ADVANCES ON HEURISTICS FOR SURVIVABLE LARGE-SCALE NETWORK DESIGN

by

Samira Doostie

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

The increasing importance of telecommunication networks is evident to everyone. These networks are the infrastructure that facilitates communication and transactions worldwide. Most of our daily routine activities are highly dependent on the proper function of these networks. Therefore, properly addressing and resolving issues to minimize network failures and service outages by improving the network performance and boosting their survivability is extremely important.

The work in this thesis has a special focus on expanding and improving the survivable telecommunication network design process. This thesis presents computationally efficient approaches for telecommunication network design including network topology design, routing traffic demands, and establishing and analyzing survivability against complex failure scenarios. The approaches described in this work are designed to create benefits by creating synergies between deterministic and non-deterministic approaches based on Integer Linear Programing and heuristic algorithms. The methodologies and the experimental analysis within this work, provide a comprehensive set of tools for network designers who can benefit when dealing with various design constraints.

Preface

Parts of this thesis have been published as a journal paper with Samira Doostie as the primary author and Professor Tetsuhei Nakashima-Paniagua and Professor John Doucette as second and third authors.

Parts of Chapter 2 and Chapter 3 of this thesis have been published as S. Doostie, T. Nakashima-Paniagua, and J. Doucette, “A Novel Genetic Algorithm-Based Methodology for Large-Scale Fixed Charge plus Routing Network Design Problem with Efficient Operators,” IEEE Access, vol. 9, pp. 114836–114853, 2021. Samira was responsible for the conceptualization, methodology, analysis, and writing the manuscript. Professor John Doucette and Professor Tetsuhei Nakashima-Paniagua were involved with the concept development, methodology, and editing of the manuscript.

Portions of Chapters 2 and Chapter 4 are slated for publication as a conference paper: S. Doostie, T. Nakashima-Paniagua, and J. Doucette, “An Improvement on the Network Topology Design and Routing Problem for Large-Scale Networks”. The specific conference that will be the recipient of this work is to be determined (TBD). Samira was responsible for the conceptualization, methodology, analysis, and writing the manuscript. Professor John Doucette and Professor Tetsuhei Nakashima-Paniagua were involved with the concept development, methodology, and editing of the manuscript.

Parts of Chapter 2, along with portions of Chapter 5, and Chapter 6 will be submitted for publication as two journal papers as S. Doostie, T. Nakashima-Paniagua,

and J. Doucette, with the following respective titles: “A Novel Design Approach for Dual-Failure Span-Restorable Large-Scale Networks using Integer Linear Programming” and “A Novel Genetic Algorithm-based Approach for Design of Dual-Failure Span-Restorable Large-Scale Networks”. The specific journals where these works will be submitted are TBD. Samira was responsible for the conceptualization, methodology, analysis, and writing the manuscript. Professor John Doucette and Professor Tetsuhei Nakashima-Paniagua were involved with the concept development, methodology, and editing of the manuscript.

To my family who has made so many sacrifices for me with love.

Acknowledgment

This work is a direct result of the support, guidance, and intelligence of many people.

First, I would like to greatly thank my supervisors Professor John Doucette and Dr. Tetsuhei Nakashima who have been not only my technical supervisors but also my professional mentors. They guided me every step of the way with their knowledge and insights and enlightened me with their wisdom throughout this incredible and demanding journey. So, I would like to take this opportunity and graciously thank both of them for their profound support and help.

I also would like to highly appreciate and thank Professor Michael Lipsett for his invaluable insights and guidance throughout my Ph.D. research. I was fortunate to benefit from his knowledge and suggestions as a member of my supervisory committee.

Moreover, I would like to greatly thank other professors who served as examiners in my Ph.D. candidacy exam and defence exam.

Also, I would like to thank the Alberta Innovates Graduate Student Scholarship, The Natural Science and Engineering Research Council of Canada (NSERC) under Discovery Grant 2017-06198, Alberta Graduate Excellence Scholarship, University of Alberta Graduate Fellowship, and University of Alberta Doctoral Recruitment Scholarship for their financial support for this research work.

Lastly, I would like to wholeheartedly thank my friends and family who have always shown me their love and support, especially during these hard times of global pandemic from across the world.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADM **–**Add-Drop Multiplexer

APS – Automatic Protection Switching

CO **–** Central Office

FCR **–** Fixed Charge plus Routing

GA **–** Genetic Algorithm

IGA **–** Improved Genetic Algorithm

ILP **–** Integer Linear Programming

MTRS **–** Mesh Topology Routing and Spare capacity

OXC **–** Optical Cross-Connect

Dual-failure RN-FCS **–** Reserved Network Fixed Charge plus Spare capacity problem for the restoration of dual failures

SBPP – Shared Backup Path Protection

SR – Span Restoration

TSP **–** Travelling Salesman Problem

WDM **–** Wavelength Division Multiplexing

Chapter 1 : Introduction

Today, the quality of our daily lives depends on the quality of Internet-based services, which are mainly provided and managed by modern communication networks. Communication networks facilitate services such as emergency services, business management infrastructures, social media connections, transportation platforms, etc. based on audio/video data transfer worldwide [1], [2]. This level of dependency on critical services, along with the universal growth in the demand for such services, calls for larger communication networks with higher levels of quality and functionality [3].

The three main constituent tiers of transport networks that facilitate the connection among individuals around the world are *access networks*, *metro networks*, and *long-haul networks* [4]. The access networks provide a connection between the local individuals and central offices (COs). At a higher level, there are metro networks that facilitate the connection among the COs within local distances. Finally, there are long-haul networks that can provide a connection among the metro networks all around the world [4].

The foundation of the communication networks is a transport network (i.e., backbone) that facilitates the transfer of various types of data through many physical fibre optic cables. The data transmitted via fibre optic cables travel over specific routes between desired origin-destination nodal points (terminals) equipped with optical cross-connect switches (OXCs) or add-drop multiplexers (ADMs) [4], [5]. A breakthrough technology in the transport network transmission capacity is Wavelength Division Multiplexing (WDM) technology in which a fibre optic cable is able to carry several

wavelengths of data simultaneously [4], [6], [7]. Throughout this thesis, for the network topology design, the networks were simulated in the form of graphs where the nodal devices were represented by nodes and fiber optic cables were represented by the spans that connect the nodes [8], [9].

The network's functionality is dependent on its constituent components, including the nodes and spans. The functionality of a network may be disrupted due to various causes such as severe weather conditions, acts of sabotage, outdated or over-loaded equipment, or accidents, that can cause component malfunctioning or failures [10], [11]. Such component failures range over a vast variety of failures from a single or dual node or span failure, to massive regional failures that affect several components simultaneously [10], [12]. As the components (e.g., nodes and spans) along the *working routes* –routes that primarily transfer the data traffic- fail, the route becomes interrupted and unable to transfer the data as before. Thus, the affected data will be switched from the failed route to the *backup route* [13]. The lack of sufficient backup routes results in a service outage. In recent years, both government and private sectors in Canada have faced sizeable financial losses due to communication service disruptions and outages [14]–[17]. To avoid losses and improve the quality of services, we need sustainable networks that are equipped with survivability mechanisms to overcome such component failures and minimize service disruptions. There are various survivability mechanisms designed for the restoration of the network's functionality upon component failures [10]. Employing survivable networks provides levels of protection against failures and consequently increases the availability of network services [18]–[20].

1.1 Motivation & Objectives

In recent decades, communications have faced many challenges and required upgrades, and researchers were able to invent novel and efficient solution approaches for emerging problems. However, previously developed approaches might lose their effectiveness in tackling large-scale network design problems due to: (1) the rapid growth of communication networks during the past couple of years, their protocols, governing constraints, requirements, and limitations have been changed and become more complex than before, and (2) the network design problem itself has a complexity level of NP-hard [1], [11], [21]–[24]. Thus, in this thesis, we aim to provide efficient solution approaches for the large-scale network design problem. Following, our main objectives, which are threefold, are listed:

1. Optimal topology design of large-scale networks:

   We propose a novel heuristic approach based on a genetic algorithm (GA) for the topology design of large-scale networks. Given the set of nodes, a set of traffic demands to be routed between specified origin-destination nodes, and the costs associated with each span establishment and working capacity allocation, our approach finds the optimal network topology in terms of the optimal span establishment to accommodate the routing of all the required traffic demands with minimum cost in reasonable processing time. We also study the effect of various traffic patterns on the efficiency of the proposed approach.

2. Survivable topology design of large-scale networks based on a deterministic approach:

We proposed a deterministic-based approach for the survivable topology design and spare capacity allocation. Given a fixed set of nodes, a set of established spans with working traffic on them, and the costs associated with additional span establishment and spare capacity allocation, we studied the minimum cost topology design of large-scale networks that are survivable against dual span failures.

3. Survivable topology design of large-scale networks based on a heuristic approach:

   We proposed a Genetic Algorithm-based approach for the survivable topology design and spare capacity allocation. Given a fixed set of nodes, a set of established spans with working traffic on them, and the costs associated with additional span establishment and spare capacity allocation, we studied the minimum cost topology design of large-scale networks that are survivable against dual span failures.

1.2 Thesis Outline

This thesis comprises of 7 chapters covering the achievements accomplished to satisfy the three main goals mentioned previously. In Chapter 2, we present a comprehensive background on the thesis topic. We introduced the fundamental mathematical concepts and used terminology in Sub-section 2.1 including the graph theory concepts, the mathematical notation for representing the problem, and employed solution approaches. Moreover, in Sub-section 2.2 we represented the fundamental sub-problems each of which correlates to one of the main goals of this research.

In Chapter 3, we present a novel heuristic approach based on a GA for the optimal topology design of large-scale networks in the format of a journal paper. This chapter starts with an abstract of the research work, followed by a comprehensive introduction to the problem including the related works. We extensively described the designed GA, its constitutive operators and the methodology that we used to fine-tune those operators. Finally, we compared the obtained results (over 18 different test cases) to the existing benchmark that provides the exact optimal solution to the problem.

In Chapter 4, we present an improvement in the designed GA in Chapter 3 in the format of a conference paper. The contribution of Chapter 4 is focused on the pattern of the traffic demands that are going to be routed over the networks in a large-scale network design problem. We studied the impacts of the order of routing the traffic demands according to the traffic demand's magnitude, on the total efficiency of the designed algorithm. We analyzed two different traffic demand patterns as scattered and clustered (with hub nodes) on 20 different test cases with 60 and 100 nodes.

As the main content of Chapter 5, we present a heuristic approach for dual failure span restorable large-scale network design problem using pre-enumeration of backup routes based on an ILP model. We assumed the nodal points in a network are fixed and there is a set of spans in the network carrying the working traffics. Our goal was to design a minimum-cost network topology that is survivable against any dual span failure among the spans carrying the working traffic.

In Chapter 6, as an advancement on the contributions of Chapter 5, we introduced a novel heuristic approach based on a GA for the design of dual failure span

restorable large-scale network design problem. The introduced GA is embedded in a problem-specific heuristic algorithm for routing the affected traffics upon any dual span failure. The GA's along with the heuristic algorithm's parameters were determined over a comprehensive set of tuning tests. The obtained results were validated using the approaches introduced in Chapter 5 and they were comparable concerning the efficiency of the solving process.

In Chapter 7, a comprehensive conclusion on the Ph.D. thesis work and future directions is presented.

Chapter 2 : Background

In this chapter, a background on the employed mathematical and computational tools and concepts are provided, followed by a detailed background on the related research works and the thesis outline. In this thesis, sections of the following chapters are edited/adapted from parts of our work ready to be submitted and/or accepted for publication as conference/journal papers[1]. Appropriate citations and references are provided to identify such publications.

