunications Conference (GLOBECOM 1996)*.vol. 1, pp. 442-446, London, UK, November 1996.
[41] Q. Guo, P. H. Ho, A. Haque, H. T. Mouftah, "Availability-Constrained Shared Backup Path Protection (SBPP) for GMPLS-Based Spare Capacity Reprovisioning," *2007 IEEE International Conference on Communications*, pp. 2186-2191, Glasgow, Scotland, June 2007.
[42] A. Al-Rumaih, D. Tipper, Y. Liu, B. A. Norman, "Spare Capacity Planning for Survivable Mesh Networks," *International Conference On Research in Networking*, pp. 957-968. Paris, France, May 2000.
[43] B. Zhou, H. T. Mouftah, "Spare Capacity Planning Using Survivable Alternate Routing for Long-Haul WDM Networks," *7th International Symposium on Computers and Communications (ISCC 2002)*, pp. 732-738, Taormina-Giardini Naxos, Italy, July 2002.
[44] D. Ardagna, C. Ghezzi, B. Panicucci, M. Trubian, "Service Provisioning on the Cloud: Distributed Algorithms for Joint Capacity Allocation and Admission Control," *European Conference on a Service-Based Internet*, pp. 1-12, Ghent, Belgium, December 2010.
[45] H. Wang, *Telecommunications Network Management*, McGraw-Hill, New York City, USA, 1999.
[46] Nwtwork Maps: USA Longhaul, available online: http://www.telecomramblings.com/network-maps/usa-fiber-backbone-map-resources/, retrieved on November 9 2016.
[47] S. Ruepp, J. Buron, N. Andriolli, H. Wessing, "Span Restoration in Optical Networks with Limited Wavelength Conversion," *Second International Conference on Communications and Networking* (CHINACOM 2007), pp. 479-483, Shanghai, China, August 2007.
[48] W. Yue, G. Shen, S. K. Bose, "Span-Restorable Elastic Optical Networks under Different Spectrum Conversion Capabilities," *IEEE Transactions on Reliability*, vol. 63, no. 2, pp. 401-411, June 2014.
[49] W. D. Grover, "Self-Organizing Broad-Band Transport Networks," *Proceedings of the IEEE*, vol. 85, no. 10, pp. 1582-1611, 1997.
[50] R. Ramaswami, K. N. Sivarajan, G. H. Sasaki, *Optical Networks: A Practical Perspective*, Morgan Kaufmann Publishers, Burlington, Massachusetts, USA, 2010.
[51] T. Stidsen, B. Petersen, S. Spoorendonk, M. Zachariasen, K. B. Rasmussen, "Optimal Routing with Failure Independent Path Protection," *Networks*, vol. 55, no. 2, pp. 125-137, 2010.
[52] D. Lucerna, M. Tornatore, B. Mukherjee, A. Pattavina, "Availability Target Redefinition for Dynamic Connections in WDM Networks with Shared Path Protection," *7th International*

Workshop on Design of Reliable Communication Networks (DRCN 2009), pp. 235-242, Washington, DC, USA, October 2009.

[53] M. Furdek, N. Skorin-Kapov, L. Wosinska, "Shared Path Protection under the Risk of High-Power Jamming," *19th European Conference on Networks and Optical Communications* (NOC 2014), pp. 23-28, Milano, Italy, June 2014.

[54] M. M. A. Azim, M. N. Kabir, "Availability Analysis of Shared Backup Path Protection Under Multiple-Link Failure Scenario in WDM Networks," *Annals of Telecommunications*, vol. 70, no. 5-6, pp. 249-262, June 2015.

[55] J. Doucette, M. Clouqueur, W. D. Grover, "On the Availability and Capacity Requirements of Shared Backup Path-Protected Mesh Networks," *Optical Networks Magazine*, vol. 4, no. 6, pp. 29-44, 2003.

[56] C. M. Delgado, H. P. Silva, M. M. Mosso, "A Novel Approach to Automatic Protection Switching for Ethernet Optical Networks," *SBMO/IEEE MTT-S International Microwave & Optoelectronics Conference* (IMOC), pp. 1-5, Rio de Janeiro, RJ, Brazil, August 2013.

[57] W. Grover, J. Doucette, M. Clouqueur, D. Leung, D. Stamatelakis, "New Options and Insights for Survivable Transport Networks," *IEEE Communications Magazine*, vol. 40, no. 1, pp. 34-41, 2002.

[58] J. Akpuh, J. Doucette, "Sizing Eligible Route Sets for Restorable Network Design and Optimization," *IEEE International Conference on Communications*, pp. 5292-5299, Beijing, China, May 2008.

[59] M. Tornatore, G. Maier, A. Pattavina, "Variable Aggregation in The ILP Design of WDM Networks with Dedicated Protection," *Journal of Communications and Networks*, vol. 9, no. 4, pp. 419-427, 2007.

[60] A. J. Gonzalez, B. E. Helvik, "Dynamic Sharing Mechanism for Guaranteed Availability in MPLS Based Networks," *2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2010)*, pp. 1-6, Vancouver, BC, Canada, June 2010.

[61] B. Kantarci, H. T. Mouftah, S. Oktug, "Arranging Shareability Dynamically for The Availability-Constrained Design of Optical Transport Networks," *IEEE Symposium on Computers and Communications (ISCC 2008)*, pp. 68-73, Marrakech, Morocco, July 2008.

[62] B. G. Józsa, D. Orincsay, "Shared Backup Path Optimization in Telecommunication Networks," *3rd Design of Reliable Communication Networks Workshop (DRCN 2001)*, pp. 251-257, Budapest, Hungary, October, 2001.

[63] G. Shen, W. D. Grover, "Survey and Performance Comparison of Dynamic Provisioning Methods for Optical Shared Backup Path Protection," *2nd International Conference on Broadband Networks*, pp. 1310-1319, Boston, MA, USA, October 2005.

[64] C. Ou, K. Zhu, H. Zang, L. H. Sahasrabuddhe, B. Mukherjee, "Traffic Grooming for Survivable WDM Networks-Shared Protection," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 9, pp. 1367-1383, 2003.

[65] C. Ou, J. Zhang, H. Zang, L. H. Sahasrabuddhe, B. Mukherjee, "Near-Optimal Approaches for Shared-Path Protection in WDM Mesh Networks," *2003 IEEE International Conference on Communications (ICC2003)*, vol. 2, pp. 1320-1324, Anchorage, Alaska, USA, May 2003.

[66] H. N. Nguyen, D. Habibi, Q. V. Phung, K. Lo, "A Shared Backup Path Protection Scheme for Optical Mesh Networks," *2005 Asia-Pacific Conference on Communications*, pp. 309-313, Perth, WA, Australia, October 2005.

[67] B. G. Józsa, D. Orincsay, A. Kern, "Surviving Multiple Network Failures Using Shared Backup Path Protection," *8th IEEE International Symposium on Computers and Communication (ISCC 2003)*, pp. 1333-1340, Kemer-Antalya, Turkey, June-July 2003.

[68] J. T. Haahr, T. Stidsen, M. Zachariasen, "Heuristic Methods for Shared Backup Path Protection Planning," *4th International Congress on Ultra Modern Telecommunications and*

Control Systems and Workshops (ICUMT 2012), pp. 712-718, St. Petersburg, Russia, October 2012.

[69] K. Walkowiak, M. Klinkowski, "Shared Backup Path Protection in Elastic Optical Networks: Modeling and Optimization," *9th International Conference on the Design of Reliable Communication Networks (DRCN 2013)*, pp. 187-194, Budapest, Hungary, March 2013.

[70] C. Wang, G. Shen, S. K. Bose, "Distance Adaptive Dynamic Routing and Spectrum Allocation in Elastic Optical Networks with Shared Backup Path Protection," *Journal of Lightwave Technology*, vol. 33, no. 14, 2015.

[71] P. Ho, J. Tapolcai, A. Haque, "A Study On Dynamic Survivable Routing with Availability Constraint for GMPLS-Based Recovery," *3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS 2006)*, pp. 1-10, San Jose, CA, USA, October 2006.

[72] W. Lau, S. Jha, "Failure-Oriented Path Restoration Algorithm for Survivable Networks," *IEEE Transactions on Network and Service Management*, vol. 1, no. 1, pp. 11-20, 2004.

[73] S. Ruepp, L. Dittman, L. Ellegard, "Simulation and Comparison Of Path Restoration Techniques in SDH Mesh Networks," *5th International Workshop on Design of Reliable Communication Networks (DRCN 2005)*, pp. 47-53, Brugge, Belgium, October 2005.

[74] P. Ho, J. Tapolcai, A. Haque, "Spare Capacity Reprovisioning for Shared Backup Path Protection in Dynamic Generalized Multi-Protocol Label Switched Networks," *IEEE Transactions on Reliability*, vol. 57, no. 4, pp. 551-563, 2008.

[75] B. Kantarci, H. T. Mouftah, S. Oktug, "Connection Provisioning with Feasible Shareability Determination for Availability-Aware Design of Optical Networks," *10th Anniversary International Conference on Transparent Optical Networks (ICTON 2008)*, vol. 3, pp. 19-22, Athens, Greece, June 2008.

[76] W. Ni, E. Patzak, M. Schlosser, H. Zhang, "Availability Evaluation in Shared-Path-Protected WDM Networks with Startup-Failure-Driven Backup Path Reprovisioning," *2010 IEEE International Conference on Communications*, pp. 1-6, Cape Town, South Africa, May 2010.

[77] B. Todd, J. Doucette, "Fast Efficient Design of Shared Backup Path Protected Networks Using a Multi-Flow Optimization Model," *IEEE Transactions on Reliability*, vol. 60, no. 4, pp. 788-800, 2011.

[78] M. Kodialam, T. V. Lakshman, S. Sengupta, "Guaranteed Performance Routing of Unpredictable Traffic with Fast Path Restoration," *IEEE/ACM Transactions on Networking* (TON), vol. 17, no. 5, pp. 1427-1438, 2009.

[79] L. Zhou, M. Held, "Optimization of Path Availability of Span-Restorable Optical Networks," *Photonics Europe, Reliability of Optical Fiber Components, Devices, Systems, and Networks*, Strasbourg, France, May 2006.

[80] W. Li, J. Doucette, M. Zuo, "*p*-Cycle Network Design for Specified Minimum Dual-Failure Restorability," *IEEE International Conference on Communications* (ICC 2007), pp. 2204-2210, Glasgow, Scotland, June 2007.

[81] S. Herker, X. An, W. Kiess, A. Kirstadter, "Path Protection with Explicit Availability Constraints for Virtual Network Embedding," *Personal Indoor and Mobile Radio Communications (PIMRC)*, pp. 2978-2983, London, UK, September 2013.

[82] A. Alashaikh, T. Gomes, D. Tipper, "The Spine Concept for Improving Network Availability," *Computer Networks*, vol. 82, pp. 4-19, May 2015.

[83] A. E. Conway, "Fast Simulation of Service Availability in Mesh Networks with Dynamic Path Restoration," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, 92-101, 2011.

[84] G. Agnarsson, R. Greenlaw, *Graph Theory: Modeling, Applications, and Algorithms*, Pearson/Prentice Hall, Upper Saddle River, NJ, USA, 2007.

[85] J. A. Bondy, U. S. R. Murty, *Graph theory*, Springer, New York, USA, 2008.

[86] V. I. Voloshin, *Introduction to Graph Theory*, Nova Science Publishers, New York, USA, 2009.

[87] S. R. Santanu, *Graph Theory with Algorithms and Its Applications: in Applied Science and Technology*, Springer, New Delhi, India, 2013.
[88] D. Jungnickel, *Graphs, Networks, and Algorithms*, Springer, Berlin, Germany, 2008.
[89] D. Eppstein, "Finding the K Shortest Paths," *SIAM Journal on computing*, vol. 28, no. 2, pp. 652-673, 1998.
[90] J. Hershberger, M. Maxel, S. Suri, "Finding the K Shortest Simple Paths: A New Algorithm and Its Implementation," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 4, pp. 45, 2007.
[91] L. W. Wayne, J. B. Goldberg, *Operations Research: Applications and Algorithms*, Thomson/Brooks/Cole, Belmont, CA, USA, 2004.
[92] L. S. Lasdon, *Optimization Theory for Large Systems*, Dover Publications, Inc., Mineola, New York, USA, 2002.
[93] X. Yang, *Introduction to Mathematical Optimization: from Linear Programming to Metaheuristics*, Cambridge International Science Publishing, Cambridge, UK, 2008.
[94] G. Baumann, *Mathematics for Engineers II: Calculus and Linear Algebra*, Berlin ;Boston: Oldenbourg Wissenschaftsverlag, München, Germany, 2010.
[95] Tongji University, *Engineering Math: Linear Algebra* (in Chinese), Higher Education Press, Beijing, China, 2003.
[96] S. Zionts, *Linear and Integer Programming*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1974.
[97] A. H. Land, A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica: Journal of the Econometric Society*, pp. 497-520, 1960.
[98] J. Clausen, "Branch and Bound Algorithms-Principles and Examples," Department of Computer Science, University of Copenhagen, pp. 1-30, 1999.
[99] D. Scholz, *Deterministic Global Optimization: Geometric Branch-and-Bound Methods and Their Applications*, Springer, New York, USA, 2012.
[100] R. Fourer, D. M. Gay, B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Cengage Learning, Boston, MA, USA, 2003.
[101] Gurobi Optimization Inc., Gurobi Optimizer Reference Manual, 2015, available online: http://www.gurobi.com, retrieved on 5 April 2016.
[102] K. A. Lambert, *Fundamentals of Python: First Programs,* Cengage Learning, Boston, MA, USA, 2011.
[103] Python website, available online: https://www.python.org/, retrieved 10 April 2016.
[104] X. Jie, H. Wen, B. Wu, X. Jiang, P. Ho, L. Zhang, "Joint Design on DCN Placement and Survivable Cloud Service Provision over All-Optical Mesh Networks," *IEEE Transactions on Communications*, vol. 62, no. 1, pp. 235-245, 2014.
[105] J. Doucette, W. Li, M. Zuo, "Failure-specific p-cycle network dual-failure restorability design," *6th International Workshop on Design and Reliable Communication Networks* (DRCN 2007), pp. 1-9, La Rochelle, France, October 2007.
[106] B. Todd, J. Doucette, "DSP Survivable Network Capacity Allocation and Topology Design Using Multi-Period Network Augmentation," *11th International Conference on the Design of Reliable Communication Networks (DRCN 2015)*, pp. 41-48, Kansas City, MO, USA, 2015.
[107] W. D. Grover, J. Doucette, "Design of a Meta-Mesh of Chain Subnetworks: Enhancing the Attractiveness of Mesh-Restorable WDM Networking on Low Connectivity Graphs," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 1, pp. 47-61, 2002.
[108] M. Ben-Daya, S.O. Duffuaa, A. Raouf, J. Knezevic, D. Ait-Kadi, Handbook of Maintenance Management and Engineering. Springer Dordrecht Heidelberg London New York, 2009.
[109] X. Wang, G. Shen, Z. Zhu, X. Fu, "Benefits of Sub-band Virtual Concatenation for Enhancing Availability of Elastic Optical Networks," *Journal of Lightwave Technology*, vol. 34, no. 4, pp. 1098-1110, 2016.

[110] A. J. Vernon, J. D. Portier, "Protection of Optical Channels in All-Optical Net-Works," *National Fiber Optic Engineers Conference (NFOEC 2002)*, pp. 1695–1706, Dallas, TX, September 2002.
[111] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, M. Jaeger, "General Availability Model for Multilayer Transport Networks," *5th International Workshop on Design of Reliable Communication Networks (DRCN 2005)*, pp. 85-92, Brugge, Belgium, October 2005.
[112] W. Ni, J. Wu, C. Huang, M. Savoie, "Analytical Models of Flow Availability in Two-Layer Networks with Dedicated Path Protection," *Optical Switching and Networking*, vol. 10, no. 1, pp. 62-76, 2013.
[113] D. A. Schupke, "Multiple Failure Survivability in WDM Networks with *p*-Cycles," *IEEE International Symposium on Circuits and Systems (ISCAS 2003)*, vol. 3, pp. 866-869, Bangkok, Thailand, May 2003.
[114] C. Singh, R. Billinton, *System Reliability, Modelling and Evaluation*. Hutchinson, London, UK, 1977.
[115] L. Zhou, M. Held, U. Sennhauser, "Connection Availability Analysis of Shared Backup Path-Protected Mesh Networks," *Journal of Lightwave Technology*, vol. 25, no. 5, pp. 1111-1119, 2007.
[116] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, Reading, MA, USA, 1990.
[117] M. Held, L. Zhou, "Redundancy, Restorability and Path Availability in Optical Mesh Networks," *2006 International Conference on Transparent Optical Networks*, vol. 3, pp. 120-125, Nottingham, United Kingdom, June 2006.
[118] G. Shen, W. D. Grover, "Exploiting forcer structure to serve uncertain demands and minimize redundancy of p-cycle networks," *Proceedings of SPIE*, vol. 5285, pp. 59-70, October 2003.
[119] W. Wang, J. Doucette, "Optimized Design and Availability Analysis of Large-Scale Shared Backup Path Protected Networks," Submitted to *Telecommunication Systems (TELS)* on August 29 2016.
[120] A. Charnes, E. L. Frome, P. L. Yu, "The Equivalence of Generalized Least Squares and Maximum Likelihood Estimates in the Exponential Family," *Journal of the American Statistical Association*, vol. 71, no. 353, pp. 169-171, 1976.

Appendix A Stub-release Data

Appendix A.1 – Stub-release in design process for 30-node 45-span network

SF Spans		Stub-release
S01	S02	99
S01	S04	119
S01	S07	45
S01	S08	94
S01	S14	38
S01	S15	41
S01	S17	22
S01	S20	30
S01	S24	11
S01	S28	13
S01	S29	4
S01	S30	4
S01	S36	6
S01	S39	15
S01	S41	3
S01	S44	4
S02	S01	99
S02	S03	2
S02	S04	75
S02	S06	7
S02	S07	99
S02	S08	57
S02	S10	2
S02	S11	2
S02	S14	22
S02	S15	27
S02	S16	2
S02	S17	66
S02	S19	2
S02	S20	16
S02	S22	9
S02	S23	9
S02	S24	42
S02	S26	9
S02	S28	2
S02	S29	12
S02	S30	20
S02	S33	9
S02	S34	9
S02	S37	3
S02	S39	10
S02	S41	3
S02	S43	2
S03	S02	2
S03	S10	36
S03	S11	26
S03	S16	17
S03	S18	11
S03	S19	5
S03	S23	7
S03	S26	1
S04	S01	119
S04	S02	75
S04	S05	104
S04	S07	28
S04	S08	251
S04	S10	54
S04	S12	31
S04	S13	3
S04	S14	81
S04	S15	133
S04	S17	9
S04	S20	50
S04	S21	4
S04	S24	4
S04	S28	25
S04	S36	10
S04	S38	3
S04	S39	81
S04	S40	9
S04	S41	12
S04	S44	33
S04	S45	16
S05	S04	104
S05	S08	87
S05	S10	86
S05	S11	20
S05	S12	62
S05	S13	28
S05	S14	15
S05	S15	54
S05	S16	18
S05	S17	25
S05	S18	12
S05	S23	8
S05	S24	25
S05	S26	4
S05	S30	8
S05	S31	3
S05	S33	12
S05	S37	6
S05	S39	38
S05	S44	17
S05	S45	6
S06	S02	7
S06	S10	15
S06	S11	7
S06	S12	20
S06	S16	5
S07	S01	45
S07	S02	99
S07	S04	28
S07	S08	18
S07	S14	9
S07	S15	8
S07	S17	131
S07	S20	5
S07	S22	26
S07	S23	19
S07	S24	86
S07	S26	15
S07	S29	20
S07	S30	31
S07	S31	14
S07	S33	33
S07	S34	15
S07	S37	11
S07	S43	13
S07	S45	9
S08	S01	94
S08	S02	57
S08	S04	251
S08	S05	87
S08	S07	18
S08	S09	45
S08	S10	42
S08	S11	1
S08	S12	22
S08	S13	3
S08	S14	145
S08	S15	185
S08	S16	5
S08	S18	16
S08	S19	18
S08	S20	78
S08	S21	29
S08	S22	5
S08	S23	3
S08	S28	51
S08	S34	4
S08	S36	25
S08	S38	7
S08	S39	120
S08	S40	16
S08	S41	16
S08	S43	8
S08	S44	58
S08	S45	39
S09	S08	45
S09	S11	1
S09	S14	16
S09	S15	20
S09	S16	21
S09	S18	21
S09	S20	6
S09	S22	5
S09	S23	16
S09	S25	9
S09	S26	7
S09	S28	6
S09	S31	5
S09	S36	5
S09	S39	10
S09	S44	3
S09	S45	3
S10	S02	2
S10	S03	36
S10	S04	54
S10	S05	86
S10	S06	15
S10	S08	42
S10	S11	176
S10	S12	188
S10	S13	92
S10	S14	10
S10	S15	22
S10	S16	128
S10	S17	63
S10	S18	42
S10	S19	62
S10	S23	28
S10	S24	56
S10	S25	5
S10	S26	7
S10	S27	36
S10	S28	18
S10	S30	13
S10	S31	1
S10	S33	29
S10	S36	8
S10	S37	9
S10	S39	16
S10	S44	4
S11	S02	2
S11	S03	26
S11	S05	20
S11	S06	7
S11	S08	1
S11	S09	1
S11	S10	176
S11	S12	90
S11	S13	44
S11	S14	1
S11	S16	212
S11	S17	21
S11	S18	79
S11	S19	93
S11	S20	1
S11	S23	59
S11	S24	17
S11	S25	7
S11	S26	26
S11	S27	60
S11	S28	34
S11	S31	15
S11	S32	6
S11	S33	9
S11	S34	1
S11	S36	18
S12	S04	31
S12	S05	62
S12	S06	20
S12	S08	22
S12	S10	188

S12	S11	90
S12	S13	129
S12	S14	6
S12	S15	10
S12	S16	57
S12	S17	90
S12	S18	6
S12	S19	37
S12	S22	6
S12	S23	10
S12	S24	70
S12	S26	6
S12	S27	20
S12	S28	10
S12	S30	13
S12	S33	42
S12	S34	6
S12	S36	1
S12	S37	11
S12	S39	5
S12	S43	10
S13	S04	3
S13	S05	28
S13	S08	3
S13	S10	92
S13	S11	44
S13	S12	129
S13	S16	17
S13	S17	150
S13	S19	15
S13	S22	6
S13	S23	6
S13	S24	121
S13	S26	6
S13	S27	5
S13	S29	23
S13	S30	41
S13	S31	16
S13	S33	64
S13	S34	16
S13	S37	20
S13	S43	13
S13	S45	1
S14	S01	38
S14	S02	22
S14	S04	81
S14	S05	15
S14	S07	9
S14	S08	145
S14	S09	16
S14	S10	10
S14	S11	1
S14	S12	6
S14	S16	5
S14	S18	29
S14	S19	42
S14	S20	101
S14	S21	63
S14	S23	16
S14	S25	7
S14	S28	73
S14	S31	8
S14	S34	10
S14	S36	43
S14	S40	17
S15	S01	41
S15	S02	27
S15	S04	133
S15	S05	54
S15	S07	8
S15	S08	185
S15	S09	20
S15	S10	22
S15	S12	10
S15	S33	4
S15	S34	4
S15	S38	17
S15	S39	158
S15	S40	13
S15	S41	16
S15	S43	17
S15	S44	90
S15	S45	64
S16	S02	2
S16	S03	17
S16	S05	18
S16	S06	5
S16	S08	5
S16	S09	21
S16	S10	128
S16	S11	212
S16	S12	57
S16	S13	17
S16	S14	5
S16	S18	142
S16	S19	118
S16	S21	5
S16	S22	5
S16	S23	112
S16	S25	17
S16	S26	64
S16	S27	76
S16	S28	45
S16	S31	32
S16	S32	15
S16	S34	13
S16	S36	21
S16	S42	7
S17	S01	22
S17	S02	66
S17	S04	9
S17	S05	25
S17	S07	131
S17	S10	63
S17	S11	21
S17	S12	90
S17	S13	150
S17	S18	1
S17	S19	1
S17	S22	50
S17	S23	38
S17	S24	271
S17	S26	31
S17	S27	1
S17	S28	1
S17	S29	58
S17	S30	92
S17	S31	55
S17	S33	134
S17	S34	41
S17	S37	40
S17	S40	6
S17	S43	45
S17	S44	10
S17	S45	24
S18	S03	11
S18	S05	12
S18	S08	16
S18	S09	21
S18	S10	42
S18	S11	79
S18	S12	6
S18	S14	29
S18	S16	142
S18	S17	1
S18	S19	133
S18	S21	55
S18	S22	62
S18	S23	194
S18	S24	26
S18	S25	51
S18	S26	95
S18	S27	34
S18	S28	14
S18	S30	19
S18	S31	59
S18	S32	31
S18	S33	20
S18	S34	16
S18	S35	5
S18	S42	10
S18	S43	10
S19	S02	2
S19	S03	5
S19	S08	18
S19	S10	62
S19	S11	93
S19	S12	37
S19	S13	15
S19	S14	42
S19	S16	118
S19	S17	1
S19	S18	133
S19	S21	76
S19	S22	48
S19	S23	42
S19	S24	23
S19	S25	22
S19	S26	8
S19	S27	153
S19	S28	95
S19	S30	9
S19	S31	8
S19	S32	6
S19	S33	9
S19	S35	13
S19	S36	43
S19	S40	6
S19	S41	8
S20	S01	30
S20	S02	16
S20	S04	50
S20	S07	5
S20	S08	78
S20	S09	6
S20	S11	1
S20	S14	101
S20	S26	1
S20	S28	139
S20	S29	4
S20	S31	22
S20	S32	6
S20	S34	27
S20	S35	10
S20	S36	98
S20	S39	1
S20	S40	42
S20	S41	20
S20	S43	6
S20	S44	18
S20	S45	13
S21	S04	4
S21	S08	29
S21	S14	63
S21	S16	5
S21	S18	55
S21	S19	76
S21	S22	16
S21	S23	16
S21	S24	6
S21	S25	7
S22	S02	9
S22	S07	26
S22	S08	5
S22	S09	5
S22	S12	6
S22	S13	6
S22	S16	5
S22	S17	50
S22	S18	62
S22	S19	48
S22	S21	16
S22	S23	40
S22	S24	38
S22	S26	31
S22	S27	9
S22	S28	5
S22	S31	16
S22	S33	15
S22	S34	15
S22	S43	3
S23	S02	9
S23	S03	7
S23	S05	8
S23	S07	19
S23	S08	3
S23	S09	16
S23	S10	28
S23	S11	59
S23	S12	10
S23	S13	6
S23	S14	16

S23	S16	112
S23	S17	38
S23	S18	194
S23	S19	42
S23	S21	16
S23	S22	40
S23	S25	69
S23	S26	167
S23	S27	9
S23	S29	6
S23	S30	23
S23	S31	88
S23	S32	40
S23	S33	12
S23	S34	51
S23	S36	3
S23	S38	6
S23	S40	13
S23	S41	2
S23	S42	12
S23	S43	11
S24	S01	11
S24	S02	42
S24	S04	4
S24	S05	25
S24	S07	86
S24	S10	56
S24	S11	17
S24	S12	70
S24	S13	121
S24	S17	271
S24	S18	26
S24	S19	23
S24	S21	6
S24	S22	38
S24	S29	92
S24	S30	133
S24	S31	66
S24	S33	174
S24	S34	45
S24	S37	47
S24	S39	2
S24	S40	16
S24	S43	62
S24	S44	20
S24	S45	37
S25	S09	9
S25	S10	5
S25	S11	7
S25	S14	7
S25	S16	17
S25	S18	51
S25	S19	22
S25	S21	7
S25	S23	69
S25	S27	9
S25	S29	6
S25	S30	33
S25	S32	6
S25	S33	21
S25	S38	6
S25	S43	11
S26	S02	9
S26	S03	1
S26	S05	4
S26	S07	15
S26	S09	7
S26	S10	7
S26	S11	26
S26	S12	6
S26	S13	6
S26	S16	64
S26	S17	31
S26	S18	95
S26	S19	8
S26	S20	1
S26	S22	31
S26	S23	167
S26	S28	8
S26	S31	107
S26	S32	47
S26	S34	78
S26	S36	14
S26	S38	10
S26	S39	5
S26	S40	32
S26	S41	9
S26	S42	18
S27	S10	36
S27	S11	60
S27	S12	20
S27	S13	5
S27	S16	76
S27	S17	1
S27	S18	34
S27	S19	153
S27	S22	9
S27	S23	9
S27	S25	9
S27	S28	142
S27	S30	9
S27	S34	9
S27	S35	18
S27	S36	78
S27	S40	32
S27	S41	11
S27	S42	16
S27	S45	6
S28	S01	13
S28	S02	2
S28	S04	25
S28	S08	51
S28	S09	6
S28	S10	18
S28	S11	34
S28	S12	10
S28	S14	73
S28	S16	45
S28	S17	1
S28	S18	14
S28	S19	95
S28	S20	139
S28	S22	5
S28	S26	8
S28	S27	142
S28	S29	14
S28	S31	43
S28	S32	10
S28	S34	73
S28	S35	39
S28	S36	238
S28	S39	3
S28	S40	118
S28	S41	45
S28	S42	16
S28	S43	9
S28	S44	32
S28	S45	27
S29	S01	4
S29	S02	12
S29	S07	20
S29	S13	23
S29	S17	58
S29	S20	4
S29	S23	6
S29	S24	92
S29	S25	6
S29	S28	14
S29	S30	108
S29	S31	119
S29	S32	27
S29	S34	84
S29	S36	23
S29	S38	17
S29	S40	46
S29	S45	3
S30	S01	4
S30	S02	20
S30	S05	8
S30	S07	31
S30	S10	13
S30	S12	13
S30	S13	41
S30	S17	92
S30	S18	19
S30	S19	9
S30	S23	23
S30	S24	133
S30	S25	33
S30	S27	9
S30	S29	108
S30	S31	77
S30	S33	32
S30	S34	47
S30	S39	2
S30	S40	16
S30	S43	19
S30	S44	3
S30	S45	8
S31	S05	3
S31	S07	14
S31	S09	5
S31	S10	1
S31	S11	15
S31	S13	16
S31	S14	8
S31	S16	32
S31	S17	55
S31	S18	59
S31	S19	8
S31	S20	22
S31	S22	16
S31	S23	88
S31	S24	66
S31	S26	107
S31	S28	43
S31	S29	119
S31	S30	77
S31	S32	83
S31	S34	152
S31	S36	61
S31	S37	2
S31	S38	15
S31	S39	3
S31	S40	107
S31	S41	5
S32	S11	6
S32	S16	15
S32	S18	31
S32	S19	6
S32	S20	6
S32	S23	40
S32	S25	6
S32	S26	47
S32	S28	10
S32	S29	27
S32	S31	83
S32	S34	22
S32	S36	11
S32	S37	9
S32	S38	39
S32	S40	20
S32	S45	4
S33	S02	9
S33	S05	12
S33	S07	33
S33	S10	29
S33	S11	9
S33	S12	42
S33	S13	64
S33	S15	4
S33	S17	134
S33	S18	20
S33	S19	9
S33	S22	15
S33	S23	12
S33	S24	174
S33	S25	21
S33	S30	32
S33	S36	3
S33	S37	54
S33	S39	14
S33	S41	4
S33	S43	107
S33	S44	44
S33	S45	70
S34	S02	9
S34	S07	15
S34	S08	4
S34	S11	1
S34	S12	6
S34	S13	16

S34	S14	10
S34	S15	4
S34	S16	13
S34	S17	41
S34	S18	16
S34	S20	27
S34	S22	15
S34	S23	51
S34	S24	45
S34	S26	78
S34	S27	9
S34	S28	73
S34	S29	84
S34	S30	47
S34	S31	152
S34	S32	22
S34	S36	107
S34	S39	21
S34	S40	195
S34	S41	31
S34	S42	26
S35	S18	5
S35	S19	13
S35	S20	10
S35	S27	18
S35	S28	39
S35	S39	10
S35	S44	3
S35	S45	2
S36	S01	6
S36	S04	10
S36	S08	25
S36	S09	5
S36	S10	8
S36	S11	18
S36	S12	1
S36	S14	43
S36	S16	21
S36	S19	43
S36	S20	98
S36	S23	3
S36	S26	14
S36	S27	78
S36	S28	238
S36	S29	23
S36	S31	61
S36	S32	11
S36	S33	3
S36	S34	107
S36	S40	178
S36	S41	48
S36	S42	38
S36	S43	22
S36	S44	34
S36	S45	47
S37	S02	3
S37	S05	6
S37	S07	11
S37	S10	9
S37	S12	11
S37	S13	20
S37	S17	40
S37	S24	47
S37	S31	2
S37	S32	9
S37	S33	54
S38	S04	3
S38	S08	7
S38	S15	17
S38	S23	6
S38	S25	6
S38	S26	10
S38	S29	17
S38	S31	15
S38	S32	39
S38	S39	25
S38	S44	32
S38	S45	43
S39	S01	15
S39	S02	10
S39	S04	81
S39	S05	38
S39	S08	120
S39	S09	10
S39	S10	16
S39	S12	5
S39	S15	158
S39	S20	1
S39	S24	2
S39	S26	5
S39	S28	3
S39	S30	2
S39	S31	3
S39	S33	14
S39	S34	21
S39	S35	10
S39	S38	25
S39	S40	36
S39	S41	43
S39	S43	36
S39	S44	129
S39	S45	96
S40	S04	9
S40	S08	16
S40	S14	17
S40	S15	13
S40	S17	6
S40	S19	6
S40	S20	42
S40	S23	13
S40	S24	16
S40	S26	32
S40	S27	32
S40	S28	118
S40	S29	46
S40	S30	16
S40	S31	107
S40	S32	20
S40	S34	195
S40	S36	178
S40	S39	36
S40	S41	51
S40	S42	51
S40	S43	14
S40	S45	27
S41	S01	3
S41	S02	3
S41	S04	12
S41	S08	16
S41	S15	16
S41	S19	8
S41	S20	20
S41	S23	2
S41	S26	9
S41	S27	11
S41	S28	45
S41	S31	5
S41	S33	4
S41	S34	31
S41	S36	48
S41	S39	43
S41	S40	51
S41	S43	13
S41	S44	38
S41	S45	28
S42	S16	7
S42	S18	10
S42	S23	12
S42	S26	18
S42	S27	16
S42	S28	16
S42	S34	26
S42	S36	38
S42	S40	51
S42	S43	23
S42	S45	44
S43	S02	2
S43	S07	13
S43	S08	8
S43	S12	10
S43	S13	13
S43	S15	17
S43	S17	45
S43	S18	10
S43	S20	6
S43	S22	3
S43	S23	11
S43	S24	62
S43	S25	11
S43	S28	9
S43	S30	19
S43	S33	107
S43	S36	22
S43	S39	36
S43	S40	14
S43	S41	13
S43	S42	23
S43	S44	79
S43	S45	132
S44	S01	4
S44	S04	33
S44	S05	17
S44	S08	58
S44	S09	3
S44	S10	4
S44	S15	90
S44	S17	10
S44	S20	18
S44	S24	20
S44	S28	32
S44	S30	3
S44	S33	44
S44	S35	3
S44	S36	34
S44	S38	32
S44	S39	129
S44	S41	38
S44	S43	79
S44	S45	169
S45	S04	16
S45	S05	6
S45	S07	9
S45	S08	39
S45	S09	3
S45	S13	1
S45	S15	64
S45	S17	24
S45	S20	13
S45	S24	37
S45	S27	6
S45	S28	27
S45	S29	3
S45	S30	8
S45	S32	4
S45	S33	70
S45	S35	2
S45	S36	47
S45	S38	43
S45	S39	96
S45	S40	27
S45	S41	28
S45	S42	44
S45	S43	132
S45	S44	169

Appendix A.2 – Stub-release in optimization process for 30-node 45-span network

Note: only the data for dual failures with S01 as the first failure is shown here. Full set of data takes up almost 200 pages under current layout, so we decide not to show all of them here. However, full seat of data is available upon request.

FF	SF	Spans	Stub-release
S01	S02	S03	2
S01	S02	S04	75
S01	S02	S06	7
S01	S02	S07	99
S01	S02	S08	57
S01	S02	S10	2
S01	S02	S11	2
S01	S02	S14	22
S01	S02	S15	27
S01	S02	S16	2
S01	S02	S17	66
S01	S02	S19	2
S01	S02	S20	16
S01	S02	S22	9
S01	S02	S23	9
S01	S02	S24	42
S01	S02	S26	9
S01	S02	S28	2
S01	S02	S29	12
S01	S02	S30	20
S01	S02	S33	9
S01	S02	S37	3
S01	S02	S39	10
S01	S02	S41	3
S01	S02	S43	2
S01	S03	S02	2
S01	S03	S10	36
S01	S03	S11	26
S01	S03	S16	17
S01	S03	S18	11
S01	S03	S19	5
S01	S03	S23	7
S01	S03	S26	1
S01	S04	S02	75
S01	S04	S05	104
S01	S04	S07	28
S01	S04	S08	251
S01	S04	S10	54
S01	S04	S12	31
S01	S04	S13	3
S01	S04	S14	81
S01	S04	S15	133
S01	S04	S17	9
S01	S04	S20	50
S01	S04	S21	4
S01	S04	S24	4
S01	S04	S28	25
S01	S04	S36	10
S01	S04	S38	3
S01	S04	S39	81
S01	S04	S40	9
S01	S04	S41	12
S01	S04	S44	33
S01	S04	S45	16
S01	S05	S04	104
S01	S05	S08	87
S01	S05	S10	86
S01	S05	S11	20
S01	S05	S12	62
S01	S05	S13	28
S01	S05	S14	15
S01	S05	S15	54
S01	S05	S16	18
S01	S05	S17	25
S01	S05	S18	12
S01	S05	S23	8
S01	S05	S24	25
S01	S05	S26	4
S01	S05	S30	8
S01	S05	S31	3
S01	S05	S33	12
S01	S05	S37	6
S01	S05	S39	38
S01	S05	S44	17
S01	S05	S45	6
S01	S06	S02	7
S01	S06	S10	15
S01	S06	S11	7
S01	S06	S12	20
S01	S06	S16	5
S01	S07	S02	99
S01	S07	S04	28
S01	S07	S08	18
S01	S07	S14	9
S01	S07	S15	8
S01	S07	S17	131
S01	S07	S20	5
S01	S07	S22	26
S01	S07	S23	19
S01	S07	S24	86
S01	S07	S26	15
S01	S07	S29	20
S01	S07	S30	31
S01	S07	S31	14
S01	S07	S33	33
S01	S07	S34	15
S01	S07	S37	11
S01	S07	S43	13
S01	S07	S45	9
S01	S08	S02	57
S01	S08	S04	251
S01	S08	S05	87
S01	S08	S07	18
S01	S08	S09	45
S01	S08	S10	42
S01	S08	S11	1
S01	S08	S12	22
S01	S08	S13	3
S01	S08	S14	145
S01	S08	S15	185
S01	S08	S16	5
S01	S08	S18	16
S01	S08	S19	18
S01	S08	S20	78
S01	S08	S21	29
S01	S08	S22	5
S01	S08	S23	3
S01	S08	S28	51
S01	S08	S34	4
S01	S08	S36	25
S01	S08	S38	7
S01	S08	S39	120
S01	S08	S40	16
S01	S08	S41	16
S01	S08	S43	8
S01	S08	S44	58
S01	S08	S45	39
S01	S09	S08	45
S01	S09	S11	1
S01	S09	S14	16
S01	S09	S15	20
S01	S09	S16	21
S01	S09	S18	21
S01	S09	S20	6
S01	S09	S22	5
S01	S09	S23	16
S01	S09	S25	9
S01	S09	S26	7
S01	S09	S28	6
S01	S09	S31	5
S01	S09	S36	5
S01	S09	S39	10
S01	S09	S44	3
S01	S09	S45	3
S01	S10	S02	2
S01	S10	S03	36
S01	S10	S04	54
S01	S10	S05	86
S01	S10	S06	15
S01	S10	S08	42
S01	S10	S11	176
S01	S10	S12	188
S01	S10	S13	92
S01	S10	S14	10
S01	S10	S15	22
S01	S10	S16	128
S01	S10	S17	63
S01	S10	S18	42
S01	S10	S19	62
S01	S10	S23	28
S01	S10	S24	56
S01	S10	S25	5
S01	S10	S26	7
S01	S10	S27	36
S01	S10	S28	18
S01	S10	S30	13
S01	S10	S31	1
S01	S10	S33	29
S01	S10	S36	8
S01	S10	S37	9
S01	S10	S39	16
S01	S10	S44	4
S01	S11	S02	2
S01	S11	S03	26
S01	S11	S05	20
S01	S11	S06	7
S01	S11	S08	1
S01	S11	S09	1
S01	S11	S10	176
S01	S11	S12	90
S01	S11	S13	44
S01	S11	S14	1
S01	S11	S16	212
S01	S11	S17	21
S01	S11	S18	79
S01	S11	S19	93
S01	S11	S20	1
S01	S11	S23	59
S01	S11	S24	17
S01	S11	S25	7
S01	S11	S26	26
S01	S11	S27	60
S01	S11	S28	34
S01	S11	S31	15
S01	S11	S32	6
S01	S11	S33	9
S01	S11	S34	1
S01	S11	S36	18
S01	S12	S04	31
S01	S12	S05	62
S01	S12	S06	20
S01	S12	S08	22
S01	S12	S10	188
S01	S12	S11	90
S01	S12	S13	129
S01	S12	S14	6
S01	S12	S15	10
S01	S12	S16	57
S01	S12	S17	90
S01	S12	S18	6
S01	S12	S19	37
S01	S12	S22	6