2.1 Mathematical Terminology

In this thesis, we employ various mathematical tools and notations for defining the structure of the network design problem and representing the problem's solution. In this section, an overview of the concepts and notation that we used in this thesis is provided.

2.1.1 Graph Theory Representation of Networks

According to Harary's definition of graph [8], [9], a graph consists of a set of points and lines in which every line connects two points.

Networks can be represented by their corresponding graphs. To conform with the telecommunication network's phraseology, from now on we use *node* and *span* instead

[1] Parts of this chapter are taken and/or adapted from our journal paper in IEEE Access [129].
Portions of this chapter and Chapter 4 are slated for publication as a conference paper: S. Doostie, T. Nakashima-Paniagua, and J. Doucette, "An Improvement on the Network Topology Design and Routing Problem for Large-Scale Networks", conference TBD.
Portions of this chapter and Chapter 5 are slated for publication as a journal paper: S. Doostie, T. Nakashima-Paniagua, and J. Doucette, "A Novel Design Approach for Dual-Failure Span-Restorable Large-Scale Networks using Integer Linear Programming", journal TBD.
Portions of this chapter and Chapter 6 are slated for publication as a journal paper: S. Doostie, T. Nakashima-Paniagua, and J. Doucette, "A Novel Genetic Algorithm-based Approach for Design of Dual-Failure Span-Restorable Large-Scale Networks", journal TBD.

of *point* and *line* in the graph definition. Thus, we can represent a graph G with N nodes and S spans as G (N, S).

The structure of a graph that determines how its nodes and spans are established in relation to each other is called the graph's topology. If the end nodes of a span are the same node, that span is called a *loop*. If there is more than one span connecting the same two nodes, they are *parallel spans*. A graph with no loops and no parallel spans is a *simple* graph. In a graph with N nodes, if a span exists between every pair of nodes, we call it a *complete* graph (K_N). In a complete graph with N nodes, the maximum number of spans is calculated using Equation (1). If there are at least two nodes in a graph that are not connected (via one or several intermediate spans), the graph is *disconnected*. The abovementioned graph topologies are depicted in Figure 2.1.

$$Maximum\ number\ of\ spans = \frac{N \times (N-1)}{2} \quad (1)$$

The followings are other graph theory terminologies used in this thesis [25]:

1. Nodal degree: the number of spans connected to a node is called the *nodal degree* of that node.
2. Average nodal degree: the *average nodal degree* of all the nodes in a graph is called the graph's average nodal degree and it can be calculated using Equation (2).

$$Graph's\ average\ nodal\ degree = \frac{2 \times (|S|)}{|N|} \quad (2)$$

3. Order: The number of nodes in a graph is called the *graph's order*.
4. Stub: A node with a degree of one is called a *stub*.

5. Link: abstraction of a single unit of data (bandwidth) between two nodes in a graph is called a *link*.
6. Span: an accumulated set of all the links between two nodes in a graph is called a span.

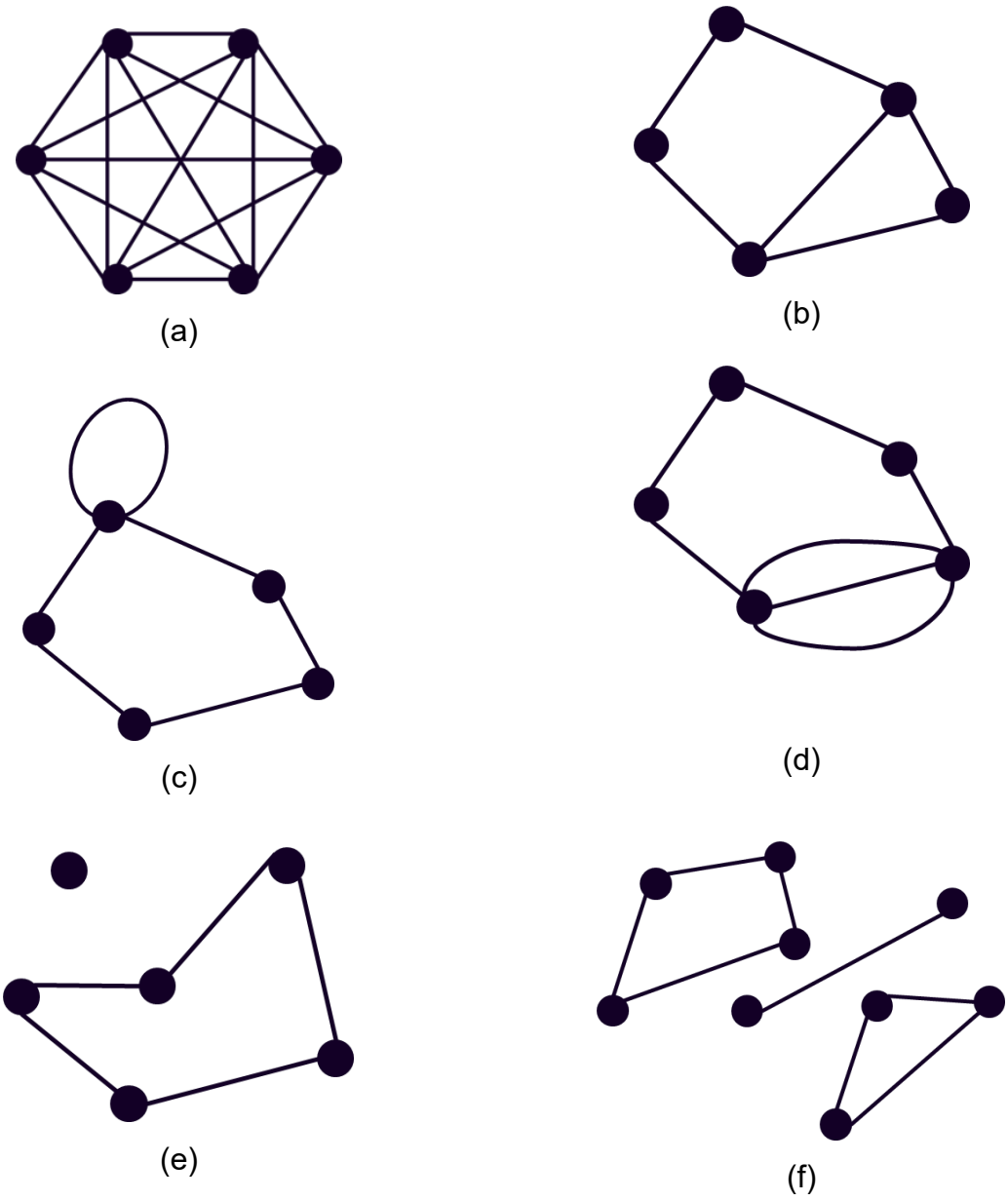


Figure 2.1. (a) A complete K_6 graph with 6 nodes and 15 spans, (b) a simple graph with 5 nodes and 6 spans, (c) a graph with a loop, (d) a graph with three parallel spans between two of its nodes, (e) a disconnected graph consisting of two components, (f) a disconnected graph consisting of three components.

7. Path: A sequence of adjacent links with distinct nodes that starts from a node and ends at another node is called a path. The node that the path starts from, is called the “origin node” and the node that the path ends at, is called the “destination node”.

8. Route: A concatenation of adjacent spans with distinct nodes that starts and ends between an origin-destination node-pair.
9. Cycle: A closed route that may pass through several nodes and its start and end nodes are the same.
10. Straddling Span: A span whose end nodes are on a cycle but the span itself is not a part of the cycle.
11. Two-connected graph: If there are at least two span-disjoint paths between every pair of nodes in the graph, the graph is two-connected.
12. N-connected graph: If there are at least N span-disjoint paths between every pair of nodes in the graph, the graph is N-connected.
13. Bi-connected graph: If there are at least two node-disjoint paths between every pair of nodes in the graph, the graph is bi-connected.

An example of a two-connected and a bi-connected graph has been depicted in Figure 2.2.

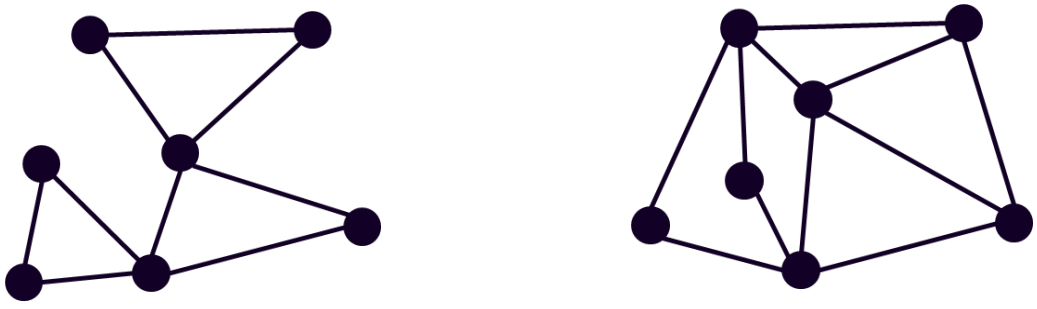


Figure 2.2. Examples of graphs of the order of 7 (a) a two-connected graph, (b) a bi-connected graph.

2.1.2 Parameters, Sets, and Variable notation

An unordered collection of elements (e.g., numbers, names, objects, etc.) is called a *set*. For instance, a set Z that consists of odd numbers between 0 and 8 can be represented by Z= {1, 3, 5, 7}. Here, the numbers 1, 3, 5, and 7 are members of set Z. The science of mathematical analysis of sets is known as the *set theory* [26]. The followings are the mathematical notation of sets and fundamental set operations that we used in this thesis.

1. Empty set (∅): A set with no elements is an empty set. An empty set A can be represented by either A={ } or A= ∅.
2. Equality (=): Two sets A and B are equal if all of the elements of A exist in B and vice versa (A = B).
3. Membership (∈): An element of a set is a member of that set. For example, for the set Z mentioned before, we can say 3 is a member of the set Z or 3 ∈ Z.
4. Cardinality (| |): The total number of elements in a set is the cardinality of that set. For instance, the mentioned set Z has four elements or |Z|=4.
5. Subset (⊂): Set B is a subset of set A if all the elements of B exist in A or B ⊂ A.
6. Not subset (⊄): If there exists an element in B that does not exist in A, the set B is not a subset of set A or B ⊄ A. Accordingly, an slash / on equality and membership signs also negates the meaning of those signs.
7. Universal quantifier (∀): Universal quantifier points out to all of the members of a set. For example, ∀x ∈ Z means for all of the members (x) of set Z.

8. Existential quantifier (∃): Existential quantifier denotes that there exists at least one element. For example, ∃x ∈Z | x > 3 represents that there exists a member (x) of set Z such that x is greater than 3.
9. Union (∪): The union of two (or more) sets is a set of all of the elements in the two (or more) sets. For example, if W={2,3,6} and Z={1,3,5,7}, then the union of A and Z is A ∪ Z={1,2,3,5,6,7}.
10. Intersection (∩): The intersection of two (or more) sets is a set that includes only the elements that are a member of both (or all) of the sets. For example, if W={2,3,6} and Z={1,3,5,7}, then the intersection of A and Z is A ∩ Z={3}.