S01	S12	S23	10	S01	S16	S12	57	S01	S19	S18	133
S01	S12	S24	70	S01	S16	S13	17	S01	S19	S21	76
S01	S12	S26	6	S01	S16	S14	5	S01	S19	S22	48
S01	S12	S27	20	S01	S16	S18	142	S01	S19	S23	42
S01	S12	S28	10	S01	S16	S19	118	S01	S19	S24	23
S01	S12	S30	13	S01	S16	S21	5	S01	S19	S25	22
S01	S12	S33	42	S01	S16	S22	5	S01	S19	S26	8
S01	S12	S34	6	S01	S16	S23	112	S01	S19	S27	153
S01	S12	S36	1	S01	S16	S25	17	S01	S19	S28	95
S01	S12	S37	11	S01	S16	S26	64	S01	S19	S30	9
S01	S12	S39	5	S01	S16	S27	76	S01	S19	S31	8
S01	S12	S43	10	S01	S16	S28	45	S01	S19	S32	6
S01	S13	S04	3	S01	S16	S31	32	S01	S19	S33	9
S01	S13	S05	28	S01	S16	S32	15	S01	S19	S35	13
S01	S13	S08	3	S01	S16	S34	13	S01	S19	S36	43
S01	S13	S10	92	S01	S16	S36	21	S01	S19	S40	6
S01	S13	S11	44	S01	S16	S42	7	S01	S19	S41	8
S01	S13	S12	129	S01	S17	S02	66	S01	S20	S04	50
S01	S13	S16	17	S01	S17	S04	9	S01	S20	S08	78
S01	S13	S17	150	S01	S17	S05	25	S01	S20	S09	6
S01	S13	S19	15	S01	S17	S07	131	S01	S20	S11	1
S01	S13	S22	6	S01	S17	S10	63	S01	S20	S14	101
S01	S13	S23	6	S01	S17	S11	21	S01	S20	S26	1
S01	S13	S24	121	S01	S17	S12	90	S01	S20	S28	139
S01	S13	S26	6	S01	S17	S13	150	S01	S20	S29	4
S01	S13	S27	5	S01	S17	S18	1	S01	S20	S31	22
S01	S13	S29	23	S01	S17	S19	1	S01	S20	S32	6
S01	S13	S30	41	S01	S17	S22	50	S01	S20	S34	27
S01	S13	S31	16	S01	S17	S23	38	S01	S20	S35	10
S01	S13	S33	64	S01	S17	S24	271	S01	S20	S36	98
S01	S13	S34	16	S01	S17	S26	31	S01	S20	S39	1
S01	S13	S37	20	S01	S17	S27	1	S01	S20	S40	42
S01	S13	S43	13	S01	S17	S28	1	S01	S20	S41	20
S01	S13	S45	1	S01	S17	S29	58	S01	S20	S43	6
S01	S14	S02	22	S01	S17	S30	92	S01	S20	S44	18
S01	S14	S04	81	S01	S17	S31	55	S01	S20	S45	13
S01	S14	S05	15	S01	S17	S33	134	S01	S21	S04	4
S01	S14	S07	9	S01	S17	S34	41	S01	S21	S08	29
S01	S14	S08	145	S01	S17	S37	40	S01	S21	S14	63
S01	S14	S09	16	S01	S17	S40	6	S01	S21	S16	5
S01	S14	S10	10	S01	S17	S43	45	S01	S21	S18	55
S01	S14	S11	1	S01	S17	S44	10	S01	S21	S19	76
S01	S14	S12	6	S01	S17	S45	24	S01	S21	S22	16
S01	S14	S16	5	S01	S18	S03	11	S01	S21	S23	16
S01	S14	S18	29	S01	S18	S05	12	S01	S21	S24	6
S01	S14	S19	42	S01	S18	S08	16	S01	S21	S25	7
S01	S14	S20	101	S01	S18	S09	21	S01	S22	S02	9
S01	S14	S21	63	S01	S18	S10	42	S01	S22	S07	26
S01	S14	S23	16	S01	S18	S11	79	S01	S22	S08	5
S01	S14	S25	7	S01	S18	S12	6	S01	S22	S09	5
S01	S14	S28	73	S01	S18	S14	29	S01	S22	S12	6
S01	S14	S31	8	S01	S18	S16	142	S01	S22	S13	6
S01	S14	S34	10	S01	S18	S17	1	S01	S22	S16	5
S01	S14	S36	43	S01	S18	S19	133	S01	S22	S17	50
S01	S14	S40	17	S01	S18	S21	55	S01	S22	S18	62
S01	S15	S02	27	S01	S18	S22	62	S01	S22	S19	48
S01	S15	S04	133	S01	S18	S23	194	S01	S22	S21	16
S01	S15	S05	54	S01	S18	S24	26	S01	S22	S23	40
S01	S15	S07	8	S01	S18	S25	51	S01	S22	S24	38
S01	S15	S08	185	S01	S18	S26	95	S01	S22	S26	31
S01	S15	S09	20	S01	S18	S27	34	S01	S22	S27	9
S01	S15	S10	22	S01	S18	S28	14	S01	S22	S28	5
S01	S15	S12	10	S01	S18	S30	19	S01	S22	S31	16
S01	S15	S33	4	S01	S18	S31	59	S01	S22	S33	15
S01	S15	S34	4	S01	S18	S32	31	S01	S22	S34	15
S01	S15	S38	17	S01	S18	S33	20	S01	S22	S43	3
S01	S15	S39	158	S01	S18	S34	16	S01	S23	S02	9
S01	S15	S40	13	S01	S18	S35	5	S01	S23	S03	7
S01	S15	S41	16	S01	S18	S42	10	S01	S23	S05	8
S01	S15	S43	17	S01	S18	S43	10	S01	S23	S07	19
S01	S15	S44	90	S01	S19	S02	2	S01	S23	S08	3
S01	S15	S45	64	S01	S19	S03	5	S01	S23	S09	16
S01	S16	S02	2	S01	S19	S08	18	S01	S23	S10	28
S01	S16	S03	17	S01	S19	S10	62	S01	S23	S11	59
S01	S16	S05	18	S01	S19	S11	93	S01	S23	S12	10
S01	S16	S06	5	S01	S19	S12	37	S01	S23	S13	6
S01	S16	S08	5	S01	S19	S13	15	S01	S23	S14	16
S01	S16	S09	21	S01	S19	S14	42	S01	S23	S16	112
S01	S16	S10	128	S01	S19	S16	118	S01	S23	S17	38
S01	S16	S11	212	S01	S19	S17	1	S01	S23	S18	194

S01	S23	S19	42	S01	S26	S39	5	S01	S30	S24	133
S01	S23	S21	16	S01	S26	S40	32	S01	S30	S25	33
S01	S23	S22	40	S01	S26	S41	9	S01	S30	S27	9
S01	S23	S25	69	S01	S26	S42	18	S01	S30	S29	108
S01	S23	S26	167	S01	S27	S10	36	S01	S30	S31	77
S01	S23	S27	9	S01	S27	S11	60	S01	S30	S33	32
S01	S23	S29	6	S01	S27	S12	20	S01	S30	S34	47
S01	S23	S30	23	S01	S27	S13	5	S01	S30	S39	2
S01	S23	S31	88	S01	S27	S16	76	S01	S30	S40	16
S01	S23	S32	40	S01	S27	S17	1	S01	S30	S43	19
S01	S23	S33	12	S01	S27	S18	34	S01	S30	S44	3
S01	S23	S34	51	S01	S27	S19	153	S01	S30	S45	8
S01	S23	S36	3	S01	S27	S22	9	S01	S31	S05	3
S01	S23	S38	6	S01	S27	S23	9	S01	S31	S07	14
S01	S23	S40	13	S01	S27	S25	9	S01	S31	S09	5
S01	S23	S41	2	S01	S27	S28	142	S01	S31	S10	1
S01	S23	S42	12	S01	S27	S30	9	S01	S31	S11	15
S01	S23	S43	11	S01	S27	S34	9	S01	S31	S13	16
S01	S24	S02	42	S01	S27	S35	18	S01	S31	S14	8
S01	S24	S04	4	S01	S27	S36	78	S01	S31	S16	32
S01	S24	S05	25	S01	S27	S40	32	S01	S31	S17	55
S01	S24	S07	86	S01	S27	S41	11	S01	S31	S18	59
S01	S24	S10	56	S01	S27	S42	16	S01	S31	S19	8
S01	S24	S11	17	S01	S27	S45	6	S01	S31	S20	22
S01	S24	S12	70	S01	S28	S02	2	S01	S31	S22	16
S01	S24	S13	121	S01	S28	S04	25	S01	S31	S23	88
S01	S24	S17	271	S01	S28	S08	51	S01	S31	S24	66
S01	S24	S18	26	S01	S28	S09	6	S01	S31	S26	107
S01	S24	S19	23	S01	S28	S10	18	S01	S31	S28	43
S01	S24	S21	6	S01	S28	S11	34	S01	S31	S29	119
S01	S24	S22	38	S01	S28	S12	10	S01	S31	S30	77
S01	S24	S29	92	S01	S28	S14	73	S01	S31	S32	83
S01	S24	S30	133	S01	S28	S16	45	S01	S31	S34	152
S01	S24	S31	66	S01	S28	S17	1	S01	S31	S36	61
S01	S24	S33	174	S01	S28	S18	14	S01	S31	S37	2
S01	S24	S34	45	S01	S28	S19	95	S01	S31	S38	15
S01	S24	S37	47	S01	S28	S20	139	S01	S31	S39	3
S01	S24	S39	2	S01	S28	S22	5	S01	S31	S40	107
S01	S24	S40	16	S01	S28	S26	8	S01	S31	S41	5
S01	S24	S43	62	S01	S28	S27	142	S01	S32	S11	6
S01	S24	S44	20	S01	S28	S29	14	S01	S32	S16	15
S01	S24	S45	37	S01	S28	S31	43	S01	S32	S18	31
S01	S25	S09	9	S01	S28	S32	10	S01	S32	S19	6
S01	S25	S10	5	S01	S28	S34	73	S01	S32	S20	6
S01	S25	S11	7	S01	S28	S35	39	S01	S32	S23	40
S01	S25	S14	7	S01	S28	S36	238	S01	S32	S25	6
S01	S25	S16	17	S01	S28	S39	3	S01	S32	S26	47
S01	S25	S18	51	S01	S28	S40	118	S01	S32	S28	10
S01	S25	S19	22	S01	S28	S41	45	S01	S32	S29	27
S01	S25	S21	7	S01	S28	S42	16	S01	S32	S31	83
S01	S25	S23	69	S01	S28	S43	9	S01	S32	S34	22
S01	S25	S27	9	S01	S28	S44	32	S01	S32	S36	11
S01	S25	S29	6	S01	S28	S45	27	S01	S32	S37	9
S01	S25	S30	33	S01	S29	S02	12	S01	S32	S38	39
S01	S25	S32	6	S01	S29	S07	20	S01	S32	S40	20
S01	S25	S33	21	S01	S29	S13	23	S01	S32	S45	4
S01	S25	S38	6	S01	S29	S17	58	S01	S33	S02	9
S01	S25	S43	11	S01	S29	S20	4	S01	S33	S05	12
S01	S26	S02	9	S01	S29	S23	6	S01	S33	S07	33
S01	S26	S03	1	S01	S29	S24	92	S01	S33	S10	29
S01	S26	S05	4	S01	S29	S25	6	S01	S33	S11	9
S01	S26	S07	15	S01	S29	S28	14	S01	S33	S12	42
S01	S26	S09	7	S01	S29	S30	108	S01	S33	S13	64
S01	S26	S10	7	S01	S29	S31	119	S01	S33	S15	4
S01	S26	S11	26	S01	S29	S32	27	S01	S33	S17	134
S01	S26	S12	6	S01	S29	S34	84	S01	S33	S18	20
S01	S26	S13	6	S01	S29	S36	23	S01	S33	S19	9
S01	S26	S16	64	S01	S29	S38	17	S01	S33	S22	15
S01	S26	S17	31	S01	S29	S40	46	S01	S33	S23	12
S01	S26	S18	95	S01	S29	S45	3	S01	S33	S24	174
S01	S26	S19	8	S01	S30	S02	20	S01	S33	S25	21
S01	S26	S20	1	S01	S30	S05	8	S01	S33	S30	32
S01	S26	S22	31	S01	S30	S07	31	S01	S33	S36	3
S01	S26	S23	167	S01	S30	S10	13	S01	S33	S37	54
S01	S26	S28	8	S01	S30	S12	13	S01	S33	S39	14
S01	S26	S31	107	S01	S30	S13	41	S01	S33	S41	4
S01	S26	S32	47	S01	S30	S17	92	S01	S33	S43	107
S01	S26	S34	78	S01	S30	S18	19	S01	S33	S44	44
S01	S26	S36	14	S01	S30	S19	9	S01	S33	S45	70
S01	S26	S38	10	S01	S30	S23	23	S01	S34	S02	9

S01	S34	S07	15	S01	S38	S29	17	S01	S42	S28	16
S01	S34	S08	4	S01	S38	S31	15	S01	S42	S34	26
S01	S34	S11	1	S01	S38	S32	39	S01	S42	S36	38
S01	S34	S12	6	S01	S38	S39	25	S01	S42	S40	51
S01	S34	S13	16	S01	S38	S44	32	S01	S42	S43	23
S01	S34	S14	10	S01	S38	S45	43	S01	S42	S45	44
S01	S34	S15	4	S01	S39	S02	10	S01	S43	S02	2
S01	S34	S16	13	S01	S39	S04	81	S01	S43	S07	13
S01	S34	S17	41	S01	S39	S05	38	S01	S43	S08	8
S01	S34	S18	16	S01	S39	S08	120	S01	S43	S12	10
S01	S34	S20	27	S01	S39	S09	10	S01	S43	S13	13
S01	S34	S22	15	S01	S39	S10	16	S01	S43	S15	17
S01	S34	S23	51	S01	S39	S12	5	S01	S43	S17	45
S01	S34	S24	45	S01	S39	S15	158	S01	S43	S18	10
S01	S34	S26	78	S01	S39	S20	1	S01	S43	S20	6
S01	S34	S27	9	S01	S39	S24	2	S01	S43	S22	3
S01	S34	S28	73	S01	S39	S26	5	S01	S43	S23	11
S01	S34	S29	84	S01	S39	S28	3	S01	S43	S24	62
S01	S34	S30	47	S01	S39	S30	2	S01	S43	S25	11
S01	S34	S31	152	S01	S39	S31	3	S01	S43	S28	9
S01	S34	S32	22	S01	S39	S33	14	S01	S43	S30	19
S01	S34	S36	107	S01	S39	S34	21	S01	S43	S33	107
S01	S34	S39	21	S01	S39	S35	10	S01	S43	S36	22
S01	S34	S40	195	S01	S39	S38	25	S01	S43	S39	36
S01	S34	S41	31	S01	S39	S40	36	S01	S43	S40	14
S01	S34	S42	26	S01	S39	S41	43	S01	S43	S41	13
S01	S35	S18	5	S01	S39	S43	36	S01	S43	S42	23
S01	S35	S19	13	S01	S39	S44	129	S01	S43	S44	79
S01	S35	S20	10	S01	S39	S45	96	S01	S43	S45	132
S01	S35	S27	18	S01	S40	S04	9	S01	S44	S04	33
S01	S35	S28	39	S01	S40	S08	16	S01	S44	S05	17
S01	S35	S39	10	S01	S40	S14	17	S01	S44	S08	58
S01	S35	S44	3	S01	S40	S15	13	S01	S44	S09	3
S01	S35	S45	2	S01	S40	S17	6	S01	S44	S10	4
S01	S36	S04	10	S01	S40	S19	6	S01	S44	S15	90
S01	S36	S08	25	S01	S40	S20	42	S01	S44	S17	10
S01	S36	S09	5	S01	S40	S23	13	S01	S44	S20	18
S01	S36	S10	8	S01	S40	S24	16	S01	S44	S24	20
S01	S36	S11	18	S01	S40	S26	32	S01	S44	S28	32
S01	S36	S12	1	S01	S40	S27	32	S01	S44	S30	3
S01	S36	S14	43	S01	S40	S28	118	S01	S44	S33	44
S01	S36	S16	21	S01	S40	S29	46	S01	S44	S35	3
S01	S36	S19	43	S01	S40	S30	16	S01	S44	S36	34
S01	S36	S20	98	S01	S40	S31	107	S01	S44	S38	32
S01	S36	S23	3	S01	S40	S32	20	S01	S44	S39	129
S01	S36	S26	14	S01	S40	S34	195	S01	S44	S41	38
S01	S36	S27	78	S01	S40	S36	178	S01	S44	S43	79
S01	S36	S28	238	S01	S40	S39	36	S01	S44	S45	169
S01	S36	S29	23	S01	S40	S41	51	S01	S45	S04	16
S01	S36	S31	61	S01	S40	S42	51	S01	S45	S05	6
S01	S36	S32	11	S01	S40	S43	14	S01	S45	S07	9
S01	S36	S33	3	S01	S40	S45	27	S01	S45	S08	39
S01	S36	S34	107	S01	S41	S02	3	S01	S45	S09	3
S01	S36	S40	178	S01	S41	S04	12	S01	S45	S13	1
S01	S36	S41	48	S01	S41	S08	16	S01	S45	S15	64
S01	S36	S42	38	S01	S41	S15	16	S01	S45	S17	24
S01	S36	S43	22	S01	S41	S19	8	S01	S45	S20	13
S01	S36	S44	34	S01	S41	S20	20	S01	S45	S24	37
S01	S36	S45	47	S01	S41	S23	2	S01	S45	S27	6
S01	S37	S02	3	S01	S41	S26	9	S01	S45	S28	27
S01	S37	S05	6	S01	S41	S27	11	S01	S45	S29	3
S01	S37	S07	11	S01	S41	S28	45	S01	S45	S30	8
S01	S37	S10	9	S01	S41	S31	5	S01	S45	S32	4
S01	S37	S12	11	S01	S41	S33	4	S01	S45	S33	70
S01	S37	S13	20	S01	S41	S34	31	S01	S45	S35	2
S01	S37	S17	40	S01	S41	S36	48	S01	S45	S36	47
S01	S37	S24	47	S01	S41	S39	43	S01	S45	S38	43
S01	S37	S31	2	S01	S41	S40	51	S01	S45	S39	96
S01	S37	S32	9	S01	S41	S43	13	S01	S45	S40	27
S01	S37	S33	54	S01	S41	S44	38	S01	S45	S41	28
S01	S38	S04	3	S01	S41	S45	28	S01	S45	S42	44
S01	S38	S08	7	S01	S42	S16	7	S01	S45	S43	132
S01	S38	S15	17	S01	S42	S18	10	S01	S45	S44	169
S01	S38	S23	6	S01	S42	S23	12				
S01	S38	S25	6	S01	S42	S26	18				
S01	S38	S26	10	S01	S42	S27	16				

Appendix B –Network Topology Files

Topology file for 10-node 25-span network

NODE	X	Y
N01	315.00	133.00
N02	113.00	185.00
N03	523.00	258.00
N04	291.00	377.00
N05	103.00	474.00
N06	628.00	525.00
N07	440.00	529.00
N08	279.00	646.00
N09	537.00	818.00
N10	174.00	866.00

SPAN	O	D	LENGTH	MTTF(h)	MTTR(h)	UA
S01	N01	N02	208.5857	14008.6296	12	0.0009
S02	N01	N03	242.6706	12041.0157	12	0.0010
S03	N01	N04	245.1775	11917.8969	12	0.0010
S04	N01	N05	401.5283	7277.1951	12	0.0016
S05	N01	N06	501.6303	5825.0065	12	0.0021
S06	N02	N05	289.1730	10104.6793	12	0.0012
S07	N02	N06	617.1102	4734.9728	12	0.0025
S08	N02	N09	761.8825	3835.2369	12	0.0031
S09	N03	N04	260.7393	11206.5946	12	0.0011
S10	N03	N06	286.9042	10184.5855	12	0.0012
S11	N03	N07	283.4255	10309.5884	12	0.0012
S12	N04	N05	211.5490	13812.3996	12	0.0009
S13	N04	N07	212.8497	13727.9960	12	0.0009
S14	N04	N08	269.2675	10851.6614	12	0.0011
S15	N04	N10	502.8021	5811.4310	12	0.0021
S16	N05	N08	246.0894	11873.7330	12	0.0010
S17	N05	N10	398.3780	7334.7431	12	0.0016
S18	N06	N07	188.0425	15539.0364	12	0.0008
S19	N06	N08	369.3806	7910.5408	12	0.0015
S20	N06	N09	306.8061	9523.9297	12	0.0013
S21	N07	N08	199.0226	14681.7488	12	0.0008
S22	N07	N09	304.8442	9585.2235	12	0.0013
S23	N08	N09	310.0774	9423.4533	12	0.0013
S24	N08	N10	243.7724	11986.5891	12	0.0010
S25	N09	N10	366.1598	7980.1223	12	0.0015

Appendix C –Network Demand Files

Demand file for 10-node 25-span network

DEMAND	O	D	NBUNITS
D01	N01	N02	2
D02	N01	N03	9
D03	N01	N04	4
D04	N01	N05	1
D05	N01	N06	3
D06	N01	N07	10
D07	N01	N08	1
D08	N01	N09	6
D09	N01	N10	6
D10	N02	N03	5
D11	N02	N04	7
D12	N02	N05	2
D13	N02	N06	8
D14	N02	N07	9
D15	N02	N08	10
D16	N02	N09	2
D17	N02	N10	1
D18	N03	N04	2
D19	N03	N05	1
D20	N03	N06	4
D21	N03	N07	3
D22	N03	N08	8
D23	N03	N09	2
D24	N03	N10	1
D25	N04	N05	9
D26	N04	N06	1
D27	N04	N07	9
D28	N04	N08	5
D29	N04	N09	9
D30	N04	N10	10
D31	N05	N06	4
D32	N05	N07	7
D33	N05	N08	4
D34	N05	N09	3
D35	N05	N10	5
D36	N06	N07	1
D37	N06	N08	10
D38	N06	N09	10
D39	N06	N10	4
D40	N07	N08	10
D41	N07	N09	7
D42	N07	N10	7
D43	N08	N09	8
D44	N08	N10	8
D45	N09	N10	10