Employing the set notations, we can define the sub-graph concept. Considering a graph $G(N_1, S_1)$ where N_1 is the set of nodes and S_1 is the set of spans that connect the nodes, we can write $N_1 = \{n_1, n_2, \dots, n_{|N|}\}$ and $S_1 = \{s_1, s_2, \dots, s_{|S|}\}$. If there exists a graph $P(N_2, S_2)$ that $N_2 \subset N_1$ and $S_2 \subset S_1$, we can say that graph P is a sub-graph of G. Figure 2.3 depicts a complete graph K_5 with two of its sub-graphs.

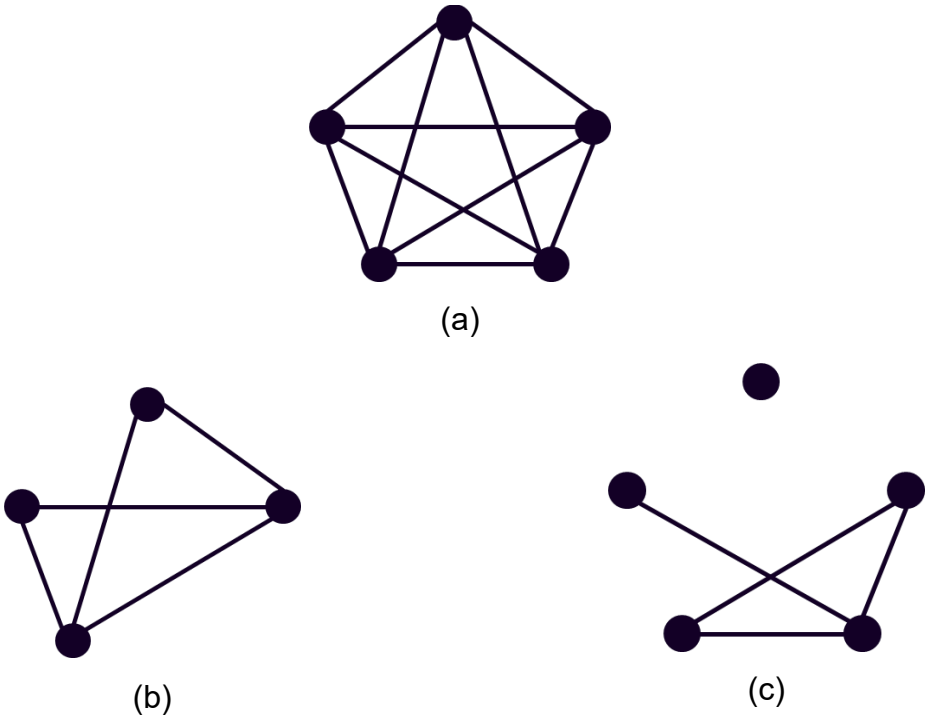


Figure 2.3. (a) A complete graph K_5, (b) a sub-graph of presented K_5 in (a) with 4 nodes and 5 spans, and (c) another sub-graph with all of the nodes and only 4 spans.

2.1.3 Mathematical Programing and Solution Approaches to Optimization Problems

An optimization problem is a problem that searches for one desired solution from a set of various potential solutions [27]. In the mathematical context, the optimization problem can be considered as maximizing/minimizing the value of a mathematical function based on the values of the mathematical function's variables [27]. The pool consisting of all of the possible solutions to an optimization problem is a *search space*. If there is no better solution in the search space than the found solution for an optimization problem, the found solution is called a *global optimal solution* [27]. Another definition used in this thesis is the *combinatorial optimization* problem which refers to an optimization problem that has a finite discrete search space where the solutions are in the form of elements, subsets, or any combination of the elements and subsets of the search space [27].

Mathematical modelling of problems can be abstracted into a combinatorial optimization problem in which the goal is to maximize/minimize the value of a target function named *objective function*. The value of the objective function is determined by the values of *decision variables*. A decision variable is a quantity whose value can be controlled and changed. We seek to determine a value for the decision variables such that the objective function value is maximized or minimized [4]. The maximization/minimization process is done subject to a set of governing *constraints*. The constraints are mathematical limitations and conditions for the decision variable values that govern the search process.

An optimization problem that has a linear objective function, linear constraint equations, and at least one of its decision variables is restricted to be an integer, is an *integer linear programming* (ILP) [28], [29].

2.2 Fundamental Sub-problems in Network Design Problem

In this section, we provide a brief overview of the fundamental sub-problems in the network design problem. Every sub-problem has been addressed in one of the following chapters.

The network design consists of several sub-problems and the specification of these sub-problems may vary depending on the network's application and the operator's discretion. To be specific, the network design problem comprises network topology design and lightpath routing problem, network survivability problem, and network availability analysis. The followings are descriptions of the abovementioned sub-problems in the general network design problem.

2.2.1 Network Topology Design and Lightpath Routing

At the logical level, the network topology can be considered as the position of the nodes (terminals) and the existing spans that connect the nodes. In transport networks, the aim is to route a set of traffic demands between the specified origin and destination nodes as the end nodes of the traffic demand [4]. The traffic demand can be thought of as a bundle of various electrical signals that each has been converted (either at the origin-destination nodes only or at every node along the route) to optical signals with specific wavelengths to be carried over fibre optic cables. In our context, a unit of traffic demand refers to one signal correlated to a specific wavelength [4], [30], [31].

In the network topology design and lightpath routing problem, given a set of fixed nodes and a set of traffic demands between various origin-destination nodes, the aim is to find the optimal network topology (in terms of minimum span establishment and routing cost) while every traffic demand is routed between its origin-destination nodes. Two factors contribute to the network design and routing cost: (1) The first component of the cost is the fixed cost of establishing spans (F) between any two node-pair. Once a span has been established, its establishment cost for the upcoming usage of this span will become zero. (2) The second cost component is the cost of assigning wavelength capacity to the spans that have been established (C), to carry the traffic demand along that span. This problem has been named fixed charge plus routing (FCR) in the literature [32]–[35]. In this thesis, we used the ILP formulation of the FCR problem as one of the benchmarks that provide the exact optimal solution to the network topology design and routing problem.

In network topology design and lightpath routing problem using GA, one starting point is to generate random cycles of nodes as feasible network topologies. Generating initial random graphs based on cycles has been used in the literature [36]. Herein, we benefited from the *Travelling Salesman Problem* (TSP) [37] which intends to find the shortest cycle that starts from a node, goes through all of the nodes once and only once, and ends at the starting node. There are various ways to solve the TSP problem efficiently such as using the Lin-Kernighan algorithm [38] and GA [39] and [40]**.** In this thesis, we employed GA to solve TSP. As having sub-optimal solutions provide our algorithm with a good starting point, we used GA with an initial population of random cycles. We have provided a detailed description of this approach in Chapter 3.

Another element in the topology design and lightpath routing problem using GA is the routing function which is responsible for determining traffic-carrying routes. Based on circumstances, the desired routes might have various requirements such as maximum length, minimum cost, or any other specification. Throughout this thesis, for the minimum-cost network design problem, the length of a route is associated with its total cost. Thus, by finding the shortest route, we are actually finding the cheapest route. Herein, we employ a well-known shortest path routing algorithm named *Dijkstra's algorithm* [41], [42] to find the shortest route for any traffic demand between the traffic demand's designated origin and destination nodes. In the FCR problem, the cost associated with a route comprises the fixed cost of span establishment (for any disestablished span along the route) plus the cost of carrying the amount of lightpath equal to the traffic demand $r\ (d^r)$ by every span along the route. For instance, for a route consisting of 3 disestablished spans (S_1, S_2, and S_3) that require carrying 10 units

of lightpath, the total cost would be equal to $F1 + F2 + F3 + (C1 + C1 + C1) \times 10$. By employing Dijkstra's algorithm, we aim to find the cheapest possible route for traffic demand between the origin and the destination nodes of the traffic demand. The pseudocode of Dijkstra's algorithm has been presented in Algorithm 1. Considering a graph G with specific nodes and spans, and cost parameters (*F* and *C*) associated with every span in G, Dijkstra's algorithm is aimed to find the cheapest route for the traffic demand r between its origin (O_r) and destination (D_r) nodes.

Algorithm 1. Pseudocode of Dijkstra's algorithm

Input:

Graph G(N, S),
Length of a span in terms of its Cost
Traffic demand r (O_r, D_r, d^r)
V: list of visited nodes in G with their corresponding costs
E: list of evaluated nodes in G

Output:

The shortest route for traffic demand r

Dijkstra's algorithm:

1. E(last) ← O_r (Append O_r to E)
2. For ∀n∈N | (n connected to O_r & n≠E(last)) {
   V ← (n, n's immediate predecessor node, n's cost) }
3. E(last) ← minimum cost n from 2
4. While D_r ∉ E {
   5. for n∈N | (n connected to O_r & n∉E) {
      if n∉V {
         V ← (n, n's immediate predecessor node, n's temporary cost)
      else if n's cost > n's temporary cost
         V ← (n, n's immediate predecessor node, n's temporary cost) }
   6. E(last) ← minimum cost n from 5}

2.2.2 Network Survivability

As mentioned earlier in the Chapter1, networks are susceptible to various sources of component failure that may lead to node or span failures. Thus, to have robust networks able to overcome such failures and maintain their required quality of service at all times, networks must be survivable. The global growth in the telecommunication network services along with more complex service requirements, calls for more efficient survivability mechanisms that can act faster with less cost [1]. Network survivability refers to the ability of the network to survive its current traffic affected by component failures. Generally, network survivability refers to two different concepts: (1) network protection in which the protection scheme (including the protection routes and spare

capacities) is defined and in-placed in advance of the occurrence of a specific failure, and it is reserved for that specific failure, only. (2) Network restoration in which the restoration resources (e.g., backup routes and spare capacities) are not on standby but will be utilized upon a failure occurrence [10], [43], [44]. The network survivability needs to be studied from two viewpoints: (1) the survivable topology design in which based on the required survivability pattern, the network must have a minimum level of connectivity, and (2) the spare capacity placement in which on a minimum-cost scheme, a set of spare capacities for rerouting the affected traffics is placed on the survivable topology.

With regards to the survivable topology design, depending on the failure patterns and the required survivability level, the network's topology needs to have a specific connectivity level. Assuming the full restoration only against single-span failure scenarios is required, the network topology should be two-connected. In another word, for spans that carry working traffics, for them to be restorable, there should be at least two span-disjoint routes between every node pair of the graph. As the failure scenarios vary and get more complex, the connectivity requirement for network topology gets more complicated too. For instance, to have a network that is survivable against any simultaneous dual span failure, there should be at least three span-disjoint routes between every node-pair of the graph that carries working traffic. Let us consider the network represented in Figure 2.4. We assume that all of the spans in the network are carrying working traffic. In this case, if spans AB and AE simultaneously fail, there would be no route connecting node A to nodes B and E. On the other hand, upon

simultaneous failure of spans GC and GD, there are other spans connected to node G that can facilitate other routes connecting node G to nodes C and D.