Appendix D The MCSF Model

Appendix D.1 – AMPL codes of MCSF Model for Span Restoration Mechanism

```
# Span-Restoration mechanism under dual failure scenario
# March 2015 by Wenjing Wang

# ************************
# SETS
# ************************

set SPANS;
# set of all spans

set BACKUP_ROUTES{i in SPANS};
# set of all backup routes for each span failure i

# ************************
# PARAMETERS
# ************************

param Work{i in SPANS};
# amount of working capacities placed on span i

param Cost{k in SPANS};
# cost of each unit of capacity on span k

param Delta{i in SPANS, k in SPANS, b in BACKUP_ROUTES[i]} default 0;
# binary, takes 1 if backup route b for failure of span i crosses span k

# ************************
# VIRAIABLES
# ************************

var spare{k in SPANS} >=0 integer, <=10000;
# amount of spare capacity place on span j

var flow_single{i in SPANS, b in BACKUP_ROUTES[i]} >=0 integer, <=10000;
# flow through backup route b for failure of span i

# ************************
# OBJECTIVE FUNCTION
# ************************

minimize tot_cost:
sum{k in SPANS} spare[k] * Cost[k];
# minimize total cost under full single failure

# ************************
# CONSTRAINTS
# ************************

subject to c_13{i in SPANS}: sum{b in BACKUP_ROUTES[i]} flow_single[i, b] >= Work[i];
# guarantee enough restoration flow for full single failure restorability

subject to c_14{i in SPANS, k in SPANS: k <> i}:
  spare[k] >= sum{b in BACKUP_ROUTES[i]} (flow_single[i, b] * Delta[i, k, b]);
# translate flow requirements in c_13 to each span
```

Appendix D.2 – An Example of *.Dat Files of MCSF Model for Span Restoration Mechanism

```
# *.dat file for 20-node 35-span network
# MCSF model
# Created in January 2015 by Wenjing

set SPANS := S01 S02 S03 S04 S05 S07 S08 S10 S11 S12 S13 S14 S15
        S16 S17 S18 S19 S21 S22 S23 S26 S27 S28 S29 S30 S31
        S32 S33 S34 S35 S36 S37 S38 S39 S40;

param Cost :=
S01     139.560
S02     179.360
S03     116.181
S04     167.601
S05     50.606
S07     134.302
S08     227.002
S10     136.356
S11     152.506
S12     127.475
S13     148.772
S14     260.923
S15     92.779
S16     86.822
S17     124.631
S18     145.055
S19     131.320
S21     99.705
S22     131.187
S23     173.118
S26     198.497
S27     82.970
S28     128.725
S29     166.066
S30     151.427
S31     130.173
S32     64.070
S33     204.924
S34     108.074
S35     104.805
S36     54.129
S37     121.037
S38     86.833
S39     156.984
S40     120.150;

param Work :=
S01     87
S02     69
S03     200
S04     26
S05     31
S07     2
S08     65
S10     63
S11     60
S12     154
S13     68
S14     20
S15     50
S16     85
S17     133
S18     36
S19     99
S21     236
S22     86
S23     73
S26     69
S27     52
S28     66
S29     71
S30     43
S31     22
S32     137
S33     37
S34     76
S35     126
S36     85
S37     47
S38     120
S39     85
S40     114;

set BACKUP_ROUTES[S01] := R1 R2 R3 R4 R5 R6 R7;
set BACKUP_ROUTES[S02] := R8 R9 R10 R11 R12 R13 R14;
set BACKUP_ROUTES[S03] := R15 R16 R17 R18 R19 R20 R21;
set BACKUP_ROUTES[S04] := R22 R23 R24 R25 R26 R27 R28;
set BACKUP_ROUTES[S05] := R29 R30 R31 R32 R33 R34 R35;
set BACKUP_ROUTES[S07] := R36 R37 R38 R39 R40 R41 R42;
set BACKUP_ROUTES[S08] := R43 R44 R45 R46 R47 R48 R49;
set BACKUP_ROUTES[S10] := R50 R51 R52 R53 R54 R55 R56;
set BACKUP_ROUTES[S11] := R57 R58 R59 R60 R61 R62 R63;
set BACKUP_ROUTES[S12] := R64 R65 R66 R67 R68 R69 R70;
set BACKUP_ROUTES[S13] := R71 R72 R73 R74 R75 R76 R77;
set BACKUP_ROUTES[S14] := R78 R79 R80 R81 R82 R83 R84;
set BACKUP_ROUTES[S15] := R85 R86 R87 R88 R89 R90 R91;
set BACKUP_ROUTES[S16] := R92 R93 R94 R95 R96 R97 R98;
set BACKUP_ROUTES[S17] := R99 R100 R101 R102 R103 R104 R105;
set BACKUP_ROUTES[S18] := R106 R107 R108 R109 R110 R111 R112;
set BACKUP_ROUTES[S19] := R113 R114 R115 R116 R117 R118 R119;
set BACKUP_ROUTES[S21] := R120 R121 R122 R123 R124 R125 R126;
set BACKUP_ROUTES[S22] := R127 R128 R129 R130 R131 R132 R133;
set BACKUP_ROUTES[S23] := R134 R135 R136 R137 R138 R139 R140;
set BACKUP_ROUTES[S26] := R141 R142 R143 R144 R145 R146 R147;
set BACKUP_ROUTES[S27] := R148 R149 R150 R151 R152 R153 R154;
set BACKUP_ROUTES[S28] := R155 R156 R157 R158 R159 R160 R161;
set BACKUP_ROUTES[S29] := R162 R163 R164 R165 R166 R167 R168;
set BACKUP_ROUTES[S30] := R169 R170 R171 R172 R173 R174 R175;
set BACKUP_ROUTES[S31] := R176 R177 R178 R179 R180 R181 R182;
set BACKUP_ROUTES[S32] := R183 R184 R185 R186 R187 R188 R189;
set BACKUP_ROUTES[S33] := R190 R191 R192 R193 R194 R195 R196;
set BACKUP_ROUTES[S34] := R197 R198 R199 R200 R201 R202 R203;
set BACKUP_ROUTES[S35] := R204 R205 R206 R207 R208 R209 R210;
set BACKUP_ROUTES[S36] := R211 R212 R213 R214 R215 R216 R217;
set BACKUP_ROUTES[S37] := R218 R219 R220 R221 R222 R223 R224;
set BACKUP_ROUTES[S38] := R225 R226 R227 R228 R229 R230 R231;
set BACKUP_ROUTES[S39] := R232 R233 R234 R235 R236 R237 R238;
set BACKUP_ROUTES[S40] := R239 R240 R241 R242 R243 R244 R245;

param Delta :=
[S01, *, R1] S02 1 S05 1
[S01, *, R2] S02 1 S04 1 S07 1
[S01, *, R3] S02 1 S04 1 S08 1 S10 1
[S01, *, R4] S03 1 S05 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R5] S03 1 S04 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R6] S03 1 S05 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R7] S03 1 S05 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S02, *, R8] S01 1 S05 1
[S02, *, R9] S01 1 S04 1 S07 1
[S02, *, R10] S01 1 S04 1 S08 1 S10 1
[S02, *, R11] S03 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
```

[S02, *, R12] S03 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S02, *, R13] S03 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S02, *, R14] S03 1 S08 1 S11 1 S14 1 S32 1 S35 1
[S03, *, R15] S02 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R16] S01 1 S05 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R17] S01 1 S04 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R18] S02 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R19] S02 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S03, *, R20] S01 1 S05 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R21] S01 1 S05 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S04, *, R22] S05 1 S07 1
[S04, *, R23] S05 1 S08 1 S10 1
[S04, *, R24] S01 1 S02 1 S07 1
[S04, *, R25] S01 1 S02 1 S08 1 S10 1
[S04, *, R26] S01 1 S03 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S04, *, R27] S01 1 S03 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S04, *, R28] S01 1 S03 1 S10 1 S11 1 S14 1 S32 1 S35 1
[S05, *, R29] S04 1 S07 1
[S05, *, R30] S01 1 S02 1
[S05, *, R31] S04 1 S08 1 S10 1
[S05, *, R32] S01 1 S03 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S05, *, R33] S01 1 S03 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S05, *, R34] S01 1 S03 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S05, *, R35] S01 1 S03 1 S08 1 S11 1 S14 1 S32 1 S35 1
[S07, *, R36] S04 1 S05 1
[S07, *, R37] S08 1 S10 1
[S07, *, R38] S01 1 S02 1 S04 1
[S07, *, R39] S02 1 S03 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S07, *, R40] S01 1 S03 1 S05 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S07, *, R41] S02 1 S03 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S07, *, R42] S01 1 S03 1 S05 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S08, *, R43] S07 1 S10 1
[S08, *, R44] S04 1 S05 1 S10 1
[S08, *, R45] S01 1 S02 1 S04 1 S10 1
[S08, *, R46] S02 1 S03 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S08, *, R47] S01 1 S03 1 S05 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S08, *, R48] S02 1 S03 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S08, *, R49] S01 1 S03 1 S05 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S10, *, R50] S07 1 S08 1
[S10, *, R51] S04 1 S05 1 S08 1
[S10, *, R52] S01 1 S02 1 S04 1 S08 1
[S10, *, R53] S01 1 S03 1 S04 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R54] S02 1 S03 1 S07 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R55] S01 1 S03 1 S05 1 S07 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R56] S01 1 S03 1 S04 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S11, *, R57] S12 1 S15 1
[S11, *, R58] S12 1 S13 1 S17 1 S22 1
[S11, *, R59] S12 1 S14 1 S16 1 S19 1
[S11, *, R60] S12 1 S13 1 S16 1 S18 1 S22 1
[S11, *, R61] S12 1 S14 1 S17 1 S21 1 S31 1
[S11, *, R62] S12 1 S14 1 S17 1 S21 1 S32 1 S34 1
[S11, *, R63] S12 1 S14 1 S17 1 S18 1 S19 1
[S12, *, R64] S11 1 S15 1
[S12, *, R65] S11 1 S13 1 S17 1 S22 1
[S12, *, R66] S11 1 S14 1 S16 1 S19 1
[S12, *, R67] S11 1 S13 1 S16 1 S18 1 S22 1
[S12, *, R68] S11 1 S14 1 S17 1 S21 1 S31 1
[S12, *, R69] S11 1 S14 1 S17 1 S21 1 S32 1 S34 1
[S12, *, R70] S11 1 S14 1 S17 1 S18 1 S19 1
[S13, *, R71] S15 1 S17 1 S22 1
[S13, *, R72] S15 1 S16 1 S18 1 S22 1
[S13, *, R73] S11 1 S12 1 S17 1 S22 1
[S13, *, R74] S14 1 S21 1 S22 1 S31 1
[S13, *, R75] S11 1 S12 1 S16 1 S18 1 S22 1
[S13, *, R76] S14 1 S21 1 S22 1 S32 1 S34 1
[S13, *, R77] S14 1 S18 1 S19 1 S22 1
[S14, *, R78] S15 1 S16 1 S19 1
[S14, *, R79] S15 1 S17 1 S21 1 S31 1
[S14, *, R80] S15 1 S17 1 S21 1 S32 1 S34 1
[S14, *, R81] S15 1 S17 1 S18 1 S19 1
[S14, *, R82] S11 1 S12 1 S16 1 S19 1
[S14, *, R83] S13 1 S21 1 S22 1 S31 1
[S14, *, R84] S13 1 S21 1 S22 1 S32 1 S34 1
[S15, *, R85] S11 1 S12 1
[S15, *, R86] S13 1 S17 1 S22 1
[S15, *, R87] S14 1 S16 1 S19 1
[S15, *, R88] S13 1 S16 1 S18 1 S22 1
[S15, *, R89] S14 1 S17 1 S21 1 S31 1
[S15, *, R90] S14 1 S17 1 S21 1 S32 1 S34 1
[S15, *, R91] S14 1 S17 1 S18 1 S19 1
[S16, *, R92] S17 1 S18 1
[S16, *, R93] S14 1 S15 1 S19 1
[S16, *, R94] S17 1 S19 1 S21 1 S31 1
[S16, *, R95] S13 1 S15 1 S18 1 S22 1
[S16, *, R96] S17 1 S19 1 S21 1 S32 1 S34 1
[S16, *, R97] S11 1 S12 1 S14 1 S19 1
[S16, *, R98] S11 1 S12 1 S13 1 S18 1 S22 1
[S17, *, R99] S16 1 S18 1
[S17, *, R100] S13 1 S15 1 S22 1
[S17, *, R101] S16 1 S19 1 S21 1 S31 1
[S17, *, R102] S16 1 S19 1 S21 1 S32 1 S34 1
[S17, *, R103] S11 1 S12 1 S13 1 S22 1
[S17, *, R104] S14 1 S15 1 S21 1 S31 1
[S17, *, R105] S14 1 S15 1 S21 1 S32 1 S34 1
[S18, *, R106] S16 1 S17 1
[S18, *, R107] S19 1 S21 1 S31 1
[S18, *, R108] S19 1 S21 1 S32 1 S34 1
[S18, *, R109] S13 1 S15 1 S16 1 S22 1
[S18, *, R110] S19 1 S21 1 S30 1 S32 1 S36 1 S38 1
[S18, *, R111] S14 1 S15 1 S17 1 S19 1
[S18, *, R112] S19 1 S21 1 S28 1 S32 1 S33 1
[S19, *, R113] S18 1 S21 1 S31 1
[S19, *, R114] S18 1 S21 1 S32 1 S34 1
[S19, *, R115] S14 1 S15 1 S16 1
[S19, *, R116] S16 1 S17 1 S21 1 S31 1
[S19, *, R117] S16 1 S17 1 S21 1 S32 1 S34 1
[S19, *, R118] S18 1 S21 1 S30 1 S32 1 S36 1 S38 1
[S19, *, R119] S14 1 S15 1 S17 1 S18 1
[S21, *, R120] S18 1 S19 1 S31 1
[S21, *, R121] S18 1 S19 1 S32 1 S34 1
[S21, *, R122] S16 1 S17 1 S19 1 S31 1
[S21, *, R123] S16 1 S17 1 S19 1 S32 1 S34 1
[S21, *, R124] S14 1 S15 1 S17 1 S31 1
[S21, *, R125] S18 1 S19 1 S30 1 S32 1 S36 1 S38 1
[S21, *, R126] S14 1 S15 1 S17 1 S32 1 S34 1
[S22, *, R127] S13 1 S15 1 S17 1
[S22, *, R128] S13 1 S15 1 S16 1 S18 1
[S22, *, R129] S11 1 S12 1 S13 1 S17 1
[S22, *, R130] S13 1 S14 1 S21 1 S31 1
[S22, *, R131] S11 1 S12 1 S13 1 S16 1 S18 1
[S22, *, R132] S13 1 S14 1 S21 1 S32 1 S34 1
[S22, *, R133] S13 1 S14 1 S18 1 S19 1
[S23, *, R134] S21 1 S22 1 S26 1 S29 1 S37 1
[S23, *, R135] S21 1 S22 1 S26 1 S34 1 S38 1 S39 1
[S23, *, R136] S21 1 S22 1 S26 1 S27 1 S30 1 S37 1
[S23, *, R137] S21 1 S22 1 S26 1 S30 1 S36 1 S39 1
[S23, *, R138] S21 1 S22 1 S26 1 S31 1 S32 1 S38 1 S39 1
[S23, *, R139] S21 1 S22 1 S26 1 S27 1 S34 1 S36 1 S37 1 S38 1
[S23, *, R140] S21 1 S22 1 S26 1 S27 1 S29 1 S36 1 S39 1
[S26, *, R141] S21 1 S22 1 S23 1 S29 1 S37 1
[S26, *, R142] S21 1 S22 1 S23 1 S34 1 S38 1 S39 1
[S26, *, R143] S21 1 S22 1 S23 1 S27 1 S30 1 S37 1
[S26, *, R144] S21 1 S22 1 S23 1 S30 1 S36 1 S39 1
[S26, *, R145] S21 1 S22 1 S23 1 S31 1 S32 1 S38 1 S39 1
[S26, *, R146] S21 1 S22 1 S23 1 S27 1 S34 1 S36 1 S37 1 S38 1
[S26, *, R147] S21 1 S22 1 S23 1 S27 1 S29 1 S36 1 S39 1
[S27, *, R148] S29 1 S30 1
[S27, *, R149] S36 1 S37 1 S39 1
[S27, *, R150] S29 1 S34 1 S36 1 S38 1
[S27, *, R151] S29 1 S31 1 S32 1 S36 1 S38 1
[S27, *, R152] S29 1 S34 1 S35 1 S36 1 S40 1
[S27, *, R153] S30 1 S34 1 S37 1 S38 1 S39 1
[S27, *, R154] S29 1 S31 1 S32 1 S35 1 S36 1 S40 1
[S28, *, R155] S33 1 S34 1
[S28, *, R156] S31 1 S32 1 S33 1
[S28, *, R157] S30 1 S33 1 S36 1 S38 1
[S28, *, R158] S27 1 S29 1 S33 1 S36 1 S38 1
[S28, *, R159] S30 1 S33 1 S35 1 S36 1 S40 1
[S28, *, R160] S18 1 S19 1 S21 1 S32 1 S33 1
[S28, *, R161] S16 1 S17 1 S19 1 S21 1 S32 1 S33 1
[S29, *, R162] S27 1 S30 1
[S29, *, R163] S27 1 S34 1 S36 1 S38 1
[S29, *, R164] S27 1 S31 1 S32 1 S36 1 S38 1
[S29, *, R165] S27 1 S34 1 S35 1 S36 1 S40 1
[S29, *, R166] S34 1 S37 1 S38 1 S39 1
[S29, *, R167] S30 1 S36 1 S37 1 S39 1
[S29, *, R168] S27 1 S31 1 S32 1 S35 1 S36 1 S40 1
[S30, *, R169] S34 1 S36 1 S38 1
[S30, *, R170] S27 1 S29 1
[S30, *, R171] S31 1 S32 1 S36 1 S38 1

[S30, *, R172] S34 1 S35 1 S36 1 S40 1
[S30, *, R173] S31 1 S32 1 S35 1 S36 1 S40 1
[S30, *, R174] S28 1 S33 1 S36 1 S38 1
[S30, *, R175] S29 1 S36 1 S37 1 S39 1
[S31, *, R176] S32 1 S34 1
[S31, *, R177] S30 1 S32 1 S36 1 S38 1
[S31, *, R178] S18 1 S19 1 S21 1
[S31, *, R179] S28 1 S32 1 S33 1
[S31, *, R180] S16 1 S17 1 S19 1 S21 1
[S31, *, R181] S27 1 S29 1 S32 1 S36 1 S38 1
[S31, *, R182] S30 1 S32 1 S35 1 S36 1 S40 1
[S32, *, R183] S31 1 S34 1
[S32, *, R184] S30 1 S31 1 S36 1 S38 1
[S32, *, R185] S28 1 S31 1 S33 1
[S32, *, R186] S18 1 S19 1 S21 1 S34 1
[S32, *, R187] S27 1 S29 1 S31 1 S36 1 S38 1
[S32, *, R188] S16 1 S17 1 S19 1 S21 1 S34 1
[S32, *, R189] S30 1 S31 1 S35 1 S36 1 S40 1
[S33, *, R190] S28 1 S34 1
[S33, *, R191] S28 1 S31 1 S32 1
[S33, *, R192] S28 1 S30 1 S36 1 S38 1
[S33, *, R193] S27 1 S28 1 S29 1 S36 1 S38 1
[S33, *, R194] S28 1 S30 1 S35 1 S36 1 S40 1
[S33, *, R195] S18 1 S19 1 S21 1 S28 1 S32 1
[S33, *, R196] S16 1 S17 1 S19 1 S21 1 S28 1 S32 1
[S34, *, R197] S31 1 S32 1
[S34, *, R198] S30 1 S36 1 S38 1
[S34, *, R199] S28 1 S33 1
[S34, *, R200] S27 1 S29 1 S36 1 S38 1
[S34, *, R201] S30 1 S35 1 S36 1 S40 1
[S34, *, R202] S18 1 S19 1 S21 1 S32 1
[S34, *, R203] S16 1 S17 1 S19 1 S21 1 S32 1
[S35, *, R204] S38 1 S40 1
[S35, *, R205] S30 1 S34 1 S36 1 S40 1
[S35, *, R206] S30 1 S31 1 S32 1 S36 1 S40 1
[S35, *, R207] S27 1 S29 1 S34 1 S36 1 S40 1
[S35, *, R208] S27 1 S29 1 S31 1 S32 1 S36 1 S40 1
[S35, *, R209] S28 1 S30 1 S33 1 S36 1 S40 1
[S35, *, R210] S29 1 S34 1 S37 1 S39 1 S40 1
[S36, *, R211] S30 1 S34 1 S38 1
[S36, *, R212] S27 1 S37 1 S39 1
[S36, *, R213] S30 1 S31 1 S32 1 S38 1
[S36, *, R214] S27 1 S29 1 S34 1 S38 1
[S36, *, R215] S30 1 S34 1 S35 1 S40 1
[S36, *, R216] S27 1 S29 1 S31 1 S32 1 S38 1
[S36, *, R217] S30 1 S31 1 S32 1 S35 1 S40 1
[S37, *, R218] S27 1 S36 1 S39 1
[S37, *, R219] S29 1 S34 1 S38 1 S39 1
[S37, *, R220] S29 1 S30 1 S36 1 S39 1
[S37, *, R221] S27 1 S30 1 S34 1 S38 1 S39 1
[S37, *, R222] S29 1 S31 1 S32 1 S38 1 S39 1
[S37, *, R223] S29 1 S34 1 S35 1 S39 1 S40 1
[S37, *, R224] S27 1 S30 1 S31 1 S32 1 S38 1 S39 1
[S38, *, R225] S35 1 S40 1
[S38, *, R226] S30 1 S34 1 S36 1
[S38, *, R227] S30 1 S31 1 S32 1 S36 1
[S38, *, R228] S27 1 S29 1 S34 1 S36 1
[S38, *, R229] S27 1 S29 1 S31 1 S32 1 S36 1
[S38, *, R230] S28 1 S30 1 S33 1 S36 1
[S38, *, R231] S29 1 S34 1 S37 1 S39 1
[S39, *, R232] S27 1 S36 1 S37 1
[S39, *, R233] S29 1 S34 1 S37 1 S38 1
[S39, *, R234] S29 1 S30 1 S36 1 S37 1
[S39, *, R235] S27 1 S30 1 S34 1 S37 1 S38 1
[S39, *, R236] S29 1 S31 1 S32 1 S37 1 S38 1
[S39, *, R237] S29 1 S34 1 S35 1 S37 1 S40 1
[S39, *, R238] S27 1 S30 1 S31 1 S32 1 S37 1 S38 1
[S40, *, R239] S35 1 S38 1
[S40, *, R240] S30 1 S34 1 S35 1 S36 1
[S40, *, R241] S30 1 S31 1 S32 1 S35 1 S36 1
[S40, *, R242] S27 1 S29 1 S34 1 S35 1 S36 1
[S40, *, R243] S27 1 S29 1 S31 1 S32 1 S35 1 S36 1
[S40, *, R244] S28 1 S30 1 S33 1 S35 1 S36 1
[S40, *, R245] S29 1 S34 1 S35 1 S37 1 S39 1;