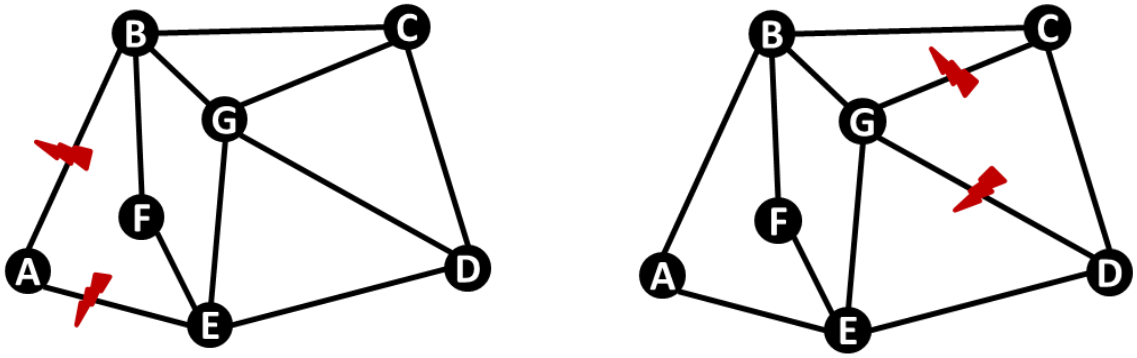


Figure 2.4. Dual span failure; (a) as spans AB and AE fail simultaneously, there is no more route connecting node A to nodes B and E. (b) As spans GC and GD fail simultaneously, there are still routes connecting node G to nodes C and D.

Another aspect of survivable network design is the problem of spare capacity allocation on the network. There have been several survivability mechanisms developed, each of which is based on a specific procedure and subject to specific constraints and limitations, place spare capacities on the network mainly with the objective to minimize the total cost of the sparing. The followings are the fundamental concepts behind some of the well-developed survivability mechanisms:

2.2.2.1 Automatic Protection Switching (APS)

The automatic protection switching (APS) mechanism provides traffic survivability based on two slightly different mechanisms: 1+1 APS and 1:1 APS [45]. In 1+1 APS mechanism, the traffic demand is being carried onto two disjoint routes simultaneously; one is the primary working route and the other is the backup route. Upon a failure happening on the primary route, the traffic on the backup route will be transferred to the

destination node. In 1:1 APS mechanism, the traffic demand is being routed over a primary route and there is a backup route on standby. Upon failure on the primary route, the traffic will be routed over the backup route to get to the destination node [4]. A schematic of the APS mechanism is depicted in Figure 2.5, where, for none of the APS mechanisms, the reserved backup routes and their allocated spare capacities were shared between various working traffics. Thus, the amount of required spare capacity is equal to the working traffic all around the network. However, there are survivability mechanisms developed that allow for sharing the spare capacity among various working traffic routes and thus have more efficiencies regarding the spare capacity allocation. The followings are such survivability mechanisms known as mesh network survivability mechanisms [46].

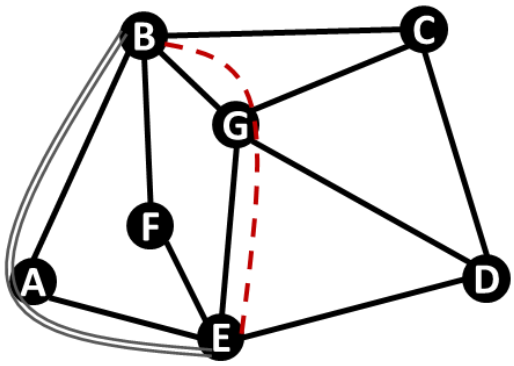


Figure 2.5. APS mechanism; the working route between nodes B and E (depicted by a double solid line) is protected by a backup route depicted by a red dashed line.

2.2.2.2 Preconfigured Cycles (p-Cycles)

p-Cycles are preconfigured cycles equipped with spare capacity to restore the traffic on failed spans either on the cycles or straddling spans of the cycle [4], [47]–[49]. A schematic of the p-cycle mechanism has been depicted in Figure 2.6 where the network includes 10 nodes and one p-cycle over the nodes BCDJI. The indicated P-cycle is able

to protect the spans on the cycle (e.g., BC, CD, DJ, JI, IB) and straddling spans (e.g., CI and DI) whose end nodes are incident on the cycle. For instance, if span BI fails, the affected traffic can be restored using the p-cycle over nodes B-C-D-J-I (Figure 2.6.b). On the other hand, if a straddling span such as ID fails, the affected traffic can be restored over both sides of the p-cycle (I-B-C-D and I-J-D). Figure 2.6.c depicts the restoration routes for the failure of span ID.

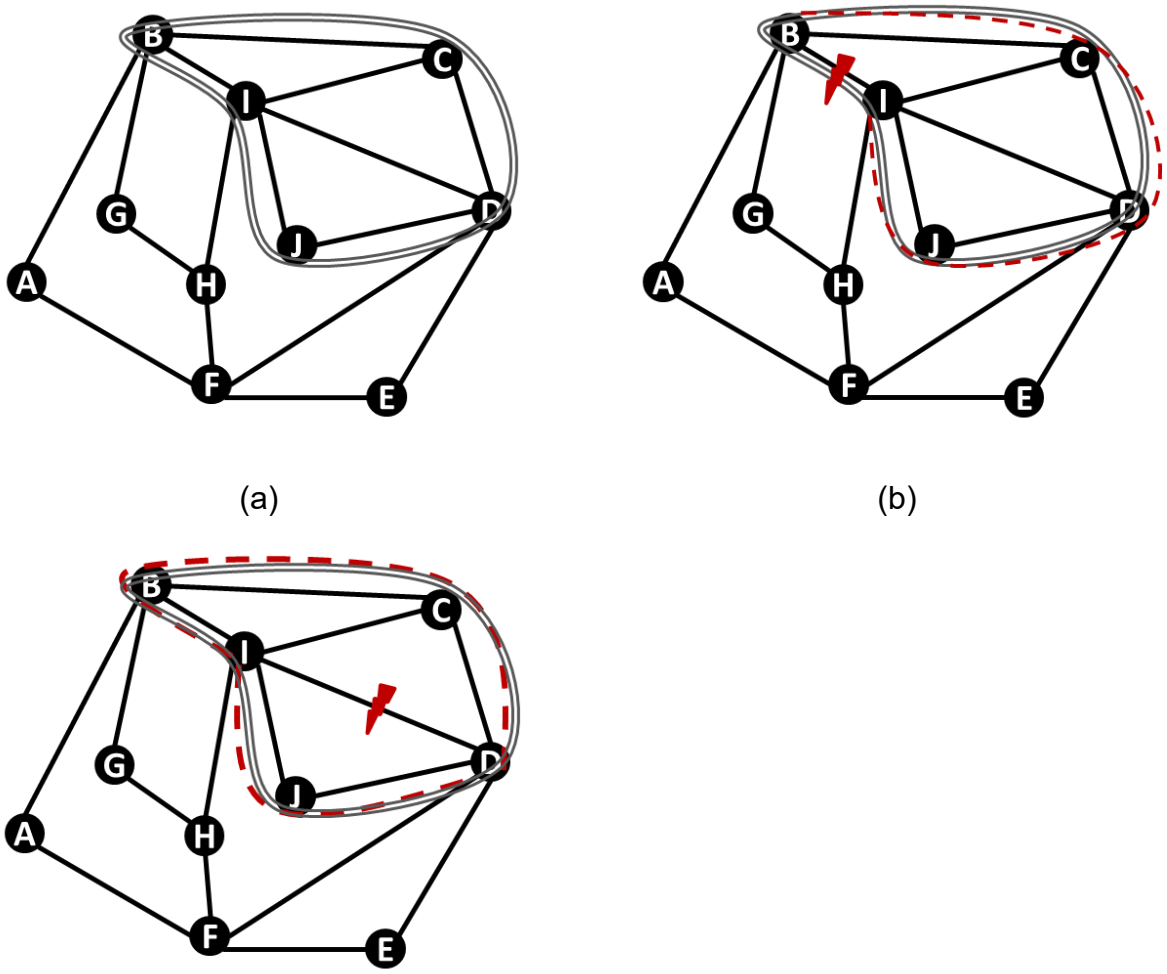


Figure 2.6. p-Cycle mechanism; (a) a p-cycle over the nodes B, C, D, J, and I have been depicted. The showed P-cycle can protect the spans on it along with the straddling spans that only their end-nodes are on the p-cycle, (b) upon failure of span BI along the p-cycle, the affected traffic can be restored through the red dashed part of the p-cycle, (c) upon failure of straddling span ID, the affected traffic can be restored through either side of the p-cycle (whether IBCD or IJD).

2.2.2.3 Span Restoration

Using a shared pool of allocated spare capacities all over the network, the failed working route will be restored between the end nodes of its failed span. In this manner, the network topology at least needs to be two-connected between the end-nodes of the failed span and should have spare capacity on the backup route/s equal to the working

traffic on the failed span [50]. A schematic of the span restoration mechanism has been depicted in Figure 2.7 where the span between node-pair OD has been failed. The span OD may be part of any route. Employing the span restoration mechanism, restoration routes between nodes O and D which have been depicted by dashed lines, restore the traffic on the failed span OD.

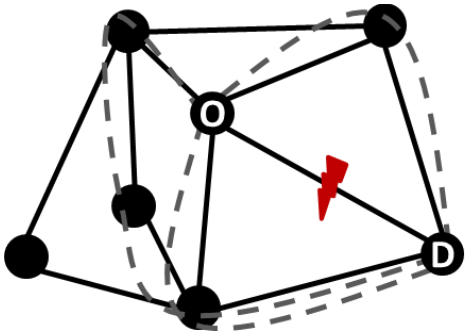


Figure 2.7. Span restoration mechanism; upon the failure of span OD, restoration routes were formed between nodes O and D and have been depicted by dashed lines.

2.2.2.4 Path Restoration

Based on the location of the failed span along the working route, the path restoration mechanism replaces the failed working route by employing one or more backup routes between the end-nodes of the primary working route [51]–[53]. In addition to the sharing ability of the spare capacities, a unique feature of the path restoration mechanism is performing the “stub-release” procedure in which the capacity on the intact sections of the affected working route will be released and made available to be used as the spare capacity for upcoming backup routes [10], [51]. Figure 2.8 illustrates the path restoration mechanism where the working route has been depicted by a dashed line between the node-pair BE. Considering the failure of span BC, by employing the path restoration mechanism, the capacities on spans CD and DE will be released for restoration

purposes. In this case, a backup route between node-pair BE has been illustrated by a solid-double line that uses the released spare capacity on span DE.

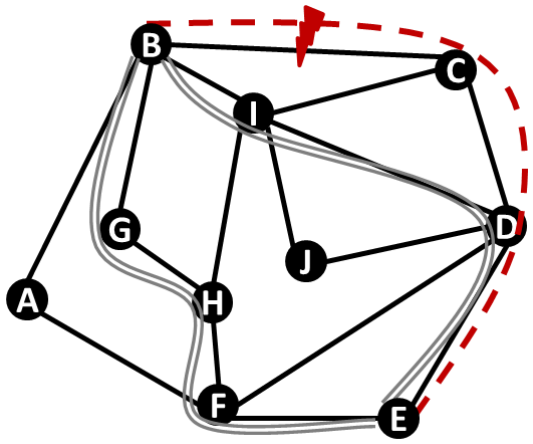


Figure 2.8. Path restoration mechanism; upon the failure of span BC as a part of the BCDE route, two backup routes were formed between nodes B and E and are depicted by solid-double lines.

2.2.2.5 Shared Backup Path Protection (SBPP)

Once there is a span failure along the working route, the traffic on the affected route will be rerouted over the backup route/s between the end nodes of the affected route. The location of the failed span along the working route will not affect the backup plan in the SBPP mechanism. Also, in SBPP the working routes that are disjoint and do not have any shared-risk spans can share the spare capacity on their backup routes [43], [54]–[56]. Figure 2.9 illustrates the SBPP mechanism on a network with 10 nodes, two working routes between node-pairs BE and ID respectively are represented by dashed lines. Employing the SBPP mechanism, as the two working routes do not share any span, their backup routes (represented by solid-double lines) can share spare capacity on the shared span JD on both backup routes.