Appendix E The MDNF Model

Appendix E.1 – AMPL codes of MDNF Model for Span Restoration Mechanism

```
# Span-Restoration mechanism under dual failure scenario
# March 2015 by Wenjing Wang

# ************************
# SETS
# ************************

set SPANS;
# set of all spans

set BACKUP_ROUTES{i in SPANS};
# set of all backup routes for each span failure i

# ************************
# PARAMETERS
# ************************

param Work{i in SPANS};
# amount of working capacity placed on span i

param Cost{k in SPANS};
# cost of each unit of capacity on span k

param Delta{i in SPANS, k in SPANS, b in BACKUP_ROUTES[i]} default 0;
# binary, takes 1 if backup route b for failure of span i crosses span k

param Cinfinit;
# a positive large constant

param Budget;
# budget limit for dual failure restoration

# ************************
# VIRAIABLES
# ************************

var spare{k in SPANS} >=0 integer, <=10000;
# amount of spare capacity placeed on span j

var flow_single{i in SPANS, b in BACKUP_ROUTES[i]} >=0 integer, <=10000;
# flow through backup route b for failure of span i

var flow_dual_i{i in SPANS, j in SPANS, b in BACKUP_ROUTES[i]: i <> j} >=0 integer, <=10000;
# flow through backup route b for failure of span i under dual failure (i, j)

var flow_dual_j{i in SPANS, j in SPANS, b in BACKUP_ROUTES[j]: i <> j} >=0 integer, <=10000;
# flow through backup route b for failure of span j under dual failure (i, j)

var non_restored{i in SPANS, j in SPANS: i <> j} >=0 integer;
# amount of non-restored working capacity under dual failure (i, j)

var non_restored_i{i in SPANS, j in SPANS: i <> j} >=0 integer;
# amount of non-restored working capacity on span i under dual failure (i, j)

var non_restored_j{i in SPANS, j in SPANS: i <> j} >=0 integer;
# amount of non-restored working capacity on span j under dual failure (i, j)

# ************************
# OBJECTIVE FUNCTION
# ************************

minimize tot_non_restored:
sum{i in SPANS, j in SPANS: i <> j} non_restored[i, j];
# minimize amount of non-restored working capacity under all dual failure scenarios

# ************************
# CONSTRAINTS
# ************************

subject to c_12{i in SPANS, j in SPANS: i <> j}:
  non_restored[i, j] = Work[i] + Work[j] - sum{b in BACKUP_ROUTES[i]} flow_dual_i[i, j, b] - sum{b in BACKUP_ROUTES[j]} flow_dual_j[i, j, b];
# define non-restored working capacities under each dual failure (i, j)

subject to c_1201{i in SPANS, j in SPANS: i <> j}:
  non_restored_i[i, j] = Work[i] - sum{b in BACKUP_ROUTES[i]} flow_dual_i[i, j, b];
```

```
# define non-restored working capacities on span i under each dual failure (i, j)

subject to c_1202{i in SPANS, j in SPANS: i <> j}:
  non_restored_j[i, j] = Work[j] - sum{b in BACKUP_ROUTES[j]} flow_dual_j[i, j, b];
# define non-restored working capacities on span j under each dual failure (i, j)

subject to c_13{i in SPANS}: sum{b in BACKUP_ROUTES[i]} flow_single[i, b] >= Work[i];
# guarantee enough restoration flow for full single failure restorability

subject to c_14{i in SPANS, k in SPANS: k <> i}:
  spare[k] >= sum{b in BACKUP_ROUTES[i]} (flow_single[i, b] * Delta[i, k, b]);
# translate flow requirements in c_13 to each span

subject to c_15{i in SPANS, j in SPANS: i <> j}:
  sum{b in BACKUP_ROUTES[i]} flow_dual_i[i, j, b] <= Work[i];
# assign restoration flow to failed span i under dual failure scenario (i, j)

subject to c_16{i in SPANS, j in SPANS: i <> j}:
  sum{b in BACKUP_ROUTES[j]} flow_dual_j[i, j, b] <= Work[j];
# assign restoration flow to failed span j under dual failure scenario (i, j)

subject to c_17{i in SPANS, j in SPANS, k in SPANS: i <> j and j <> k and i <> k}:
  spare[k] >= sum{b in BACKUP_ROUTES[i]} (flow_dual_i[i, j, b] * Delta[i, k, b]) +
            sum{b in BACKUP_ROUTES[j]} (flow_dual_j[i, j, b] * Delta[j, k, b]);
# translate flow requirements in (15)(16) on each span

subject to c_18{i in SPANS, j in SPANS, b in BACKUP_ROUTES[i]: i <> j}:
  flow_dual_i[i, j, b] <= Cinfinit * (1 - Delta[i, j, b]);
# put limitation on restoration of failed span i under dual failure (i, j)

subject to c_19{i in SPANS, j in SPANS, b in BACKUP_ROUTES[j]: i <> j}:
  flow_dual_j[i, j, b] <= Cinfinit * (1 - Delta[j, i, b]);
# put limitation on restoration of failed span j under dual failure (i, j)

subject to c_20:
  sum{k in SPANS} (Cost[k] * spare[k]) <= Budget;
# put budget limitation on dual failure restoration
```

Appendix E.2 – An Example of *.Dat Files of MDNF Model for Span Restoration Mechanism

```
# *.dat file for 20-node 35-span network
# MCSF model
# Created in January 2015 by Wenjing

set SPANS := S01 S02 S03 S04 S05 S07 S08 S10 S11 S12 S13 S14 S15
        S16 S17 S18 S19 S21 S22 S23 S26 S27 S28 S29 S30 S31
        S32 S33 S34 S35 S36 S37 S38 S39 S40;

param Cost :=
S01     139.560
S02     179.360
S03     116.181
S04     167.601
S05     50.606
S07     134.302
S08     227.002
S10     136.356
S11     152.506
S12     127.475
S13     148.772
S14     260.923
S15     92.779
S16     86.822
S17     124.631
S18     145.055
S19     131.320
S21     99.705
S22     131.187
S23     173.118
S26     198.497
S27     82.970
S28     128.725
S29     166.066
S30     151.427
S31     130.173
S32     64.070
S33     204.924
S34     108.074
S35     104.805
S36     54.129
S37     121.037
S38     86.833
S39     156.984
S40     120.150;

param Work :=
S01     87
S02     69
S03     200
S04     26
S05     31
S07     2
S08     65
S10     63
S11     60
S12     154
S13     68
S14     20
S15     50
S16     85
S17     133
S18     36
S19     99
S21     236
S22     86
S23     73
S26     69
S27     52
S28     66
S29     71
S30     43
S31     22
S32     137
S33     37
S34     76
S35     126
S36     85
S37     47
S38     120
S39     85
S40     114;


set BACKUP_ROUTES[S01] := R1 R2 R3 R4 R5 R6 R7;
set BACKUP_ROUTES[S02] := R8 R9 R10 R11 R12 R13 R14;
set BACKUP_ROUTES[S03] := R15 R16 R17 R18 R19 R20 R21;
set BACKUP_ROUTES[S04] := R22 R23 R24 R25 R26 R27 R28;
set BACKUP_ROUTES[S05] := R29 R30 R31 R32 R33 R34 R35;
set BACKUP_ROUTES[S07] := R36 R37 R38 R39 R40 R41 R42;
set BACKUP_ROUTES[S08] := R43 R44 R45 R46 R47 R48 R49;
set BACKUP_ROUTES[S10] := R50 R51 R52 R53 R54 R55 R56;
set BACKUP_ROUTES[S11] := R57 R58 R59 R60 R61 R62 R63;
set BACKUP_ROUTES[S12] := R64 R65 R66 R67 R68 R69 R70;
set BACKUP_ROUTES[S13] := R71 R72 R73 R74 R75 R76 R77;
set BACKUP_ROUTES[S14] := R78 R79 R80 R81 R82 R83 R84;
set BACKUP_ROUTES[S15] := R85 R86 R87 R88 R89 R90 R91;
set BACKUP_ROUTES[S16] := R92 R93 R94 R95 R96 R97 R98;
set BACKUP_ROUTES[S17] := R99 R100 R101 R102 R103 R104 R105;
set BACKUP_ROUTES[S18] := R106 R107 R108 R109 R110 R111 R112;
set BACKUP_ROUTES[S19] := R113 R114 R115 R116 R117 R118 R119;
set BACKUP_ROUTES[S21] := R120 R121 R122 R123 R124 R125 R126;
set BACKUP_ROUTES[S22] := R127 R128 R129 R130 R131 R132 R133;
set BACKUP_ROUTES[S23] := R134 R135 R136 R137 R138 R139 R140;
set BACKUP_ROUTES[S26] := R141 R142 R143 R144 R145 R146 R147;
set BACKUP_ROUTES[S27] := R148 R149 R150 R151 R152 R153 R154;
set BACKUP_ROUTES[S28] := R155 R156 R157 R158 R159 R160 R161;
set BACKUP_ROUTES[S29] := R162 R163 R164 R165 R166 R167 R168;
set BACKUP_ROUTES[S30] := R169 R170 R171 R172 R173 R174 R175;
set BACKUP_ROUTES[S31] := R176 R177 R178 R179 R180 R181 R182;
set BACKUP_ROUTES[S32] := R183 R184 R185 R186 R187 R188 R189;
set BACKUP_ROUTES[S33] := R190 R191 R192 R193 R194 R195 R196;
set BACKUP_ROUTES[S34] := R197 R198 R199 R200 R201 R202 R203;
set BACKUP_ROUTES[S35] := R204 R205 R206 R207 R208 R209 R210;
set BACKUP_ROUTES[S36] := R211 R212 R213 R214 R215 R216 R217;
set BACKUP_ROUTES[S37] := R218 R219 R220 R221 R222 R223 R224;
set BACKUP_ROUTES[S38] := R225 R226 R227 R228 R229 R230 R231;
set BACKUP_ROUTES[S39] := R232 R233 R234 R235 R236 R237 R238;
set BACKUP_ROUTES[S40] := R239 R240 R241 R242 R243 R244 R245;

param Delta :=
[S01, *, R1] S02 1 S05 1
[S01, *, R2] S02 1 S04 1 S07 1
[S01, *, R3] S02 1 S04 1 S08 1 S10 1
[S01, *, R4] S03 1 S05 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R5] S03 1 S04 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R6] S03 1 S05 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R7] S03 1 S05 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S02, *, R8] S01 1 S05 1
[S02, *, R9] S01 1 S04 1 S07 1
```

[S02, *, R10] S01 1 S04 1 S08 1 S10 1
[S02, *, R11] S03 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S02, *, R12] S03 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S02, *, R13] S03 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S02, *, R14] S03 1 S08 1 S11 1 S14 1 S32 1 S35 1
[S03, *, R15] S02 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R16] S01 1 S05 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R17] S01 1 S04 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R18] S02 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R19] S02 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S03, *, R20] S01 1 S05 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R21] S01 1 S05 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S04, *, R22] S05 1 S07 1
[S04, *, R23] S05 1 S08 1 S10 1
[S04, *, R24] S01 1 S02 1 S07 1
[S04, *, R25] S01 1 S02 1 S08 1 S10 1
[S04, *, R26] S01 1 S03 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S04, *, R27] S01 1 S03 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S04, *, R28] S01 1 S03 1 S10 1 S11 1 S14 1 S32 1 S35 1
[S05, *, R29] S04 1 S07 1
[S05, *, R30] S01 1 S02 1
[S05, *, R31] S04 1 S08 1 S10 1
[S05, *, R32] S01 1 S03 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S05, *, R33] S01 1 S03 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S05, *, R34] S01 1 S03 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S05, *, R35] S01 1 S03 1 S08 1 S11 1 S14 1 S32 1 S35 1
[S07, *, R36] S04 1 S05 1
[S07, *, R37] S08 1 S10 1
[S07, *, R38] S01 1 S02 1 S04 1
[S07, *, R39] S02 1 S03 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S07, *, R40] S01 1 S03 1 S05 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S07, *, R41] S02 1 S03 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S07, *, R42] S01 1 S03 1 S05 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S08, *, R43] S07 1 S10 1
[S08, *, R44] S04 1 S05 1 S10 1
[S08, *, R45] S01 1 S02 1 S04 1 S10 1
[S08, *, R46] S02 1 S03 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S08, *, R47] S01 1 S03 1 S05 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S08, *, R48] S02 1 S03 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S08, *, R49] S01 1 S03 1 S05 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S10, *, R50] S07 1 S08 1
[S10, *, R51] S04 1 S05 1 S08 1
[S10, *, R52] S01 1 S02 1 S04 1 S08 1
[S10, *, R53] S01 1 S03 1 S04 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R54] S02 1 S03 1 S07 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R55] S01 1 S03 1 S05 1 S07 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R56] S01 1 S03 1 S04 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S11, *, R57] S12 1 S15 1
[S11, *, R58] S12 1 S13 1 S17 1 S22 1
[S11, *, R59] S12 1 S14 1 S16 1 S19 1
[S11, *, R60] S12 1 S13 1 S16 1 S18 1 S22 1
[S11, *, R61] S12 1 S14 1 S17 1 S21 1 S31 1
[S11, *, R62] S12 1 S14 1 S17 1 S21 1 S32 1 S34 1
[S11, *, R63] S12 1 S14 1 S17 1 S18 1 S19 1
[S12, *, R64] S11 1 S15 1
[S12, *, R65] S11 1 S13 1 S17 1 S22 1
[S12, *, R66] S11 1 S14 1 S16 1 S19 1
[S12, *, R67] S11 1 S13 1 S16 1 S18 1 S22 1
[S12, *, R68] S11 1 S14 1 S17 1 S21 1 S31 1
[S12, *, R69] S11 1 S14 1 S17 1 S21 1 S32 1 S34 1
[S12, *, R70] S11 1 S14 1 S17 1 S18 1 S19 1
[S13, *, R71] S15 1 S17 1 S22 1
[S13, *, R72] S15 1 S16 1 S18 1 S22 1
[S13, *, R73] S11 1 S12 1 S17 1 S22 1
[S13, *, R74] S14 1 S21 1 S22 1 S31 1
[S13, *, R75] S11 1 S12 1 S16 1 S18 1 S22 1
[S13, *, R76] S14 1 S21 1 S22 1 S32 1 S34 1
[S13, *, R77] S14 1 S18 1 S19 1 S22 1
[S14, *, R78] S15 1 S16 1 S19 1
[S14, *, R79] S15 1 S17 1 S21 1 S31 1
[S14, *, R80] S15 1 S17 1 S21 1 S32 1 S34 1
[S14, *, R81] S15 1 S17 1 S18 1 S19 1
[S14, *, R82] S11 1 S12 1 S16 1 S19 1
[S14, *, R83] S13 1 S21 1 S22 1 S31 1
[S14, *, R84] S13 1 S21 1 S22 1 S32 1 S34 1
[S15, *, R85] S11 1 S12 1
[S15, *, R86] S13 1 S17 1 S22 1
[S15, *, R87] S14 1 S16 1 S19 1
[S15, *, R88] S13 1 S16 1 S18 1 S22 1
[S15, *, R89] S14 1 S17 1 S21 1 S31 1
[S15, *, R90] S14 1 S17 1 S21 1 S32 1 S34 1
[S15, *, R91] S14 1 S17 1 S18 1 S19 1
[S16, *, R92] S17 1 S18 1
[S16, *, R93] S14 1 S15 1 S19 1
[S16, *, R94] S17 1 S19 1 S21 1 S31 1
[S16, *, R95] S13 1 S15 1 S18 1 S22 1
[S16, *, R96] S17 1 S19 1 S21 1 S32 1 S34 1
[S16, *, R97] S11 1 S12 1 S14 1 S19 1
[S16, *, R98] S11 1 S12 1 S13 1 S18 1 S22 1
[S17, *, R99] S16 1 S18 1
[S17, *, R100] S13 1 S15 1 S22 1
[S17, *, R101] S16 1 S19 1 S21 1 S31 1
[S17, *, R102] S16 1 S19 1 S21 1 S32 1 S34 1
[S17, *, R103] S11 1 S12 1 S13 1 S22 1
[S17, *, R104] S14 1 S15 1 S21 1 S31 1
[S17, *, R105] S14 1 S15 1 S21 1 S32 1 S34 1
[S18, *, R106] S16 1 S17 1
[S18, *, R107] S19 1 S21 1 S31 1
[S18, *, R108] S19 1 S21 1 S32 1 S34 1
[S18, *, R109] S13 1 S15 1 S16 1 S22 1
[S18, *, R110] S19 1 S21 1 S30 1 S32 1 S36 1 S38 1
[S18, *, R111] S14 1 S15 1 S17 1 S19 1
[S18, *, R112] S19 1 S21 1 S28 1 S32 1 S33 1
[S19, *, R113] S18 1 S21 1 S31 1
[S19, *, R114] S18 1 S21 1 S32 1 S34 1
[S19, *, R115] S14 1 S15 1 S16 1
[S19, *, R116] S16 1 S17 1 S21 1 S31 1
[S19, *, R117] S16 1 S17 1 S21 1 S32 1 S34 1
[S19, *, R118] S18 1 S21 1 S30 1 S32 1 S36 1 S38 1
[S19, *, R119] S14 1 S15 1 S17 1 S18 1
[S21, *, R120] S18 1 S19 1 S31 1
[S21, *, R121] S18 1 S19 1 S32 1 S34 1
[S21, *, R122] S16 1 S17 1 S19 1 S31 1
[S21, *, R123] S16 1 S17 1 S19 1 S32 1 S34 1
[S21, *, R124] S14 1 S15 1 S17 1 S31 1
[S21, *, R125] S18 1 S19 1 S30 1 S32 1 S36 1 S38 1
[S21, *, R126] S14 1 S15 1 S17 1 S32 1 S34 1
[S22, *, R127] S13 1 S15 1 S17 1
[S22, *, R128] S13 1 S15 1 S16 1 S18 1
[S22, *, R129] S11 1 S12 1 S13 1 S17 1
[S22, *, R130] S13 1 S14 1 S21 1 S31 1
[S22, *, R131] S11 1 S12 1 S13 1 S16 1 S18 1
[S22, *, R132] S13 1 S14 1 S21 1 S32 1 S34 1
[S22, *, R133] S13 1 S14 1 S18 1 S19 1
[S23, *, R134] S21 1 S22 1 S26 1 S29 1 S37 1
[S23, *, R135] S21 1 S22 1 S26 1 S34 1 S38 1 S39 1
[S23, *, R136] S21 1 S22 1 S26 1 S27 1 S30 1 S37 1
[S23, *, R137] S21 1 S22 1 S26 1 S30 1 S36 1 S39 1
[S23, *, R138] S21 1 S22 1 S26 1 S31 1 S32 1 S38 1 S39 1
[S23, *, R139] S21 1 S22 1 S26 1 S27 1 S34 1 S36 1 S37 1 S38 1
[S23, *, R140] S21 1 S22 1 S26 1 S27 1 S29 1 S36 1 S39 1
[S26, *, R141] S21 1 S22 1 S23 1 S29 1 S37 1
[S26, *, R142] S21 1 S22 1 S23 1 S34 1 S38 1 S39 1
[S26, *, R143] S21 1 S22 1 S23 1 S27 1 S30 1 S37 1
[S26, *, R144] S21 1 S22 1 S23 1 S30 1 S36 1 S39 1
[S26, *, R145] S21 1 S22 1 S23 1 S31 1 S32 1 S38 1 S39 1
[S26, *, R146] S21 1 S22 1 S23 1 S27 1 S34 1 S36 1 S37 1 S38 1
[S26, *, R147] S21 1 S22 1 S23 1 S27 1 S29 1 S36 1 S39 1
[S27, *, R148] S29 1 S30 1
[S27, *, R149] S36 1 S37 1 S39 1
[S27, *, R150] S29 1 S34 1 S36 1 S38 1
[S27, *, R151] S29 1 S31 1 S32 1 S36 1 S38 1
[S27, *, R152] S29 1 S34 1 S35 1 S36 1 S40 1
[S27, *, R153] S30 1 S34 1 S37 1 S38 1 S39 1
[S27, *, R154] S29 1 S31 1 S32 1 S35 1 S36 1 S40 1
[S28, *, R155] S33 1 S34 1
[S28, *, R156] S31 1 S32 1 S33 1
[S28, *, R157] S30 1 S33 1 S36 1 S38 1
[S28, *, R158] S27 1 S29 1 S33 1 S36 1 S38 1
[S28, *, R159] S30 1 S33 1 S35 1 S36 1 S40 1
[S28, *, R160] S18 1 S19 1 S21 1 S32 1 S33 1
[S28, *, R161] S16 1 S17 1 S19 1 S21 1 S32 1 S33 1
[S29, *, R162] S27 1 S30 1
[S29, *, R163] S27 1 S34 1 S36 1 S38 1
[S29, *, R164] S27 1 S31 1 S32 1 S36 1 S38 1
[S29, *, R165] S27 1 S34 1 S35 1 S36 1 S40 1
[S29, *, R166] S34 1 S37 1 S38 1 S39 1
[S29, *, R167] S30 1 S36 1 S37 1 S39 1
[S29, *, R168] S27 1 S31 1 S32 1 S35 1 S36 1 S40 1
[S30, *, R169] S34 1 S36 1 S38 1