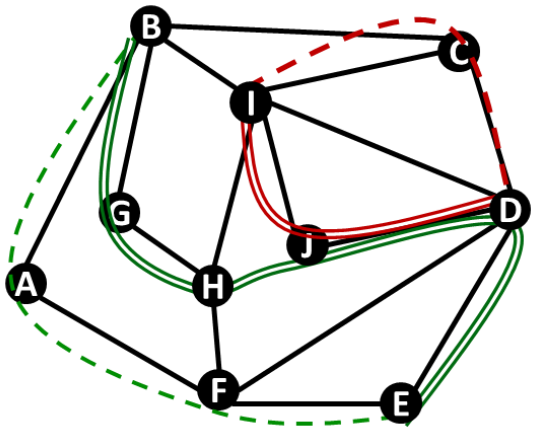


Figure 2.9. Shared Backup Path Protection mechanism; the backup routes (shown by solid lines) of the disjoint working routes (shown by dashed lines) share the spare capacity on the shared span DJ. The failures can happen anywhere along the working routes.

2.2.3 Network Availability

Considering a network as a whole system under study, based on the general definition of availability, network availability can be defined as the probability of the network being fully functional under its specified conditions at any random time. Mathematically, this probability can be defined as the ratio of the network's working-state duration to the network's total expected operating time [20] and [57]. In a detailed description, a fully-functioning state of a network can be defined as its ability to carry the required traffic demands over the specified working routes between its desired end nodes. In this context, the availability of a network will be dependent upon the availability of its constitutive components; nodes and spans. As mentioned earlier, there could be numerous sources of failure threatening the network components. In the event of such failures, the duration that takes to repair the failed components is referred to as the downtime of those components during which the components are unavailable. Therefore, upon a component failure, if the affected traffic is under delay because of

maintenance work, the network is not at a fully-functioning state and thus the network availability is less than 100% (unity) [10], [58].

Depending on the governing survivability scheme on the network and the failure scenarios under protection, a specific *service level agreement* (SLA) can be offered as a guaranteeing level of service quality by the service provider [20]. For instance, if a network is designed to be survivable against any single-span failure, the network's availability would be 100% in the event of any single-span failure, but if a dual span failure occurs, there might not be enough spare capacity allocated on the network to survive the dual span failure scenario. In such a case, some parts of the affected traffic may not be restored and thus will contribute to the unavailability of the network's service.

2.2.4 Solution Approaches

There are two general approaches for solving optimization problems: deterministic approaches and non-deterministic approaches. Based on the definition, the deterministic approaches are based on deterministic algorithms in which there is only one way of proceeding in every step of the algorithm [27]. In other words, by solving a problem more than once using a deterministic approach, we will find the exact solution that we had found the first time. On the contrary, there are non-deterministic approaches wherein every step of the solution approach, there might be decisions made based on a random approximation process that can result in different solutions every time a problem is being solved. Non-deterministic approaches can be further categorized as heuristic and metaheuristic approach. A heuristic approach is a problem-dependent approximation approach for solutions to a problem [27]. A metaheuristic is a

non-deterministic solution approach that can consist of one or more heuristics and approximation functions specifically designed for a problem [27].

2.2.4.1 Deterministic Approaches

In this thesis, we employed several ILP models as deterministic approaches whether as benchmarks or as proposed models for various network design problems. We used a mathematical modelling software package named AMPL to generate the computer-coded mathematical version of the ILPs [59]. The computer-coded ILP models were then solved using a mathematical programming solver named Gurobi [60].

2.2.4.2 Heuristics and Metaheuristics

Although the ILP models (upon feasibility) are guaranteed to find an optimal value for the decision variables, there are drawbacks associated with their performance in various conditions. As opposed to the deterministic approaches such as ILP, there is another category of mathematical modelling approaches named non-deterministic approaches such as *heuristics* and *metaheuristics*. Heuristics are problem-specific algorithmic approaches that are not guaranteed to find the exact optimal solution, but they are rather fast approaches that can find sub-optimal solutions to the problems [27], [61]. Metaheuristics are problem-independent approaches to search for the optimal solution to optimization problems [27]. There are various metaheuristic algorithms developed and most of them are nature-inspired, among them, tabu search (TS) –an algorithm based on local search, which avoids visiting a member of the search space more than once– [62], [63] and simulated annealing (SA) –an approximation-based algorithm inspired by the pattern from the heating process of metals– [64]. A subset of

metaheuristics is *evolutionary algorithms*. Evolutionary algorithms are nature-inspired optimization algorithms that are adopted from the process of evolution of populations in nature [65] and [27]. There are several well-established evolutionary algorithms developed for solving optimization problems in almost every branch of science, among them, particle swarm optimization (PSO) –an optimization approach where each solution in the search space is considered as a particle that has a position and velocity. The movement of a particle is related to its best position along with the best global position in the search space– [66], [67], ant colony optimization (ACO) – an optimization approach inspired by the ant's movement from their colony to a food source– [68], and genetic algorithm (GA) –an iterative search algorithm based on the concept of breeding and repopulating from a set of members– [69], [70].

2.2.4.3 Genetic Algorithm

The majority of contributions of this thesis have benefitted from metaheuristic algorithms and specifically novel GAs for the network design problems. GA is an iterative evolutionary algorithm formulated according to the theory of breeding mates (i.e., *parents*) and generating *offspring*. Through the breeding process, the offspring inherit some attributes and features from the parents. In this context, every parent and offspring is a standalone entity (i.e., *individual*) representing a solution (i.e., *chromosome*) to the problem and the sub-sections of a chromosome that contain the unique features and attributes, are the *genes* of the chromosome [70]. The initial set of parents can be generated randomly or using an approximation of the solution [61]. The generated set of initial parents generates a pool consisting of parents named the *initial*

population. A general representation of a GA chromosome, gene, and the population is depicted in Figure 2.10.

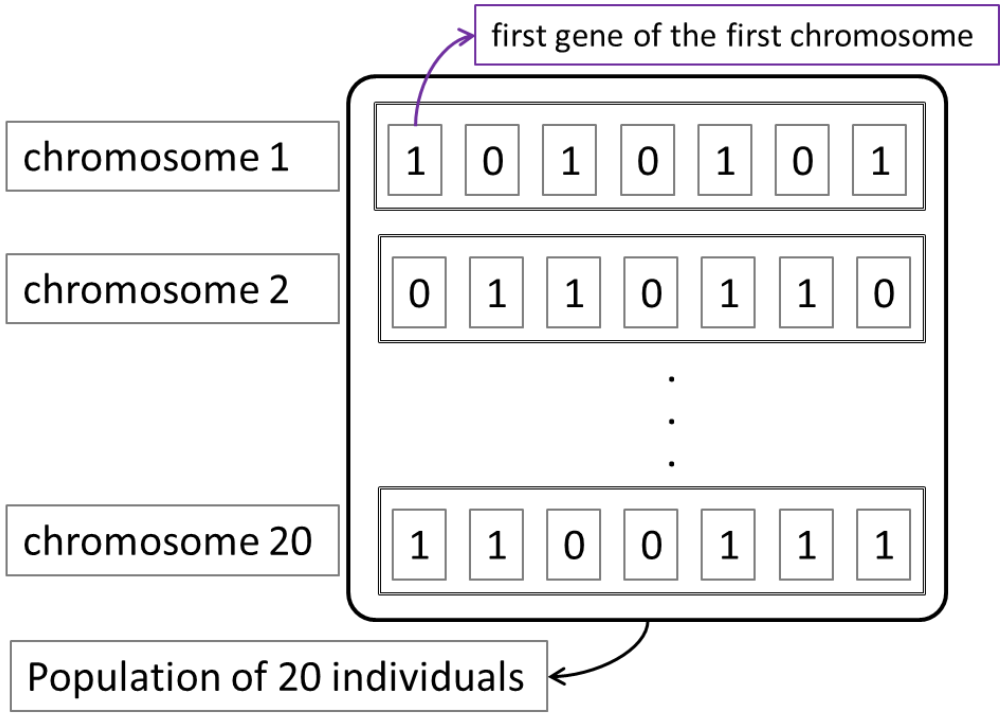


Figure 2.10. A representation of a GA population with 20 individuals, where every individual is represented by a chromosome with seven genes.

Once the initial population of individuals is generated, the *fitness* of every individual is determined. An individual's fitness is the objective function value of the chromosome that represents that individual. The members of the population are then sorted according to their fitness. If none of the population members did have the desired fitness, GA continues to generate a new population of individuals. To generate a new population, first, a number of the members of the current population should be selected as parents for the breeding process. The breeding process in GA is facilitated by two genetic operators named *crossover* and *mutation*. Crossover usually takes two individuals from the current population as parents and generates two offspring from

them. The generated offspring inherit some of the genes from each parent. Mutation usually takes one individual from the population and applies changes and alterations to its genes to create offspring. The fitness of the obtained offspring from crossover and mutation is determined and the offspring are appended to the current population of individuals. Then the population members are sorted according to their fitness value, once more. As the new offspring are added to the population, the size of the new population increases. In order to maintain the population size, the sorted population is truncated. Once a new population is obtained, if none of the members of the population have the desired fitness and other termination criteria such as runtime are not met, this process repeats, otherwise the GA stops. Algorithm 2 describes the general high-level pseudocode of a GA. Also, Figure 2.11 depicts the general process of the GA steps. A detailed description of the steps of a Genetic Algorithm has been provided in the following subsections.

Algorithm 2. High-level pseudocode of a Genetic Algorithm

Input:

The objective function,
Decision Variables,
Constitutive genes of the chromosome

Output:

Best chromosome

Genetic Algorithm:

1. Create an initial population
2. Calculate the objective function value of chromosomes
3. While termination criteria have not been met {
    4. Select parents for the breeding process
    5. Breed the parents and get new offspring
    6. Accumulate all of the current parents and offspring in the current population
    7. Evaluate the members of the current population based on their objective function value
    8. Replace the least valuable members of the population with the most valuable ones (sort the current population and truncate it to maintain the same size) }

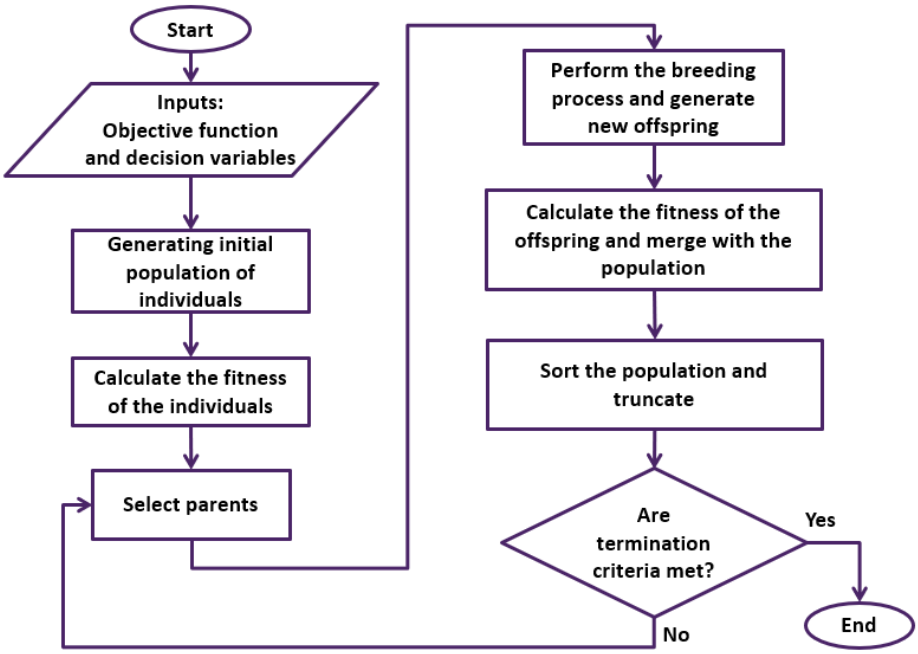


Figure 2.11. A flowchart of the general steps of a Genetic Algorithm [70]

2.2.4.3.1 Genetic Algorithm- Fitness

GA starts with an initial population which is a set of individuals. Every individual is represented by a chromosome that contains the necessary information to form a solution to the optimization problem. As mentioned earlier, in an optimization problem, we seek to maximize/minimize the value of an objective function. Thus, the measure for evaluating the fitness of every individual's chromosome can be considered as the chromosome's objective function value [70] and [71]. By calculating the objective function value of every chromosome in the initial population and sorting them from the fittest to the most unfit, we now have a sorted initial population where the individual associated with the first chromosome represents the best solution obtained so far.

2.2.4.3.2 Genetic Algorithm- Evolution

The breeding process is performed using a set of genetic operators. We have provided a detailed description of the genetic operators and how they facilitate the breeding process in the next section, but first let us elaborate more on the general evolution process and the way the chromosomes evolve in a GA.

The initial population members go under a set of breeding processes to generate a new set of chromosomes. After the breeding process, there are some newly-generated chromosomes added to the initial population. In every population, to maintain the size of the population, the parents might be replaced with the newly generated offspring. The replacement process happens based on the fitness of the offspring and parents as described in Section 2.1.4.3. After such replacements happen, the remainder of the offspring that were not used (i.e., they were not valuable) will be discarded. The process of eliminating the most unfit and keeping the fittest chromosomes in the population is called the “survival of the fittest” [70]. The set of the new parents that is a combination of parents from the initial population and offspring from the recent breeding process is called the “current population”. The members of the current population will go through the breeding process again to generate the next population. This repetitive process goes on until one or more termination criteria are met. Depending on the algorithm’s application and the available computational power, the termination criteria can vary. The termination criteria generally indicate when the algorithm reaches the desired solution that has an acceptable level of fitness [70]. The termination criteria can

be considered as a specific number of iterations, a specific process time, or a specific fitness value.

2.2.4.3.3 Genetic Algorithm- Feasibility of the Chromosomes

The chromosomes in a GA should have enough information to build a solution to the problem under study. The information embedded in a chromosome should satisfy all of the constraints of the problem to be considered a *feasible* solution. As an example, let us assume we have a function $Z = f(x, y)$ that we want to maximize subject to a set of constraints on the decision variables x and y such that $0 \leq x \leq 10$ and $0 \leq y \leq 20$. In this example, the fitness is the value of the function f for every pair of the decision variables x and y. The algebraic form of this problem is represented through Equations (3) to (5).

Maximize $$Z = f(x, y) \quad (3)$$

Subject to: $$0 \leq x \leq 10 \quad (4)$$

$$0 \leq y \leq 20 \quad (5)$$

A solution to this problem is two numeric values associated with x and y. As we mentioned earlier, a chromosome should contain sufficient information for describing a solution to the problem. Thus, herein, the chromosome includes two genes, each of which contains the numeric values of decision variables x and y. Depending on the values that every decision variable takes on, the resulting solution may differ. For instance, the values of $x = 5$ and $y = 20$ specify numeric values for both decision variables, and they represent a solution of $Z = f(5,20)$ to the problem.

Although any x and y values represent a solution to the problem, they might not satisfy all the constraints in the problem. For example, the values of $x = 5$ and $y = 22$ specify numeric values for both of the decision variables and they represent a solution of $Z = f(5,22)$ to the problem, but the value of y does not satisfy the constraint (5). To have a *feasible solution* to the problem, every decision variable's value should satisfy the governing constraints of the problem. If the information in a chromosome does not satisfy even one of the governing constraints in the problem, the chromosome is representing an *unfeasible* solution to the problem. We call such a solution an infeasible solution. Even if one of the decision variables' values in a solution, does not satisfy at least one of the governing constraints, the solution is infeasible. In this context, a feasible solution to the problem is a solution that its decision variables' values satisfy all of the constraints. Similarly, a feasible chromosome in a GA represents a set of genes that form a feasible solution. For the mentioned problem, if in a chromosome the value of x was not complying with the range $0 \leq x \leq 10$ or the value of y was outside of the range $0 \leq y \leq 20$, the chromosome is representing an infeasible solution to the problem. In the event of facing an infeasible solution, we can remove the infeasible solution from the population and continue with only the feasible ones, or we can conduct a *repair* process through which the infeasible solution becomes feasible. The repair process employs an operator that changes the value of the decision variable that is not complying with one or more of the constraints, to a value that complies with all of the constraints [72]. Therefore, the result of applying a repair process to an infeasible solution is a feasible solution.

2.2.4.3.4 Genetic Algorithm Operators

The effectiveness of the genetic operators depends on their ability to perform a robust search that benefits from both *exploration* and *exploitation*. Exploration can simply be interpreted as searching for new parts of the search space that during the search process so far, have not been explored yet [27]. Exploration can help the search process to avoid trapping in local minima. On the other hand, exploitation can be defined as the ability of the search process to find new solutions based on modifying the previously-found solutions [27]. If the genetic operators tend to exploit more than explore, the risk of trapping in local minima will be increased [27]. For example, let us consider the search space depicted in Figure 2.12 where every circle point represents a potential solution to the problem. Considering the points indicated with a star are the previously-found solutions from the visited areas of the search space. The visited areas of the search space can be considered as the left-hand side of the search space where there are some solutions found. On the contrary, the unvisited areas of the search space can be considered as the right-hand side of the search space where there is almost no solution found there yet. By employing genetic operators for breeding that will exploit more than explore, it will probably result in finding new solutions indicated by square marks within the visited areas of the search space. Thus, the probability of finding a solution from the unvisited areas in the search space is low. In this scenario, if the optimal solution is in the unvisited areas of the search space, the search algorithm will have a slight chance of finding it. Similarly, employing the genetic operators that tend to explore more than exploit, may make the search algorithm unable to find the global optimum [27].

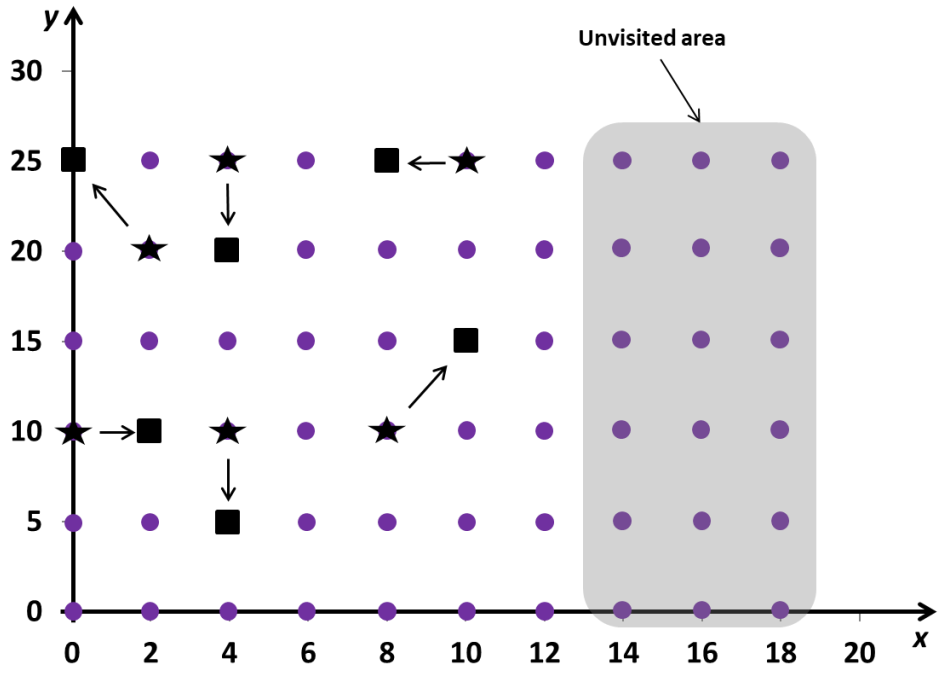


Figure 2.12. Demonstrating the effect of exploitation on finding the optimal solution in a search space; the circle points represent the potential solutions, the starred points are the previously found solutions, and the squared points are the newly-found solutions resulting from exploiting the search space.

As an example, to illustrate the effect of sole exploration, let us consider the search space depicted in Figure 2.13 where every circle point represents a potential solution to the problem. Considering the points indicated with a star are the previously-found solutions from the visited areas of the search space. By favouring exploration over exploitation in finding new solutions, the new solutions (indicated by square marks) from unvisited areas of the search space may be found. In this scenario, unless the optimal solution is exactly one of the newly-found solutions (which is a very optimistic chance), the chance of the search algorithm being able to find the optimal solution is very low. Even if the optimal solution lies close to any of the square solutions, only by conducting an exploration-based search, the chance of the search algorithm being able to find the optimal solution is low.

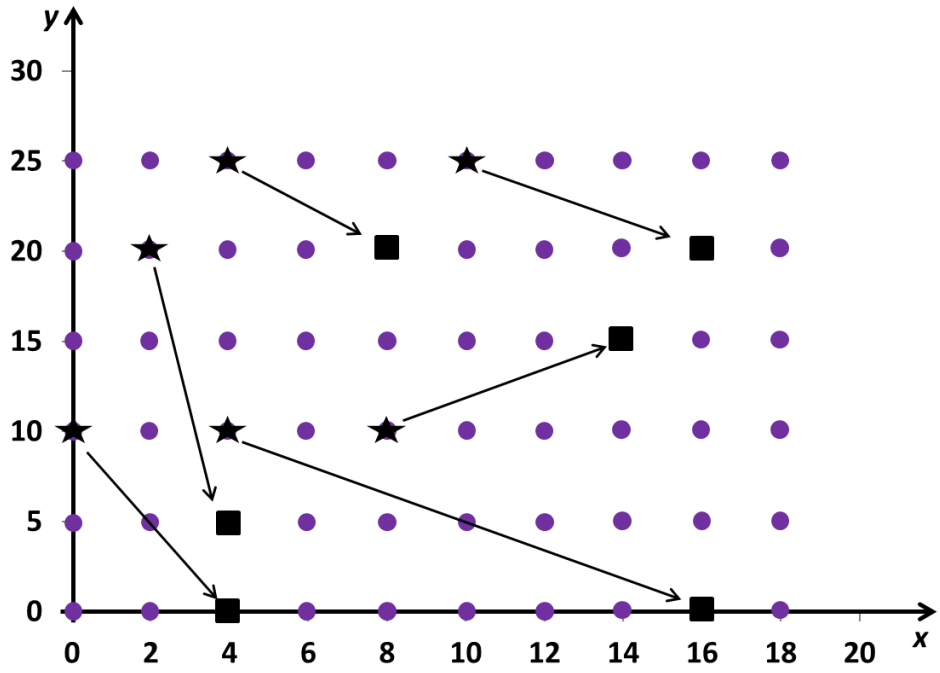


Figure 2.13. Demonstrating the effect of exploration on finding the optimal solution in a search space; the circle points represent the potential solutions, the starred points are the previously found solutions, and the squared points are the newly-found solutions resulting from exploring the search space.