[S30, *, R170] S27 1 S29 1
[S30, *, R171] S31 1 S32 1 S36 1 S38 1
[S30, *, R172] S34 1 S35 1 S36 1 S40 1
[S30, *, R173] S31 1 S32 1 S35 1 S36 1 S40 1
[S30, *, R174] S28 1 S33 1 S36 1 S38 1
[S30, *, R175] S29 1 S36 1 S37 1 S39 1
[S31, *, R176] S32 1 S34 1
[S31, *, R177] S30 1 S32 1 S36 1 S38 1
[S31, *, R178] S18 1 S19 1 S21 1
[S31, *, R179] S28 1 S32 1 S33 1
[S31, *, R180] S16 1 S17 1 S19 1 S21 1
[S31, *, R181] S27 1 S29 1 S32 1 S36 1 S38 1
[S31, *, R182] S30 1 S32 1 S35 1 S36 1 S40 1
[S32, *, R183] S31 1 S34 1
[S32, *, R184] S30 1 S31 1 S36 1 S38 1
[S32, *, R185] S28 1 S31 1 S33 1
[S32, *, R186] S18 1 S19 1 S21 1 S34 1
[S32, *, R187] S27 1 S29 1 S31 1 S36 1 S38 1
[S32, *, R188] S16 1 S17 1 S19 1 S21 1 S34 1
[S32, *, R189] S30 1 S31 1 S35 1 S36 1 S40 1
[S33, *, R190] S28 1 S34 1
[S33, *, R191] S28 1 S31 1 S32 1
[S33, *, R192] S28 1 S30 1 S36 1 S38 1
[S33, *, R193] S27 1 S28 1 S29 1 S36 1 S38 1
[S33, *, R194] S28 1 S30 1 S35 1 S36 1 S40 1
[S33, *, R195] S18 1 S19 1 S21 1 S28 1 S32 1
[S33, *, R196] S16 1 S17 1 S19 1 S21 1 S28 1 S32 1
[S34, *, R197] S31 1 S32 1
[S34, *, R198] S30 1 S36 1 S38 1
[S34, *, R199] S28 1 S33 1
[S34, *, R200] S27 1 S29 1 S36 1 S38 1
[S34, *, R201] S30 1 S35 1 S36 1 S40 1
[S34, *, R202] S18 1 S19 1 S21 1 S32 1
[S34, *, R203] S16 1 S17 1 S19 1 S21 1 S32 1
[S35, *, R204] S38 1 S40 1
[S35, *, R205] S30 1 S34 1 S36 1 S40 1
[S35, *, R206] S30 1 S31 1 S32 1 S36 1 S40 1
[S35, *, R207] S27 1 S29 1 S34 1 S36 1 S40 1
[S35, *, R208] S27 1 S29 1 S31 1 S32 1 S36 1 S40 1
[S35, *, R209] S28 1 S30 1 S33 1 S36 1 S40 1
[S35, *, R210] S29 1 S34 1 S37 1 S39 1 S40 1
[S36, *, R211] S30 1 S34 1 S38 1
[S36, *, R212] S27 1 S37 1 S39 1
[S36, *, R213] S30 1 S31 1 S32 1 S38 1
[S36, *, R214] S27 1 S29 1 S34 1 S38 1
[S36, *, R215] S30 1 S34 1 S35 1 S40 1
[S36, *, R216] S27 1 S29 1 S31 1 S32 1 S38 1
[S36, *, R217] S30 1 S31 1 S32 1 S35 1 S40 1
[S37, *, R218] S27 1 S36 1 S39 1
[S37, *, R219] S29 1 S34 1 S38 1 S39 1
[S37, *, R220] S29 1 S30 1 S36 1 S39 1
[S37, *, R221] S27 1 S30 1 S34 1 S38 1 S39 1
[S37, *, R222] S29 1 S31 1 S32 1 S38 1 S39 1
[S37, *, R223] S29 1 S34 1 S35 1 S39 1 S40 1
[S37, *, R224] S27 1 S30 1 S31 1 S32 1 S38 1 S39 1
[S38, *, R225] S35 1 S40 1
[S38, *, R226] S30 1 S34 1 S36 1
[S38, *, R227] S30 1 S31 1 S32 1 S36 1
[S38, *, R228] S27 1 S29 1 S34 1 S36 1
[S38, *, R229] S27 1 S29 1 S31 1 S32 1 S36 1
[S38, *, R230] S28 1 S30 1 S33 1 S36 1
[S38, *, R231] S29 1 S34 1 S37 1 S39 1
[S39, *, R232] S27 1 S36 1 S37 1
[S39, *, R233] S29 1 S34 1 S37 1 S38 1
[S39, *, R234] S29 1 S30 1 S36 1 S37 1
[S39, *, R235] S27 1 S30 1 S34 1 S37 1 S38 1
[S39, *, R236] S29 1 S31 1 S32 1 S37 1 S38 1
[S39, *, R237] S29 1 S34 1 S35 1 S37 1 S40 1
[S39, *, R238] S27 1 S30 1 S31 1 S32 1 S37 1 S38 1
[S40, *, R239] S35 1 S38 1
[S40, *, R240] S30 1 S34 1 S35 1 S36 1
[S40, *, R241] S30 1 S31 1 S32 1 S35 1 S36 1
[S40, *, R242] S27 1 S29 1 S34 1 S35 1 S36 1
[S40, *, R243] S27 1 S29 1 S31 1 S32 1 S35 1 S36 1
[S40, *, R244] S28 1 S30 1 S33 1 S35 1 S36 1
[S40, *, R245] S29 1 S34 1 S35 1 S37 1 S39 1;

Appendix F The MDNF-ml Model

Appendix F.1 – AMPL codes of MDNF-ml Model for Span Restoration Mechanism

```
# Span-Restoration mechanism under dual failure scenario
# March 2015 by Wenjing Wang

# ************************
# SETS
# ************************

set SPANS;
# set of all spans

set BACKUP_ROUTES{i in SPANS};
# set of all backup routes for each span failure i

# ************************
# PARAMETERS
# ************************

param Work{i in SPANS};
# amount of working capacity placed on span i

param Cost{k in SPANS};
# cost of each unit of capacity on span k

param Delta{i in SPANS, k in SPANS, b in BACKUP_ROUTES[i]} default 0;
# binary, takes 1 if backup route b for failure of span i crosses span k

param Cinfinit;
# a positive large constant

param Budget;
# budget limit for dual failure restoration

param NWCmin;
# limitation on NWC

# ************************
# VIRAIABLES
# ************************

var spare{k in SPANS} >=0 integer, <=10000;
# amount of spare capacity placed on span j

var flow_single{i in SPANS, b in BACKUP_ROUTES[i]} >=0 integer, <=10000;
# flow through backup route b for failure of span i

var flow_dual_i{i in SPANS, j in SPANS, b in BACKUP_ROUTES[i]: i <> j} >=0 integer, <=10000;
# flow through backup route b for failure of span i under dual failure (i, j)

var flow_dual_j{i in SPANS, j in SPANS, b in BACKUP_ROUTES[j]: i <> j} >=0 integer, <=10000;
# flow through backup route b for failure of span j under dual failure (i, j)

var non_restored{i in SPANS, j in SPANS: i <> j} >=0 integer;
# amount of non-restored working capacity under dual failure (i, j)

var non_restored_i{i in SPANS, j in SPANS: i <> j} >=0 integer;
# amount of non-restored working capacity on span i under dual failure (i, j)

var non_restored_j{i in SPANS, j in SPANS: i <> j} >=0 integer;
# amount of non-restored working capacity on span j under dual failure (i, j)

# ************************
# OBJECTIVE FUNCTION
# ************************

minimize tot_non_restored:
sum{i in SPANS, j in SPANS: i <> j} non_restored[i, j];
# minimize amount of non-restored working capacity under all dual failure scenarios

# ************************
# CONSTRAINTS
# ************************


subject to Tot_NonRestored:
  sum{i in SPANS, j in SPANS: i <> j} non_restored[i, j] >= NWCmin;
```

```
# Comfine the value of OBJ to get N(i, j) under different total non-restored working capacities.

subject to c_12{i in SPANS, j in SPANS: i <> j}:
  non_restored[i, j] = Work[i] + Work[j] - sum{b in BACKUP_ROUTES[i]} flow_dual_i[i, j, b] - sum{b in BACKUP_ROUTES[j]} flow_dual_j[i, j, b];
# define non-restored working capacities under each dual failure (i, j)

subject to c_1201{i in SPANS, j in SPANS: i <> j}:
  non_restored_i[i, j] = Work[i] - sum{b in BACKUP_ROUTES[i]} flow_dual_i[i, j, b];
# define non-restored working capacities on span i under each dual failure (i, j)

subject to c_1202{i in SPANS, j in SPANS: i <> j}:
  non_restored_j[i, j] = Work[j] - sum{b in BACKUP_ROUTES[j]} flow_dual_j[i, j, b];
# define non-restored working capacities on span j under each dual failure (i, j)

subject to c_13{i in SPANS}: sum{b in BACKUP_ROUTES[i]} flow_single[i, b] >= Work[i];
# guarantee enough restoration flow for full single failure restorability

subject to c_14{i in SPANS, k in SPANS: k <> i}:
  spare[k] >= sum{b in BACKUP_ROUTES[i]} (flow_single[i, b] * Delta[i, k, b]);
# translate flow requirements in c_13 to each span

subject to c_15{i in SPANS, j in SPANS: i <> j}:
  sum{b in BACKUP_ROUTES[i]} flow_dual_i[i, j, b] <= Work[i];
# assign restoration flow to failed span i under dual failure scenario (i, j)

subject to c_16{i in SPANS, j in SPANS: i <> j}:
  sum{b in BACKUP_ROUTES[j]} flow_dual_j[i, j, b] <= Work[j];
# assign restoration flow to failed span j under dual failure scenario (i, j)

subject to c_17{i in SPANS, j in SPANS, k in SPANS: i <> j and j <> k and i <> k}:
  spare[k] >= sum{b in BACKUP_ROUTES[i]} (flow_dual_i[i, j, b] * Delta[i, k, b]) +
          sum{b in BACKUP_ROUTES[j]} (flow_dual_j[i, j, b] * Delta[j, k, b]);
# translate flow requirements in (15)(16) on each span

subject to c_18{i in SPANS, j in SPANS, b in BACKUP_ROUTES[i]: i <> j}:
  flow_dual_i[i, j, b] <= Cinfinit * (1 - Delta[i, j, b]);
# put limitation on restoration of failed span i under dual failure (i, j)

subject to c_19{i in SPANS, j in SPANS, b in BACKUP_ROUTES[j]: i <> j}:
  flow_dual_j[i, j, b] <= Cinfinit * (1 - Delta[j, i, b]);
# put limitation on restoration of failed span j under dual failure (i, j)

subject to c_20:
  sum{k in SPANS} (Cost[k] * spare[k]) <= Budget;
# put budget limitation on dual failure restoration
```

Appendix F.2 – An Example of *.Dat File of MDNF-ml Model for Span Restoration Mechanism

```
# *.dat file for 20-node 35-span network
# MNDF-ml model
# Created in January 2015 by Wenjing

set SPANS := S01 S02 S03 S04 S05 S07 S08 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S21 S22 S23 S26 S27 S28 S29 S30 S31 S32 S33 S34 S35 S36
S37 S38 S39 S40;

param Cost :=
S01     139.560
S02     179.360
S03     116.181
S04     167.601
S05     50.606
S07     134.302
S08     227.002
S10     136.356
S11     152.506
S12     127.475
S13     148.772
S14     260.923
S15     92.779
S16     86.822
S17     124.631
S18     145.055
S19     131.320
S21     99.705
S22     131.187
S23     173.118
S26     198.497
S27     82.970
S28     128.725
S29     166.066
S30     151.427
S31     130.173
S32     64.070
S33     204.924
S34     108.074
S35     104.805
S36     54.129
S37     121.037
S38     86.833
S39     156.984
S40     120.150;

param Work :=
S01     87
S02     69
S03     200
S04     26
S05     31
S07     2
S08     65
S10     63
S11     60
S12     154
S13     68
S14     20
S15     50
S16     85
S17     133
S18     36
S19     99
S21     236
S22     86
S23     73
S26     69
S27     52
S28     66
S29     71
S30     43
S31     22
S32     137
S33     37
S34     76
S35     126
S36     85
S37     47
S38     120
S39     85
S40     114;

set BACKUP_ROUTES[S01] := R1 R2 R3 R4 R5 R6 R7;
set BACKUP_ROUTES[S02] := R8 R9 R10 R11 R12 R13 R14;
set BACKUP_ROUTES[S03] := R15 R16 R17 R18 R19 R20 R21;
set BACKUP_ROUTES[S04] := R22 R23 R24 R25 R26 R27 R28;
set BACKUP_ROUTES[S05] := R29 R30 R31 R32 R33 R34 R35;
set BACKUP_ROUTES[S07] := R36 R37 R38 R39 R40 R41 R42;
set BACKUP_ROUTES[S08] := R43 R44 R45 R46 R47 R48 R49;
set BACKUP_ROUTES[S10] := R50 R51 R52 R53 R54 R55 R56;
set BACKUP_ROUTES[S11] := R57 R58 R59 R60 R61 R62 R63;
set BACKUP_ROUTES[S12] := R64 R65 R66 R67 R68 R69 R70;
set BACKUP_ROUTES[S13] := R71 R72 R73 R74 R75 R76 R77;
set BACKUP_ROUTES[S14] := R78 R79 R80 R81 R82 R83 R84;
set BACKUP_ROUTES[S15] := R85 R86 R87 R88 R89 R90 R91;
set BACKUP_ROUTES[S16] := R92 R93 R94 R95 R96 R97 R98;
set BACKUP_ROUTES[S17] := R99 R100 R101 R102 R103 R104 R105;
set BACKUP_ROUTES[S18] := R106 R107 R108 R109 R110 R111 R112;
set BACKUP_ROUTES[S19] := R113 R114 R115 R116 R117 R118 R119;
set BACKUP_ROUTES[S21] := R120 R121 R122 R123 R124 R125 R126;
set BACKUP_ROUTES[S22] := R127 R128 R129 R130 R131 R132 R133;
set BACKUP_ROUTES[S23] := R134 R135 R136 R137 R138 R139 R140;
set BACKUP_ROUTES[S26] := R141 R142 R143 R144 R145 R146 R147;
set BACKUP_ROUTES[S27] := R148 R149 R150 R151 R152 R153 R154;
set BACKUP_ROUTES[S28] := R155 R156 R157 R158 R159 R160 R161;
set BACKUP_ROUTES[S29] := R162 R163 R164 R165 R166 R167 R168;
set BACKUP_ROUTES[S30] := R169 R170 R171 R172 R173 R174 R175;
set BACKUP_ROUTES[S31] := R176 R177 R178 R179 R180 R181 R182;
set BACKUP_ROUTES[S32] := R183 R184 R185 R186 R187 R188 R189;
set BACKUP_ROUTES[S33] := R190 R191 R192 R193 R194 R195 R196;
set BACKUP_ROUTES[S34] := R197 R198 R199 R200 R201 R202 R203;
set BACKUP_ROUTES[S35] := R204 R205 R206 R207 R208 R209 R210;
set BACKUP_ROUTES[S36] := R211 R212 R213 R214 R215 R216 R217;
set BACKUP_ROUTES[S37] := R218 R219 R220 R221 R222 R223 R224;
set BACKUP_ROUTES[S38] := R225 R226 R227 R228 R229 R230 R231;
set BACKUP_ROUTES[S39] := R232 R233 R234 R235 R236 R237 R238;
set BACKUP_ROUTES[S40] := R239 R240 R241 R242 R243 R244 R245;

param Delta :=
[S01, *, R1] S02 1 S05 1
[S01, *, R2] S02 1 S04 1 S07 1
[S01, *, R3] S02 1 S04 1 S08 1 S10 1
[S01, *, R4] S03 1 S05 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R5] S03 1 S04 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R6] S03 1 S05 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S01, *, R7] S03 1 S05 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S02, *, R8] S01 1 S05 1
[S02, *, R9] S01 1 S04 1 S07 1
[S02, *, R10] S01 1 S04 1 S08 1 S10 1
[S02, *, R11] S03 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S02, *, R12] S03 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S02, *, R13] S03 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
```

[S02, *, R14] S03 1 S08 1 S11 1 S14 1 S32 1 S35 1
[S03, *, R15] S02 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R16] S01 1 S05 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R17] S01 1 S04 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R18] S02 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R19] S02 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S03, *, R20] S01 1 S05 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S03, *, R21] S01 1 S05 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S04, *, R22] S05 1 S07 1
[S04, *, R23] S05 1 S08 1 S10 1
[S04, *, R24] S01 1 S02 1 S07 1
[S04, *, R25] S01 1 S02 1 S08 1 S10 1
[S04, *, R26] S01 1 S03 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S04, *, R27] S01 1 S03 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S04, *, R28] S01 1 S03 1 S10 1 S11 1 S14 1 S32 1 S35 1
[S05, *, R29] S04 1 S07 1
[S05, *, R30] S01 1 S02 1
[S05, *, R31] S04 1 S08 1 S10 1
[S05, *, R32] S01 1 S03 1 S08 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S05, *, R33] S01 1 S03 1 S07 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S05, *, R34] S01 1 S03 1 S08 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S05, *, R35] S01 1 S03 1 S08 1 S11 1 S14 1 S32 1 S35 1
[S07, *, R36] S04 1 S05 1
[S07, *, R37] S08 1 S10 1
[S07, *, R38] S01 1 S02 1 S04 1
[S07, *, R39] S02 1 S03 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S07, *, R40] S01 1 S03 1 S05 1 S10 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S07, *, R41] S02 1 S03 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S07, *, R42] S01 1 S03 1 S05 1 S10 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S08, *, R43] S07 1 S10 1
[S08, *, R44] S04 1 S05 1 S10 1
[S08, *, R45] S01 1 S02 1 S04 1 S10 1
[S08, *, R46] S02 1 S03 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S08, *, R47] S01 1 S03 1 S05 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S08, *, R48] S02 1 S03 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S08, *, R49] S01 1 S03 1 S05 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S10, *, R50] S07 1 S08 1
[S10, *, R51] S04 1 S05 1 S08 1
[S10, *, R52] S01 1 S02 1 S04 1 S08 1
[S10, *, R53] S01 1 S03 1 S04 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R54] S02 1 S03 1 S07 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R55] S01 1 S03 1 S05 1 S07 1 S12 1 S16 1 S19 1 S32 1 S35 1
[S10, *, R56] S01 1 S03 1 S04 1 S12 1 S17 1 S21 1 S34 1 S35 1
[S11, *, R57] S12 1 S15 1
[S11, *, R58] S12 1 S13 1 S17 1 S22 1
[S11, *, R59] S12 1 S14 1 S16 1 S19 1
[S11, *, R60] S12 1 S13 1 S16 1 S18 1 S22 1
[S11, *, R61] S12 1 S14 1 S17 1 S21 1 S31 1
[S11, *, R62] S12 1 S14 1 S17 1 S21 1 S32 1 S34 1
[S11, *, R63] S12 1 S14 1 S17 1 S18 1 S19 1
[S12, *, R64] S11 1 S15 1
[S12, *, R65] S11 1 S13 1 S17 1 S22 1
[S12, *, R66] S11 1 S14 1 S16 1 S19 1
[S12, *, R67] S11 1 S13 1 S16 1 S18 1 S22 1
[S12, *, R68] S11 1 S14 1 S17 1 S21 1 S31 1
[S12, *, R69] S11 1 S14 1 S17 1 S21 1 S32 1 S34 1
[S12, *, R70] S11 1 S14 1 S17 1 S18 1 S19 1
[S13, *, R71] S15 1 S17 1 S22 1
[S13, *, R72] S15 1 S16 1 S18 1 S22 1
[S13, *, R73] S11 1 S12 1 S17 1 S22 1
[S13, *, R74] S14 1 S21 1 S22 1 S31 1
[S13, *, R75] S11 1 S12 1 S16 1 S18 1 S22 1
[S13, *, R76] S14 1 S21 1 S22 1 S32 1 S34 1
[S13, *, R77] S14 1 S18 1 S19 1 S22 1
[S14, *, R78] S15 1 S16 1 S19 1
[S14, *, R79] S15 1 S17 1 S21 1 S31 1
[S14, *, R80] S15 1 S17 1 S21 1 S32 1 S34 1
[S14, *, R81] S15 1 S17 1 S18 1 S19 1
[S14, *, R82] S11 1 S12 1 S16 1 S19 1
[S14, *, R83] S13 1 S21 1 S22 1 S31 1
[S14, *, R84] S13 1 S21 1 S22 1 S32 1 S34 1
[S15, *, R85] S11 1 S12 1
[S15, *, R86] S13 1 S17 1 S22 1
[S15, *, R87] S14 1 S16 1 S19 1
[S15, *, R88] S13 1 S16 1 S18 1 S22 1
[S15, *, R89] S14 1 S17 1 S21 1 S31 1
[S15, *, R90] S14 1 S17 1 S21 1 S32 1 S34 1
[S15, *, R91] S14 1 S17 1 S18 1 S19 1
[S16, *, R92] S17 1 S18 1
[S16, *, R93] S14 1 S15 1 S19 1
[S16, *, R94] S17 1 S19 1 S21 1 S31 1
[S16, *, R95] S13 1 S15 1 S18 1 S22 1
[S16, *, R96] S17 1 S19 1 S21 1 S32 1 S34 1
[S16, *, R97] S11 1 S12 1 S14 1 S19 1
[S16, *, R98] S11 1 S12 1 S13 1 S18 1 S22 1
[S17, *, R99] S16 1 S18 1
[S17, *, R100] S13 1 S15 1 S22 1
[S17, *, R101] S16 1 S19 1 S21 1 S31 1
[S17, *, R102] S16 1 S19 1 S21 1 S32 1 S34 1
[S17, *, R103] S11 1 S12 1 S13 1 S22 1
[S17, *, R104] S14 1 S15 1 S21 1 S31 1
[S17, *, R105] S14 1 S15 1 S21 1 S32 1 S34 1
[S18, *, R106] S16 1 S17 1
[S18, *, R107] S19 1 S21 1 S31 1
[S18, *, R108] S19 1 S21 1 S32 1 S34 1
[S18, *, R109] S13 1 S15 1 S16 1 S22 1
[S18, *, R110] S19 1 S21 1 S30 1 S32 1 S36 1 S38 1
[S18, *, R111] S14 1 S15 1 S17 1 S19 1
[S18, *, R112] S19 1 S21 1 S28 1 S32 1 S33 1
[S19, *, R113] S18 1 S21 1 S31 1
[S19, *, R114] S18 1 S21 1 S32 1 S34 1
[S19, *, R115] S14 1 S15 1 S16 1
[S19, *, R116] S16 1 S17 1 S21 1 S31 1
[S19, *, R117] S16 1 S17 1 S21 1 S32 1 S34 1
[S19, *, R118] S18 1 S21 1 S30 1 S32 1 S36 1 S38 1
[S19, *, R119] S14 1 S15 1 S17 1 S18 1
[S21, *, R120] S18 1 S19 1 S31 1
[S21, *, R121] S18 1 S19 1 S32 1 S34 1
[S21, *, R122] S16 1 S17 1 S19 1 S31 1
[S21, *, R123] S16 1 S17 1 S19 1 S32 1 S34 1
[S21, *, R124] S14 1 S15 1 S17 1 S31 1
[S21, *, R125] S18 1 S19 1 S30 1 S32 1 S36 1 S38 1
[S21, *, R126] S14 1 S15 1 S17 1 S32 1 S34 1
[S22, *, R127] S13 1 S15 1 S17 1
[S22, *, R128] S13 1 S15 1 S16 1 S18 1
[S22, *, R129] S11 1 S12 1 S13 1 S17 1
[S22, *, R130] S13 1 S14 1 S21 1 S31 1
[S22, *, R131] S11 1 S12 1 S13 1 S16 1 S18 1
[S22, *, R132] S13 1 S14 1 S21 1 S32 1 S34 1
[S22, *, R133] S13 1 S14 1 S18 1 S19 1
[S23, *, R134] S21 1 S22 1 S26 1 S29 1 S37 1
[S23, *, R135] S21 1 S22 1 S26 1 S34 1 S38 1 S39 1
[S23, *, R136] S21 1 S22 1 S26 1 S27 1 S30 1 S37 1
[S23, *, R137] S21 1 S22 1 S26 1 S30 1 S36 1 S39 1
[S23, *, R138] S21 1 S22 1 S26 1 S31 1 S32 1 S38 1 S39 1
[S23, *, R139] S21 1 S22 1 S26 1 S27 1 S34 1 S36 1 S37 1 S38 1
[S23, *, R140] S21 1 S22 1 S26 1 S27 1 S29 1 S36 1 S39 1
[S26, *, R141] S21 1 S22 1 S23 1 S29 1 S37 1
[S26, *, R142] S21 1 S22 1 S23 1 S34 1 S38 1 S39 1
[S26, *, R143] S21 1 S22 1 S23 1 S27 1 S30 1 S37 1
[S26, *, R144] S21 1 S22 1 S23 1 S30 1 S36 1 S39 1
[S26, *, R145] S21 1 S22 1 S23 1 S31 1 S32 1 S38 1 S39 1
[S26, *, R146] S21 1 S22 1 S23 1 S27 1 S34 1 S36 1 S37 1 S38 1
[S26, *, R147] S21 1 S22 1 S23 1 S27 1 S29 1 S36 1 S39 1
[S27, *, R148] S29 1 S30 1
[S27, *, R149] S36 1 S37 1 S39 1
[S27, *, R150] S29 1 S34 1 S36 1 S38 1
[S27, *, R151] S29 1 S31 1 S32 1 S36 1 S38 1
[S27, *, R152] S29 1 S34 1 S35 1 S36 1 S40 1
[S27, *, R153] S30 1 S34 1 S37 1 S38 1 S39 1
[S27, *, R154] S29 1 S31 1 S32 1 S35 1 S36 1 S40 1
[S28, *, R155] S33 1 S34 1
[S28, *, R156] S31 1 S32 1 S33 1
[S28, *, R157] S30 1 S33 1 S36 1 S38 1
[S28, *, R158] S27 1 S29 1 S33 1 S36 1 S38 1
[S28, *, R159] S30 1 S33 1 S35 1 S36 1 S40 1
[S28, *, R160] S18 1 S19 1 S21 1 S32 1 S33 1
[S28, *, R161] S16 1 S17 1 S19 1 S21 1 S32 1 S33 1
[S29, *, R162] S27 1 S30 1
[S29, *, R163] S27 1 S34 1 S36 1 S38 1
[S29, *, R164] S27 1 S31 1 S32 1 S36 1 S38 1
[S29, *, R165] S27 1 S34 1 S35 1 S36 1 S40 1
[S29, *, R166] S34 1 S37 1 S38 1 S39 1
[S29, *, R167] S30 1 S36 1 S37 1 S39 1
[S29, *, R168] S27 1 S31 1 S32 1 S35 1 S36 1 S40 1
[S30, *, R169] S34 1 S36 1 S38 1
[S30, *, R170] S27 1 S29 1
[S30, *, R171] S31 1 S32 1 S36 1 S38 1
[S30, *, R172] S34 1 S35 1 S36 1 S40 1
[S30, *, R173] S31 1 S32 1 S35 1 S36 1 S40 1

[S30, *, R174] S28 1 S33 1 S36 1 S38 1
[S30, *, R175] S29 1 S36 1 S37 1 S39 1
[S31, *, R176] S32 1 S34 1
[S31, *, R177] S30 1 S32 1 S36 1 S38 1
[S31, *, R178] S18 1 S19 1 S21 1
[S31, *, R179] S28 1 S32 1 S33 1
[S31, *, R180] S16 1 S17 1 S19 1 S21 1
[S31, *, R181] S27 1 S29 1 S32 1 S36 1 S38 1
[S31, *, R182] S30 1 S32 1 S35 1 S36 1 S40 1
[S32, *, R183] S31 1 S34 1
[S32, *, R184] S30 1 S31 1 S36 1 S38 1
[S32, *, R185] S28 1 S31 1 S33 1
[S32, *, R186] S18 1 S19 1 S21 1 S34 1
[S32, *, R187] S27 1 S29 1 S31 1 S36 1 S38 1
[S32, *, R188] S16 1 S17 1 S19 1 S21 1 S34 1
[S32, *, R189] S30 1 S31 1 S35 1 S36 1 S40 1
[S33, *, R190] S28 1 S34 1
[S33, *, R191] S28 1 S31 1 S32 1
[S33, *, R192] S28 1 S30 1 S36 1 S38 1
[S33, *, R193] S27 1 S28 1 S29 1 S36 1 S38 1
[S33, *, R194] S28 1 S30 1 S35 1 S36 1 S40 1
[S33, *, R195] S18 1 S19 1 S21 1 S28 1 S32 1
[S33, *, R196] S16 1 S17 1 S19 1 S21 1 S28 1 S32 1
[S34, *, R197] S31 1 S32 1
[S34, *, R198] S30 1 S36 1 S38 1
[S34, *, R199] S28 1 S33 1
[S34, *, R200] S27 1 S29 1 S36 1 S38 1
[S34, *, R201] S30 1 S35 1 S36 1 S40 1
[S34, *, R202] S18 1 S19 1 S21 1 S32 1
[S34, *, R203] S16 1 S17 1 S19 1 S21 1 S32 1
[S35, *, R204] S38 1 S40 1
[S35, *, R205] S30 1 S34 1 S36 1 S40 1
[S35, *, R206] S30 1 S31 1 S32 1 S36 1 S40 1
[S35, *, R207] S27 1 S29 1 S34 1 S36 1 S40 1
[S35, *, R208] S27 1 S29 1 S31 1 S32 1 S36 1 S40 1
[S35, *, R209] S28 1 S30 1 S33 1 S36 1 S40 1
[S35, *, R210] S29 1 S34 1 S37 1 S39 1 S40 1
[S36, *, R211] S30 1 S34 1 S38 1
[S36, *, R212] S27 1 S37 1 S39 1
[S36, *, R213] S30 1 S31 1 S32 1 S38 1
[S36, *, R214] S27 1 S29 1 S34 1 S38 1
[S36, *, R215] S30 1 S34 1 S35 1 S40 1
[S36, *, R216] S27 1 S29 1 S31 1 S32 1 S38 1
[S36, *, R217] S30 1 S31 1 S32 1 S35 1 S40 1
[S37, *, R218] S27 1 S36 1 S39 1
[S37, *, R219] S29 1 S34 1 S38 1 S39 1
[S37, *, R220] S29 1 S30 1 S36 1 S39 1
[S37, *, R221] S27 1 S30 1 S34 1 S38 1 S39 1
[S37, *, R222] S29 1 S31 1 S32 1 S38 1 S39 1
[S37, *, R223] S29 1 S34 1 S35 1 S39 1 S40 1
[S37, *, R224] S27 1 S30 1 S31 1 S32 1 S38 1 S39 1
[S38, *, R225] S35 1 S40 1
[S38, *, R226] S30 1 S34 1 S36 1
[S38, *, R227] S30 1 S31 1 S32 1 S36 1
[S38, *, R228] S27 1 S29 1 S34 1 S36 1
[S38, *, R229] S27 1 S29 1 S31 1 S32 1 S36 1
[S38, *, R230] S28 1 S30 1 S33 1 S36 1
[S38, *, R231] S29 1 S34 1 S37 1 S39 1
[S39, *, R232] S27 1 S36 1 S37 1
[S39, *, R233] S29 1 S34 1 S37 1 S38 1
[S39, *, R234] S29 1 S30 1 S36 1 S37 1
[S39, *, R235] S27 1 S30 1 S34 1 S37 1 S38 1
[S39, *, R236] S29 1 S31 1 S32 1 S37 1 S38 1
[S39, *, R237] S29 1 S34 1 S35 1 S37 1 S40 1
[S39, *, R238] S27 1 S30 1 S31 1 S32 1 S37 1 S38 1
[S40, *, R239] S35 1 S38 1
[S40, *, R240] S30 1 S34 1 S35 1 S36 1
[S40, *, R241] S30 1 S31 1 S32 1 S35 1 S36 1
[S40, *, R242] S27 1 S29 1 S34 1 S35 1 S36 1
[S40, *, R243] S27 1 S29 1 S31 1 S32 1 S35 1 S36 1
[S40, *, R244] S28 1 S30 1 S33 1 S35 1 S36 1
[S40, *, R245] S29 1 S34 1 S35 1 S37 1 S39 1;

param Cinfinit := 10000000000000;

param Budget := 396588;

param NWCmin := 27216;

Appendix G Path Restoration

Appendix G.1 – AMPL codes of Path Restoration Mechanism

```
# Path-restoration mechanism under full single failure scenario
# June 2016 by Wenjing Wang

# *************************
# TOPOLOGY DEFINITION
# *************************

set SPANS;

set DEMANDS;

param Cost{k in SPANS};

# *************************
# DESCRIPTION OF WORKING DEMANDS AND THEIR NORMAL ROUTING
# *************************

param DemUnits{r in DEMANDS} default 0;

set WORK_ROUTES{r in DEMANDS};

set WORK_ROUTE_VECTORS{r in DEMANDS, p in WORK_ROUTES[r]} within {j in SPANS};

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.


# *************************
# FAILURE SCENARIO DEFINITIONS
# *************************

set DEMANDS_AFFECTED{i in SPANS} := {r in DEMANDS : exists {p in WORK_ROUTES[r], k in WORK_ROUTE_VECTORS[r,p]} k = i };

set WORK_ROUTES_AFFECTED{i in SPANS, r in DEMANDS_AFFECTED[i]} := {p in WORK_ROUTES[r] : exists {k in WORK_ROUTE_VECTORS[r,p]} k = i };

# *************************
# ELIGIBLE ROUTES FOR PATH-LEVEL RESTORATION OF O-D PAIRS
# *************************

set REST_ROUTES{r in DEMANDS};

set REST_ROUTE_VECTORS{r in DEMANDS, b in REST_ROUTES[r]} within {j in SPANS};

set SPECIFIC_REST_ROUTES{r in DEMANDS, i in SPANS} := {b in REST_ROUTES[r]: forall {j in REST_ROUTE_VECTORS[r,b]} j <> i};

# *************************
# VARIABLES
# *************************

var wf{r in DEMANDS, p in WORK_ROUTES[r]} integer >=0, <=MaxFlow;

var bf {i in SPANS, r in DEMANDS_AFFECTED[i], p in WORK_ROUTES_AFFECTED[i, r], b in REST_ROUTES[r]} integer >=0, <=MaxFlow;

var spare{j in SPANS} >=0, <=MaxFlow integer;

var work{j in SPANS} >=0, <=MaxFlow integer;

var Stub_release{i in SPANS, k in SPANS: i <> k} >=0, <=MaxFlow;

# *************************
# OBJECTIVE FUNCTION
# *************************

minimize TotalCost: sum{k in SPANS} (work[k] * Cost[k] + spare[k] * Cost[k]);

# *************************
# CONSTRAINTS
# *************************

subject to c_01{r in DEMANDS}:
sum{p in WORK_ROUTES[r]} wf[r,p] = DemUnits[r];

subject to c_02{k in SPANS}:
work[k] = sum{r in DEMANDS, p in WORK_ROUTES[r]: exists {j in WORK_ROUTE_VECTORS[r, p]} j = k} wf[r, p];
```

```
subject to c_03 {i in SPANS, r in DEMANDS_AFFECTED[i], p in WORK_ROUTES_AFFECTED[i, r]}:
sum{b in SPECIFIC_REST_ROUTES[r, i]} bf[i,r,p,b] = wf[r,p];

subject to c_04 {i in SPANS, k in SPANS: i <> k}:
spare[k] >= sum {r in DEMANDS_AFFECTED[i], p in WORK_ROUTES_AFFECTED[i, r], b in SPECIFIC_REST_ROUTES[r, i]: exists {j in
REST_ROUTE_VECTORS[r,b]} j = k} bf[i,r,p,b] - Stub_release[i,k];

subject to c_05 {i in SPANS, k in SPANS: i <> k}:
Stub_release[i,k] = sum {r in DEMANDS_AFFECTED[i], p in WORK_ROUTES_AFFECTED[i,r]: exists {j in WORK_ROUTE_VECTORS[r,p]} j = k }
wf[r,p];
```

Appendix G.2 – An Example of *.Data Files for Path Restoration Model

Note: only part of the dapa is shown here but full seat of data is available upon request.

```
# This data prep file is for Path-Restoration model file
# Created by Wenjing Wang in June 2016
# 10-node 15-span netowrk, routeLimit=5

set SPANS :=  S01  S02  S03  S04  S05  S06  S07  S08  S09  S10
          S11  S12  S13  S14  S15;

param Cost :=
  S01   208.5857
  S02   242.6706
  S03   245.1775
  S04   289.1730
  S05   617.1102
  S06   286.9042
  S07   283.4255
  S08   211.5490
  S09   269.2675
  S10   398.3780
  S11   306.8061
  S12   199.0226
  S13   304.8442
  S14   243.7724
  S15   366.1598;

set DEMANDS :=  D01   D02   D03   D04   D05   D06   D07   D08   D09   D10
            D11   D12   D13   D14   D15   D16   D17   D18   D19   D20
            D21   D22   D23   D24   D25   D26   D27   D28   D29   D30
            D31   D32   D33   D34   D35   D36   D37   D38   D39   D40
            D41   D42   D43   D44   D45;

param DemUnits :=
  D01     2.0000
  D02     9.0000
  D03     4.0000
  D04     1.0000
  D05     3.0000
  D06    10.0000
  D07     1.0000
  D08     6.0000
  D09     6.0000
  D10     5.0000
  D11     7.0000
  D12     2.0000
  D13     8.0000
  D14     9.0000
  D15    10.0000
  D16     2.0000
  D17     1.0000
  D18     2.0000
  D19     1.0000
  D20     4.0000
  D21     3.0000
  D22     8.0000
  D23     2.0000
  D24     1.0000
  D25     9.0000
  D26     1.0000
  D27     9.0000
  D28     5.0000
  D29     9.0000
  D30    10.0000
  D31     4.0000
  D32     7.0000
  D33     4.0000
  D34     3.0000
  D35     5.0000
  D36     1.0000
  D37    10.0000
  D38    10.0000
  D39     4.0000
  D40    10.0000
  D41     7.0000
  D42     7.0000
  D43     8.0000
  D44     8.0000
  D45    10.0000;

set WORK_ROUTES[D01]    :=  PR0001 PR0002 PR0003 PR0004 PR0005;
set REST_ROUTES[D01]    :=  BR0001 BR0002 BR0003 BR0004 BR0005 BR0006 BR0007 BR0008 BR0009;
set WORK_ROUTE_VECTORS[D01,   PR0001] :=  S01;
set WORK_ROUTE_VECTORS[D01,   PR0002] :=  S01;
set WORK_ROUTE_VECTORS[D01,   PR0003] :=  S03 S08 S04;
set WORK_ROUTE_VECTORS[D01,   PR0004] :=  S02 S06 S05;
set WORK_ROUTE_VECTORS[D01,   PR0005] :=  S01;
set REST_ROUTE_VECTORS[D01,   BR0001] :=  S03 S08 S04;
set REST_ROUTE_VECTORS[D01,   BR0002] :=  S02 S06 S05;
set REST_ROUTE_VECTORS[D01,   BR0003] :=  S03 S09 S14 S10 S04;
set REST_ROUTE_VECTORS[D01,   BR0004] :=  S02 S07 S13 S11 S05;
set REST_ROUTE_VECTORS[D01,   BR0005] :=  S03 S08 S10 S15 S11 S05;
set REST_ROUTE_VECTORS[D01,   BR0006] :=  S01;
set REST_ROUTE_VECTORS[D01,   BR0007] :=  S02 S07 S13 S15 S10 S04;
set REST_ROUTE_VECTORS[D01,   BR0008] :=  S02 S07 S12 S09 S08 S04;
set REST_ROUTE_VECTORS[D01,   BR0009] :=  S03 S09 S12 S13 S11 S05;

set WORK_ROUTES[D02]    :=  PR0006 PR0007 PR0008 PR0009 PR0010;
set REST_ROUTES[D02]    :=  BR0010 BR0011 BR0012 BR0013 BR0014 BR0015 BR0016 BR0017 BR0018 BR0019 BR0020;
set WORK_ROUTE_VECTORS[D02,   PR0006] :=  S02;
set WORK_ROUTE_VECTORS[D02,   PR0007] :=  S02;
set WORK_ROUTE_VECTORS[D02,   PR0008] :=  S02;
set WORK_ROUTE_VECTORS[D02,   PR0009] :=  S01 S05 S06;
set WORK_ROUTE_VECTORS[D02,   PR0010] :=  S03 S09 S12 S07;
set REST_ROUTE_VECTORS[D02,   BR0010] :=  S01 S05 S06;
set REST_ROUTE_VECTORS[D02,   BR0011] :=  S03 S09 S12 S07;
set REST_ROUTE_VECTORS[D02,   BR0012] :=  S03 S08 S04 S05 S06;
set REST_ROUTE_VECTORS[D02,   BR0013] :=  S01 S05 S11 S13 S07;
```

```
set REST_ROUTE_VECTORS[D02,   BR0014] :=  S03 S08 S10 S15 S13 S07;
set REST_ROUTE_VECTORS[D02,   BR0015] :=  S02;
set REST_ROUTE_VECTORS[D02,   BR0016] :=  S03 S08 S10 S15 S11 S06;
set REST_ROUTE_VECTORS[D02,   BR0017] :=  S03 S08 S10 S14 S12 S07;
set REST_ROUTE_VECTORS[D02,   BR0018] :=  S01 S04 S08 S09 S12 S07;
set REST_ROUTE_VECTORS[D02,   BR0019] :=  S01 S04 S10 S15 S13 S07;
set REST_ROUTE_VECTORS[D02,   BR0020] :=  S03 S09 S12 S13 S11 S06;

set WORK_ROUTES[D03]    :=  PR0011 PR0012 PR0013 PR0014 PR0015;
set REST_ROUTES[D03]    :=  BR0021 BR0022 BR0023 BR0024 BR0025 BR0026 BR0027 BR0028 BR0029;
set WORK_ROUTE_VECTORS[D03,   PR0011] :=  S03;
set WORK_ROUTE_VECTORS[D03,   PR0012] :=  S03;
set WORK_ROUTE_VECTORS[D03,   PR0013] :=  S03;
set WORK_ROUTE_VECTORS[D03,   PR0014] :=  S01 S04 S08;
set WORK_ROUTE_VECTORS[D03,   PR0015] :=  S03;
set REST_ROUTE_VECTORS[D03,   BR0021] :=  S01 S04 S08;
set REST_ROUTE_VECTORS[D03,   BR0022] :=  S02 S07 S12 S09;
set REST_ROUTE_VECTORS[D03,   BR0023] :=  S02 S06 S05 S04 S08;
set REST_ROUTE_VECTORS[D03,   BR0024] :=  S01 S04 S10 S14 S09;
set REST_ROUTE_VECTORS[D03,   BR0025] :=  S02 S07 S13 S15 S10 S08;
set REST_ROUTE_VECTORS[D03,   BR0026] :=  S03;
set REST_ROUTE_VECTORS[D03,   BR0027] :=  S02 S07 S13 S15 S14 S09;
set REST_ROUTE_VECTORS[D03,   BR0028] :=  S02 S07 S12 S14 S10 S08;
set REST_ROUTE_VECTORS[D03,   BR0029] :=  S02 S06 S11 S13 S12 S09;

…

set WORK_ROUTES[D45]    :=  PR0221 PR0222 PR0223 PR0224 PR0225;
set REST_ROUTES[D45]    :=  BR0377 BR0378 BR0379 BR0380 BR0381 BR0382 BR0383 BR0384 BR0385 BR0386;
set WORK_ROUTE_VECTORS[D45,   PR0221] :=  S15;
set WORK_ROUTE_VECTORS[D45,   PR0222] :=  S15;
set WORK_ROUTE_VECTORS[D45,   PR0223] :=  S13 S12 S14;
set WORK_ROUTE_VECTORS[D45,   PR0224] :=  S15;
set WORK_ROUTE_VECTORS[D45,   PR0225] :=  S13 S12 S14;
set REST_ROUTE_VECTORS[D45,   BR0377] :=  S13 S12 S14;
set REST_ROUTE_VECTORS[D45,   BR0378] :=  S11 S05 S04 S10;
set REST_ROUTE_VECTORS[D45,   BR0379] :=  S13 S12 S09 S08 S10;
set REST_ROUTE_VECTORS[D45,   BR0380] :=  S11 S06 S07 S12 S14;
set REST_ROUTE_VECTORS[D45,   BR0381] :=  S13 S07 S06 S05 S04 S10;
set REST_ROUTE_VECTORS[D45,   BR0382] :=  S15;
set REST_ROUTE_VECTORS[D45,   BR0383] :=  S11 S06 S02 S03 S08 S10;
set REST_ROUTE_VECTORS[D45,   BR0384] :=  S11 S06 S02 S03 S09 S14;
set REST_ROUTE_VECTORS[D45,   BR0385] :=  S13 S07 S02 S03 S08 S10;
set REST_ROUTE_VECTORS[D45,   BR0386] :=  S13 S07 S02 S03 S09 S14;
```