The breeding process in a GA is being facilitated by two groups of operators: “crossover” and “mutation”. The concept of crossover is to breed two or more chromosome parents and generate some new offspring that inherited genes from the parents. The crossover operator normally contributes to the exploitation aspect of the searching process. On the other hand, the mutation operator has more randomness associated with it. The mutation operator applies changes to the genes of a chromosome to generate new offspring (i.e., a new solution). The changes that a mutation may apply range from swapping the genes where the order of genes is important, replacing genes, changing the information embedded in genes, or any other action that randomly changes the structure of the chromosome. Based on this definition,

the mutation operator normally contributes to the exploration ability of the searching process [27].

In this thesis, we designed various GAs for a set of network design problems. The detailed description of the chromosome structure, genetic operators, replacement process, and termination criteria for every GA that we designed, can be found in their corresponding chapters. The designed GAs are implemented and solved in Python [73]. Python is a powerful programming language with various libraries and modules for implementing and solving mathematical programming models in an efficient and timely fashion.

2.2.5 Background on Survivable Network Design[2]

Several works in the literature consider the basics of network topology design, routing, and survivability problems using various ILP models [74]–[79][3]. In addition, several works incorporate heuristics-based approaches in combination with ILPs that also consider the survivable network design problem [1] and [80]–[86]. Many of those ILP models or heuristic approaches can find an optimal or near-optimal solution to the specific problems they address, challenges often arise associated with their processing and/or solution time, especially for large-scale networks [33] and [81].

In the work presented in [33], the authors introduced a three-step ILP-based heuristic approach for the mesh-restorable network topology problem, determining the optimal topology, working traffic and restoration routing, and working and spare capacity allocation. This near-optimal approach provided solutions with minimal optimality gaps

[2] Parts of this section are taken and/or adapted from our journal paper in IEEE Access [129].
[3] Parts of this section are taken and/or adapted from our journal paper in IEEE Access [129].

in most of the cases and is much faster than addressing the complete problem mesh topology routing and sparing (MTRS) problem in a single step. There are additional works that explored the routing and survivability problem for networks with known topologies. In [87], the authors presented a new heuristic routing method (based on space reduction) as an improvement to the *Mixed Integer Linear Programming* (MILP) runtime for large-scale optical network design. That approach is useful for moderate-sized networks with topologies that are known in advance. Another example of work that deals with known topologies is the work of [88], which developed a heuristic method for transporting demands on a shared-mesh protection network in the event of multiple failures. The work discussed in [89] introduced a two-level evolutionary method for a survivable network topology design problem. The results obtained showed 3% to 7% improvements in runtime relative to those discussed in [33], but only networks with less than 30 nodes were tested.

In the network reconfiguration problem, traffic demands may change frequently [90]–[92]. In [93], the authors presented an arc-chain formulation for the routing problem, given the network topology for medium-size networks. They employed column generation and a generalized upper bounding structure to improve ILP efficiency.

Employing evolutionary approaches has been an effective way to increase the runtime efficiency of network design problems. Heuristics and evolutionary algorithms have been proposed as alternative solution approaches for the FCR problem [94], [95], and [96]. Among the evolutionary algorithms, GA has been one of the most popular solution algorithms. In [97] a GA-based approach for a digital-data-network design was presented, with the results compared to a Tabu Search approach. In [98], the authors

presented a GA-based algorithm for capacity and routing assignment on a network with known topology and subject to single failures. GA's have also been used as an interim step of hybrid solution approaches, for example, to limit the search space for an ILP's initial solution. In [99], a hybrid ILP-GA approach was used to find a set of preselected cycles and tested on a network design problem with 200 nodes. The authors in [100] presented an approach to designing large-scale optical networks using a custom heuristic and a GA to find a near-optimal design of an optical network. The GA in that work finds near-optimal solutions based on capacity limits of each span; the GA used a simple single-point crossover and a binary flip mutation, and because infeasible solutions could arise, it required a repair operator. The work in [101] presented a GA-based algorithm for minimizing the network cost while protecting it against single failures for a network with a predefined topology. Their results showed smaller costs and longer runtimes in comparison with Tabu Search.

Depending on the network's applications, the restoration mechanism is usually designed only for restoring single failure scenarios while the rate of occurring dual failure scenarios is becoming higher in large-scale networks [102]. Considering dual-failure scenarios as a probable failure pattern in large-scale networks, the dual-failure restorability in network design problems has been studied by several researchers. Some of the mentioned research works assumed a fixed topology for the network. In another word, they considered the topology of the network to be given and cannot be modified. The authors in [103] studied the dual-failure survivability of small networks with a fixed topology using ILP models for the Dedicated Path Protection mechanism. The authors in [104] and [105] presented ILP models that solve the partial dual failure

restoration using the span restoration mechanism in networks with fixed topologies. On the other hand, there are research works that considered the network topology as a variable part of the design problem. For instance, there are research works that incorporated the topology design with the restoration problem to optimize the complete network design process. The authors in [106] and [107] in addition to the spare capacity allocation, considered the network topology as a variable for restoration purposes. They introduced ILP models capable of solving partial dual failure restoration considering an upper bound for the cost, for small networks with topology augmentation.

The authors in [108], employed ILP models to study full restorability against single failures and partial restorability against dual failure scenarios. The authors in [109], studied the dual failure restorability of 3-connected networks with fixed topologies. They analyzed various hop path distances and considered minimum hop and minimum length constraints for finding the routes. The authors in [110] presented ILP models for maximizing the dual failure restorability of single failure restorable networks by re-planning the pre-planned and optimized backup routes, considering various priorities for every failed span in every dual failure scenario.

The survivable network design and spare capacity allocation problems are NP-hard [1] and [21]. Therefore exact solution approaches like ILP cannot solve the large-scale versions of these problems efficiently (i.e., in logical time using the existing computers). Thus we focused on the approaches that can solve large-scale networks in a reasonable processing time using the existing infrastructure. Heuristic algorithms are one of the best tools for this purpose.

The literature includes employing GA for network design and spare capacity allocation considering various restoration mechanisms [111] and [112]. The authors in [113], presented a hybrid methodology based on GA for designing the survivable network topology from a complete graph. The authors in [114], employed GA to determine the optimal pre-planned backup routes for the network's restoration by categorizing the population based on various features. The authors in [115], presented a GA-based greedy algorithm for joint working and backup route selection problems, given fixed network topology. In [116], the authors presented a GA-based approach to select candidate cycles and determine the optimal set of pre-planned cycles (p-cycles) for network restoration. Moreover, the authors in [117], presented a three-step heuristic-based approach for optimizing the spare capacity allocation. In their presented approach, the heuristic algorithm finds sub-optimal working and backup routes, then using a GA, the required traffic is distributed over the found routes. In that work, the authors were able to conveniently incorporate complex non-linear constraints regarding the shared-risk link groups into their algorithm which is one of the GA's fortes. Depending on the problem specifications designing a proper set of genetic operators, all of the search space can be scanned for the optimal solution. The authors in [118], presented a GA able to solve the cost minimization of single failure restorable network design problem. They used the complete graph to randomly select the working and backup routes for the single failure restoration considering a hop limit for the routes. The authors in [119], presented a GA for designing minimum cost survivable networks. They studied path-dedicated protection for single-span failure scenarios. They used single-

point and uniform crossover and random binary mutation operators. Thus, their solution is guaranteed to be feasible in the initial population only.

The network availability analysis has been studied by several researchers. There are exact and heuristic approaches presented in the literature that analyze various measures of network availability considering different restoration mechanisms and failure scenarios. The authors in [120] presented ILP models for improving the availability levels in p-cycle protected networks by rerouting the traffics and protecting the working routes by two protection routes. The authors in [20] investigated the dual failure availability of the single failure span restorable and 1+1 APS protected networks. They concluded that span restorable networks have higher availability levels compared to 1+1 APS-protected ones. The authors in [43], [121], and [122] studied the dual failure availability of SBPP networks with fixed topologies. They presented ILP models for maximizing the network dual failure availability and studied the relationship between the allocated spare capacity and network availability. The authors in [123], analyzed the dual failure availability of SBPP-protected networks with fixed topologies. They presented an algorithm for calculating the dual failure availability of the network based on the dual failure availability of every working route. They also found an interesting relationship between the average nodal degree of the network and the network availability. Based on their results, with the increase of the network nodal degree from 3 to 5, the network availability first increases, but then levels out and decreases. The authors in [124] studied single failure restorable networks with fixed topologies. They presented ILP-based models for calculating the dual failure path availability in the networks that are designed for full single failure restoration. The authors in [125]

presented an ILP-based algorithm for maximizing the dual failure availability of single failure path restorable networks. By employing a set of predefined working and backup routes, they minimized the number of non-restored working capacities on the working routes.

There are research works that specifically investigated the dual failure availability of single failure span restorable networks [44] and [126]. These works employed a set of pre-enumerated backup routes for the restoration of spans and presented an ILP model for maximizing the network dual failure availability by allocating spare capacities to the selective backup routes. Later, the authors in [127] and [128] studied the minimum design cost of dual failure span restorable meta-mesh networks with fixed topologies. They also presented an ILP model based on pre-enumerated backup routes for maximizing the network availability subject to a given limit of total spare capacity investment.

Chapter 3 : Large-Scale Fixed Charge plus Routing Network Design Problem using a Novel Genetic Algorithm

The main content of this chapter has been adapted/edited from our work published in the journal IEEE Access [129].

In this chapter, we present a novel approach that addresses the problem of large-scale network topology design and routing. There are research works that used exact methodologies based on ILP models to develop potential solutions for this problem. However, this problem is computationally NP-hard, thus solving it is hugely demanding on computational power for large-scale networks, and in many cases, it is not even possible to generate a solution with a reasonable optimality gap. This paper presents a hybrid algorithm based on the Genetic Algorithm with efficiently designed genetic operators. This algorithm aims to design the topology of large-scale networks and generate a routing configuration for a set of predefined traffic demands on the networks while keeping the total cost of design and routing at a minimum. The results have been compared to an exact ILP model as a benchmark for validation purposes. The comparisons showed that the proposed algorithm significantly outperforms the ILP solution in all of the large-scale network configurations that were used as case studies.

3.1 Introduction

Optical transport networks have become widespread, and they have a pervasive role in most typical interactions of modern society. When designing and building out such networks, designers often assume static and known traffic demands and a known and static topology. However, over time, changes in traffic demands, or perhaps maintenance activities or upgrades in some parts of the networks, may drive inevitable

topological changes that need to be implemented. To apply these changes, the network's topology needs to be restructured or redesigned.

As mentioned earlier in Chapter 2, the network design problem, in its general form, may include (holistically or as several sub-problems) topology design, demand routing and working capacity placement, and survivability routing and spare capacity placement. The topology design problem seeks the optimal topology structure for the networks considering various factors such as cost or certain constraints on the connections among the nodes. The traffic demand routing problem aims to route all of the demands on the network by finding the optimal path for each demand to minimize the total cost of the routing. These two sub-problems can be modelled together in a single ILP as the *fixed charge plus routing* (FCR) problem [32]–[35]. Given a fixed set of nodes, the objective of the FCR problem is to minimize the total cost of the network topology design, demand routing, and working capacity placement. The FCR problem is described in detail in Section 3.2.

In the event of some failure, the survivability routing and spare capacity placement problem provides alternative routing on which the network can reroute the affected traffic, and ensures sufficient spare capacity to accommodate that routing. There are several survivability mechanisms developed for the restoration process of networks including *automatic protection switching* (APS), *survivable rings*, *span restoration*, *path restoration*, *p-cycles*, and *shared backup path protection*, to name a few [10].