Appendix H Multi-flow SBPP

Appendix H.1 – AMPL codes of New Multi-flow SBPP Mechanism

```
# SBPP mechanism under full single failure scenario
# October 2015 by Wenjing Wang

# ************************
# TOPOLOGY DEFINITION
# ************************

set SPANS;
# Set of all physical spans in the network.

set DEMANDS;
# Set of all demands that exist.

param Cost{k in SPANS};
# The cost of a unit of working or spare capacity on span k.

# ************************
# DESCRIPTION OF WORKING DEMANDS AND THEIR NORMAL ROUTING
# ************************

param DemUnits{r in DEMANDS} default 0;
# Number of demand units between node pair r.

set WORK_ROUTES{r in DEMANDS};

set WORK_ROUTE_VECTORS{r in DEMANDS, p in WORK_ROUTES[r]} within {j in SPANS};

param MaxFlow := sum {r in DEMANDS} DemUnits[r];
# Used for upper bounds on flow and capacity variables.

# ************************
# FAILURE SCENARIO DEFINITIONS
# ************************

set DEMANDS_AFFECTED{i in SPANS} := {r in DEMANDS : exists {p in WORK_ROUTES[r], k in WORK_ROUTE_VECTORS[r, p]} k = i};
# This builds a set of the demand pairs that are damaged by each possible span failure i

set WORK_ROUTES_AFFECTED{i in SPANS, r in DEMANDS_AFFECTED[i]} := {p in WORK_ROUTES[r] : exists {k in
WORK_ROUTE_VECTORS[r, p]} k = i};
# This generates the list of working routes affected by failure of span i.

# ************************
# ELIGIBLE ROUTES FOR PATH-LEVEL RESTORATION OF O-D PAIRS
# ************************

set REST_ROUTES{r in DEMANDS, p in WORK_ROUTES[r]};

set REST_ROUTE_VECTORS{r in DEMANDS, p in WORK_ROUTES[r], b in REST_ROUTES[r, p]} within {j in SPANS};

# ************************
# VARIABLES
# ************************

var wf{r in DEMANDS, p in WORK_ROUTES[r]} >=0, <=MaxFlow integer;
# The amount of working flow routed over working route q for demand relation r.

var bf{r in DEMANDS, p in WORK_ROUTES[r], b in REST_ROUTES[r, p]} >=0, <=MaxFlow integer;
# There is one restoration flow assignment variable for each REST_ROUTES with
# regard to each primary working route.

var spare{j in SPANS} >=0, <=MaxFlow integer;
# Total number of spare links placed on span j.

var work{j in SPANS} >=0, <=MaxFlow integer;
# Number of working wavelengths placed on span j.

# ************************
# OBJECTIVE FUNCTION
# ************************

minimize TotalCost: sum{k in SPANS} (work[k] * Cost[k] + spare[k] * Cost[k]);

# ************************
# CONSTRAINTS
# ************************
```

```
subject to c_01{r in DEMANDS}:
sum{p in WORK_ROUTES[r]} wf[r, p] = DemUnits[r];

subject to c_02{k in SPANS}:
work[k] = sum{r in DEMANDS, p in WORK_ROUTES[r]: exists {j in WORK_ROUTE_VECTORS[r, p]} j = k} wf[r, p];

subject to c_03 {r in DEMANDS, p in WORK_ROUTES[r]}:
sum{b in REST_ROUTES[r, p]} bf[r, p, b] = wf[r, p];

subject to c_04 {i in SPANS, k in SPANS: i <> k}:
spare[k] >= sum {r in DEMANDS_AFFECTED[i], p in WORK_ROUTES_AFFECTED[i, r], b in REST_ROUTES[r, p]:
exists {j in REST_ROUTE_VECTORS[r, p, b]} j = k} bf[r, p, b];
```

Appendix H.2 – An Example of *.Dat Files for New SBPP Model

Note: only part of the dapa is shown here but full seat of data is available upon request.

```
# This data prep file is for SBPP model file
# Created by Wenjing Wang in October 2015
# 10-node 15-span netowrk, routeLimit=5

set SPANS := S01 S02 S03 S04 S05 S06 S07 S08 S09 S10
         S11 S12 S13 S14 S15;

param Cost :=
 S01  208.5857
 S02  242.6706
 S03  245.1775
 S04  289.1730
 S05  617.1102
 S06  286.9042
 S07  283.4255
 S08  211.5490
 S09  269.2675
 S10  398.3780
 S11  306.8061
 S12  199.0226
 S13  304.8442
 S14  243.7724
 S15  366.1598;

set DEMANDS := D01 D02 D03 D04 D05 D06 D07 D08 D09 D10
          D11 D12 D13 D14 D15 D16 D17 D18 D19 D20
          D21 D22 D23 D24 D25 D26 D27 D28 D29 D30
          D31 D32 D33 D34 D35 D36 D37 D38 D39 D40
          D41 D42 D43 D44 D45;

param DemUnits :=
 D01   2.0000
 D02   9.0000
 D03   4.0000
 D04   1.0000
 D05   3.0000
 D06  10.0000
 D07   1.0000
 D08   6.0000
 D09   6.0000
 D10   5.0000
 D11   7.0000
 D12   2.0000
 D13   8.0000
 D14   9.0000
 D15  10.0000
 D16   2.0000
 D17   1.0000
 D18   2.0000
 D19   1.0000
 D20   4.0000
 D21   3.0000
 D22   8.0000
 D23   2.0000
 D24   1.0000
 D25   9.0000
 D26   1.0000
 D27   9.0000
 D28   5.0000
 D29   9.0000
 D30  10.0000
 D31   4.0000
 D32   7.0000
 D33   4.0000
 D34   3.0000
 D35   5.0000
 D36   1.0000
 D37  10.0000
 D38  10.0000
 D39   4.0000
 D40  10.0000
 D41   7.0000
 D42   7.0000
 D43   8.0000
 D44   8.0000
 D45  10.0000;

set WORK_ROUTES [D01] := PR0001 PR0002 PR0003 PR0004 PR0005;
set REST_ROUTES[D01, PR0001] := BR0001;
set REST_ROUTES[D01, PR0002] := BR0001;
set REST_ROUTES[D01, PR0003] := BR0002;
set REST_ROUTES[D01, PR0004] := BR0002;
set REST_ROUTES[D01, PR0005] := BR0001;
set WORK_ROUTE_VECTORS[D01, PR0001] := S01;
set WORK_ROUTE_VECTORS[D01, PR0002] := S01;
set WORK_ROUTE_VECTORS[D01, PR0003] := S03 S08 S04;
set WORK_ROUTE_VECTORS[D01, PR0004] := S02 S06 S05;
set WORK_ROUTE_VECTORS[D01, PR0005] := S01;
set REST_ROUTE_VECTORS[D01, PR0001, BR0001] := S03 S08 S04;
set REST_ROUTE_VECTORS[D01, PR0002, BR0001] := S03 S08 S04;
set REST_ROUTE_VECTORS[D01, PR0003, BR0002] := S01;
set REST_ROUTE_VECTORS[D01, PR0004, BR0002] := S01;
set REST_ROUTE_VECTORS[D01, PR0005, BR0001] := S03 S08 S04;

set WORK_ROUTES [D02] := PR0006 PR0007 PR0008 PR0009 PR0010;
set REST_ROUTES[D02, PR0006] := BR0003;
set REST_ROUTES[D02, PR0007] := BR0003;
set REST_ROUTES[D02, PR0008] := BR0003;
set REST_ROUTES[D02, PR0009] := BR0004;
set REST_ROUTES[D02, PR0010] := BR0004;
set WORK_ROUTE_VECTORS[D02, PR0006] := S02;
set WORK_ROUTE_VECTORS[D02, PR0007] := S02;
set WORK_ROUTE_VECTORS[D02, PR0008] := S02;
set WORK_ROUTE_VECTORS[D02, PR0009] := S01 S05 S06;
set WORK_ROUTE_VECTORS[D02, PR0010] := S03 S09 S12 S07;
set REST_ROUTE_VECTORS[D02, PR0006, BR0003] := S01 S05 S06;
set REST_ROUTE_VECTORS[D02, PR0007, BR0003] := S01 S05 S06;
set REST_ROUTE_VECTORS[D02, PR0008, BR0003] := S01 S05 S06;
set REST_ROUTE_VECTORS[D02, PR0009, BR0004] := S02;
set REST_ROUTE_VECTORS[D02, PR0010, BR0004] := S02;

set WORK_ROUTES [D03] := PR0011 PR0012 PR0013 PR0014 PR0015;
set REST_ROUTES[D03, PR0011] := BR0005;
set REST_ROUTES[D03, PR0012] := BR0005;
set REST_ROUTES[D03, PR0013] := BR0005;
```

```
set REST_ROUTES[D03, PR0014] :=  BR0006;
set REST_ROUTES[D03, PR0015] :=  BR0005;
set WORK_ROUTE_VECTORS[D03,  PR0011] :=  S03;
set WORK_ROUTE_VECTORS[D03,  PR0012] :=  S03;
set WORK_ROUTE_VECTORS[D03,  PR0013] :=  S03;
set WORK_ROUTE_VECTORS[D03,  PR0014] :=  S01 S04 S08;
set WORK_ROUTE_VECTORS[D03,  PR0015] :=  S03;
set REST_ROUTE_VECTORS[D03, PR0011, BR0005] :=  S01 S04 S08;
set REST_ROUTE_VECTORS[D03, PR0012, BR0005] :=  S01 S04 S08;
set REST_ROUTE_VECTORS[D03, PR0013, BR0005] :=  S01 S04 S08;
set REST_ROUTE_VECTORS[D03, PR0014, BR0006] :=  S03;
set REST_ROUTE_VECTORS[D03, PR0015, BR0005] :=  S01 S04 S08;

…

set WORK_ROUTES  [D45]  :=  PR0221 PR0222 PR0223 PR0224 PR0225;
set REST_ROUTES[D45, PR0221] :=  BR0101;
set REST_ROUTES[D45, PR0222] :=  BR0101;
set REST_ROUTES[D45, PR0223] :=  BR0102;
set REST_ROUTES[D45, PR0224] :=  BR0101;
set REST_ROUTES[D45, PR0225] :=  BR0102;
set WORK_ROUTE_VECTORS[D45,  PR0221] :=  S15;
set WORK_ROUTE_VECTORS[D45,  PR0222] :=  S15;
set WORK_ROUTE_VECTORS[D45,  PR0223] :=  S13 S12 S14;
set WORK_ROUTE_VECTORS[D45,  PR0224] :=  S15;
set WORK_ROUTE_VECTORS[D45,  PR0225] :=  S13 S12 S14;
set REST_ROUTE_VECTORS[D45, PR0221, BR0101] :=  S13 S12 S14;
set REST_ROUTE_VECTORS[D45, PR0222, BR0101] :=  S13 S12 S14;
set REST_ROUTE_VECTORS[D45, PR0223, BR0102] :=  S15;
set REST_ROUTE_VECTORS[D45, PR0224, BR0101] :=  S13 S12 S14;
set REST_ROUTE_VECTORS[D45, PR0225, BR0102] :=  S15;
```

Appendix I – Selection of Span's Failure Rate

In order to demonstrate that any value in the range of 2.0×10^{-7}~8.0×10^{-7} is reasonable for a span's failure rate, we will use 2.0×10^{-7} and 8.0×10^{-7} separately in span restoration analysis as well. Figure Appendix H – 1 through Appendix H – 3 show network availability for the 30-node network family with span's unit failure rate being 3.4×10^{-7}, 2.0×10^{-7}, and 8.0×10^{-7}, respectively. As the figures shown, the trends of network availability for these three situations are exactly the same.

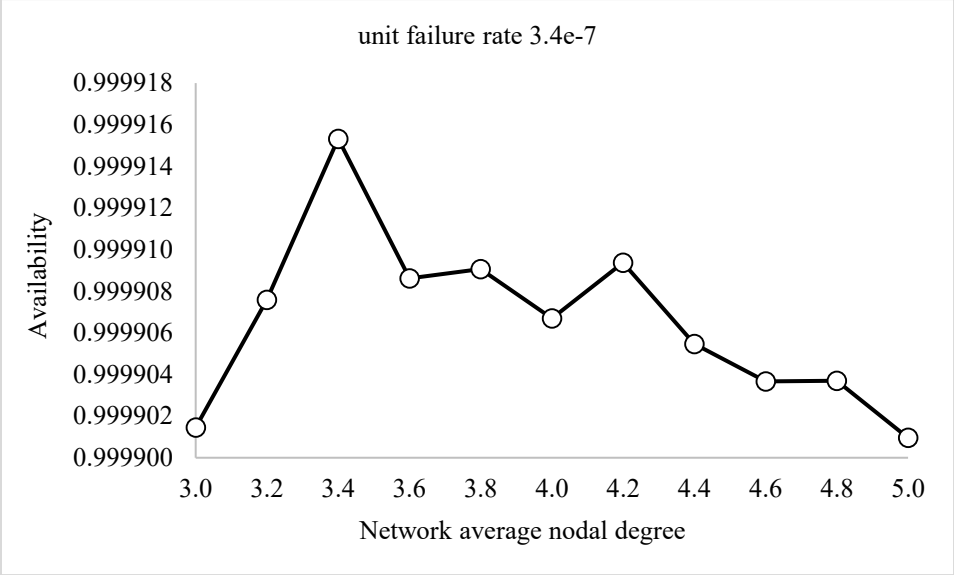


Figure Appendix H – 1 network availability for the 30-node network family with span's unit failure rate being 3.4×10^{-7}

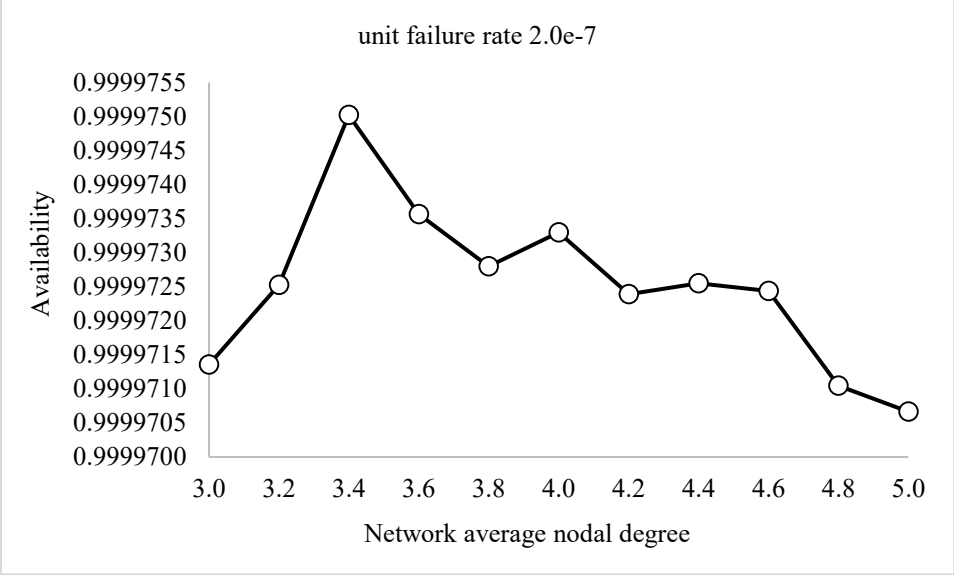


Figure Appendix H – 2 network availability for the 30-node network family with span's unit failure rate being 2.0×10^{-7}

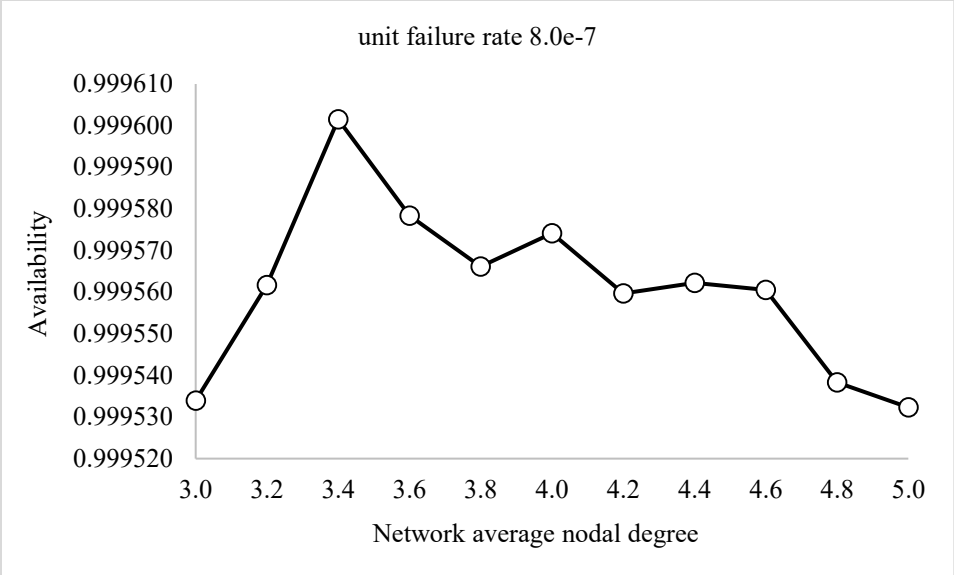


Figure Appendix H – 3 network availability for the 30-node network family with span's unit failure rate being 8.0×10^{-7}