We've identified several shortcomings that commonly arise in the existing literature, and developed an Improved Genetic Algorithm (IGA) in a manner that seeks to overcome the following common weaknesses:

(1) **Weakness**: Generating a high-merit initial population is time-consuming, due to the stochastic nature of the process and subsequent repair actions that are often required to transform infeasible solutions into feasible solutions.

Response: In our work herein, we designed our GA to ensure that the solutions generated for the initial population are connected network topologies; hence, they are inherently feasible.

(2) **Weakness**: Genetic operators (typically crossover and mutation) are often inefficient if great care is not given to designing them specifically for the structure of the chromosome, and frequently result in offspring that need to be repaired.

Response: In our work herein, we designed crossover and mutation functions that seek to satisfy the problem's constraints, thus avoiding the need for a computationally expensive repair process.

(3) **Weakness**: The complete parameter space of the control parameters (e.g., crossover and mutation rates) is vast, and although the precise settings for those parameters can have a very significant effect on the convergence rate of the GA and the subsequent processing time, many GAs in the literature does not specify any particular efforts to tune those parameters.

Response: In our work herein, we have carefully analyzed and tuned these parameters for the GA-based approach.

(4) **Weakness**: The validation of the results of the non-deterministic approaches is an important step for analyzing their effectiveness. Many GA's in the literature were only compared against non-deterministic approaches.

Response: In our work herein, we have validated our results using an ILP model as a benchmark, which can find the optimal solution for cases where the complexity of the problem allows an ILP model to be used.

3.2 The Fixed Charge plus Routing (FCR) Problem

The network design problem that we are addressing in this work is the FCR problem [33], where we seek to jointly optimize the topology and working routing (and subsequent working capacity allocation) to serve all traffic demands. Although the FCR problem doesn't originate with us, we will present the full ILP herein, for completion.

In the FCR problem, by optimizing the topology we are aiming to establish a number of the required spans with the least cost possible and by the working routing optimization, we are seeking the cheapest working routes between the end nodes of every traffic demand for transferring the traffic. Two cost components define the total cost of a designed network: (1) the fixed cost (F_{ij}) that is the cost of establishing the span between nodes i and j and (2) the capacity cost (C_{ij}) that is the cost of transferring one unit of the traffic on the span between nodes i and j. For simplicity, hereafter we refer to the span between nodes i and j as span ij. Based on the defined cost components, the total cost of a network with a set of established spans and routed traffic demands over those spans is calculated using Equation (6).

$$Cost = \sum_{ij \in A} (F_{ij} \times \delta_{ij} + C_{ij} \times w_{ij}) \tag{6}$$

Where A is the set of all the possible spans connecting the nodes, δ_{ij} is a binary decision variable that has the value of 1 if span ij exists in the topology and 0 otherwise. w_{ij} is the total number of traffic units on span ij. The goal is to minimize the total cost mentioned above subject to the traffic demand constraints, as described in Equations (7) through (11), [33]. The ILP model of the FCR problem which aims to minimize the value of Equation (6) contains Equations (6) to (11).

$$\sum_{\forall nj \in A} w_{nj}^{r} = d^{r} \qquad \forall r \in D, n = O[r] \tag{7}$$

$$\sum_{\forall jn \in A} w_{jn}^{r} = d^{r} \qquad \forall r \in D, n = T[r] \tag{8}$$

$$\sum_{\forall in \in A} w_{in}^{r} - \sum_{\forall nj \in A} w_{nj}^{r} = 0 \quad \forall r \in D, \forall n \notin \{O[r],\, T[r]\} \tag{9}$$

$$w_{ij} = \sum_{\forall r \in D} w_{ij}^{r} \qquad \forall ij \in A \tag{10}$$

$$w_{ij} \leq w_{\infty}.\delta_{ij}; \qquad \delta_{ij} \in \{0,1\}; \qquad w_{ij}\ integer \quad \forall ij \in A | i < j \tag{11}$$

Here w_{ij}^{r} represents the number of working traffic flows for demand *r* on span *ij*, and d^{r}represents the number of flow units of demand r. Also, $O[r]$ and $T[r]$ represent the origin and destination nodes of demand r, respectively. Equations (7) and (8) make sure that the total flow from the origin node to the destination node is precisely equal to the demand units. Equation set (9) makes sure the demand units pass entirely through the intermediate nodes. Equation set (10) places enough working capacity on each span to carry the flows of overlapping demands. Moreover, Equation set (11) makes our

decision variables maintain their structure throughout the solution, whether they are binary or integer.

The FCR problem is NP-hard [33]. Although the problem can be solved efficiently for relatively small networks by an appropriate commercial ILP solver (e.g., Gurobi™, [60]) it is generally too computationally complex for large-scale networks [33]. Solving the FCR problem on a suite of test case networks ranging in size from 40 nodes to 150 nodes, we observe the runtime behaviour documented in Table 3.2.

3.3 Proposed Methodology

In solving the Fixed Charge plus Routing problem, the size of the set of candidate spans for the network's topology is considerable and selecting an optimal subset of those candidate spans is a massive combinatorial problem; as stated above, FCR is an NP-hard problem. Herein, we employ GA to solve the Fixed Charge plus Routing problem for large-scale network design.

3.3.1 Genetic Algorithm Building Blocks

The building blocks of a Genetic Algorithm have been introduced in detail in Chapter 2. In this Chapter, we only provide an example of the chromosome's structure. The properties of an individual are represented by a *chromosome*, which consists of a set of *genes*. In general, a chromosome is a string of data that fully encodes the properties of an individual solution to the problem. For the FCR problem, the chromosomes used would typically include genes that represent whether the various eligible spans are present in the solution. As an example, the network shown in Figure 3.1 (where dark

lines represent spans that are present in the solution, gray-coloured thin spans are not present) could be represented by the chromosome shown.

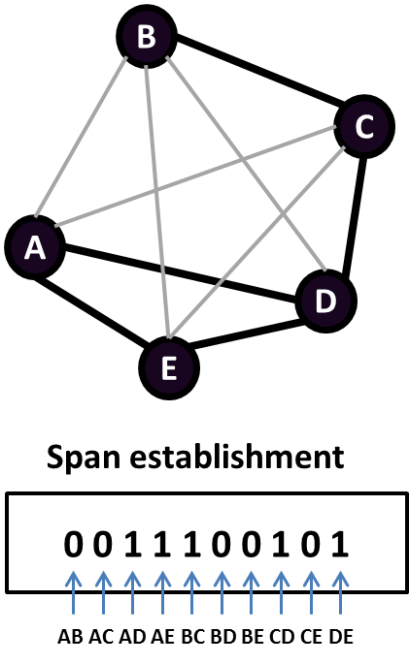


Figure 3.1. A GA chromosome representation of a network, including two genes as the established spans and their working capacities.

A general schematic of a chromosome and a population is depicted in Figures 3.2 and 3.3, respectively. The individuals in the initial population can be evaluated using their fitness values. For GA to evolve the chromosomes and generate future populations with better fitness, the breeding process should be utilized. The breeding process in GA is facilitated by a set of genetic operators and results in the generation of a new set of chromosomes which will be parts of the next population.

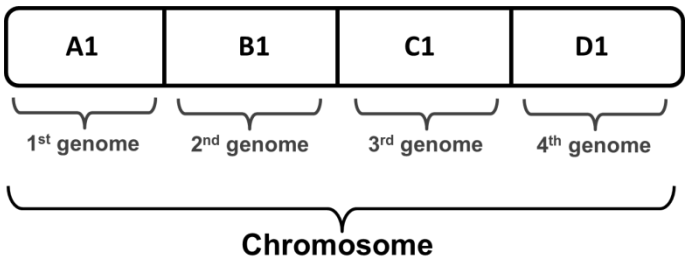


Figure 3.2. The general structure of an individual chromosome with its four constituent genes that are named A1, B1, C1, and D1

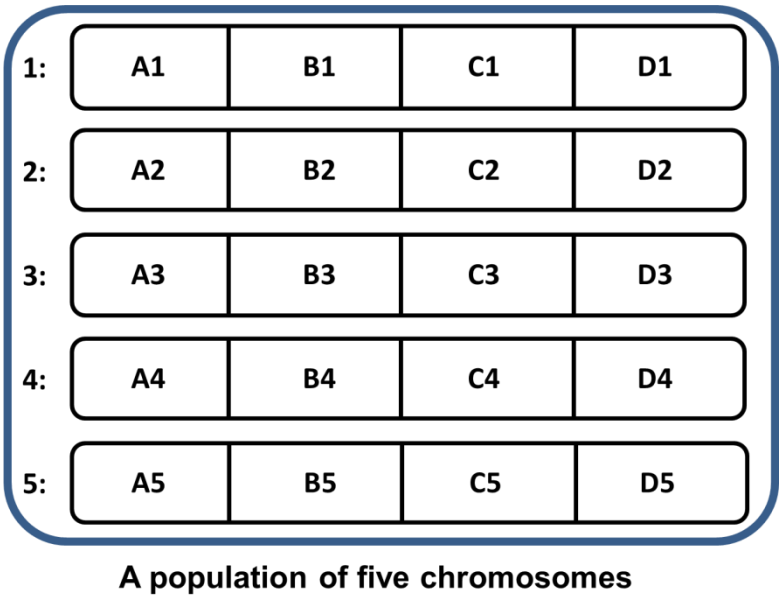


Figure 3.3. The general structure of a population that includes five chromosomes.

Figure 3.4 shows a crossover operation applied on two chromosomes (chromosomes 5 and 3 from the population from Figure 3.3) by swapping two genes between them. The resulting offspring inherited two genes from each parent. Figure 3.5 demonstrates a general mutation process on a selected chromosome. In this mutation, the second and fourth genes of the chromosome are exposed to some alterations, and as a result, their old structure as B2 and D2 has been replaced by B'2 and D'2, respectively.

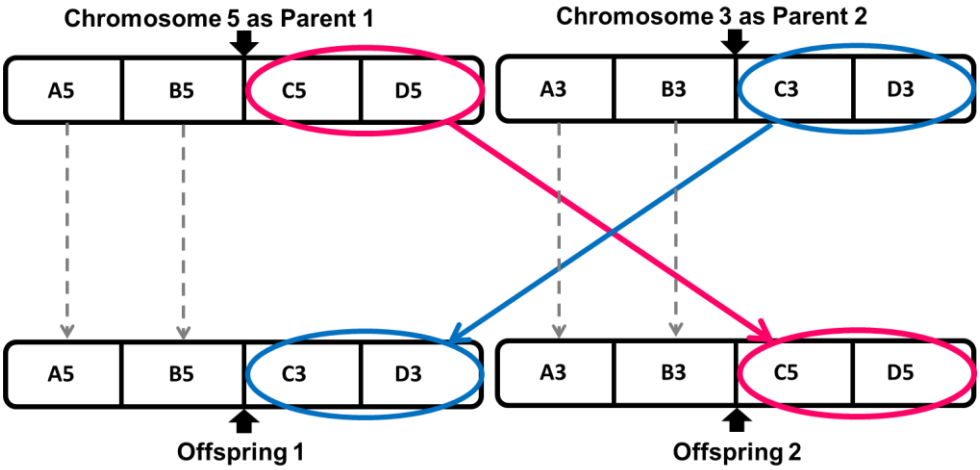


Figure 3.4. The general procedure of a basic crossover between two individual's chromosomes

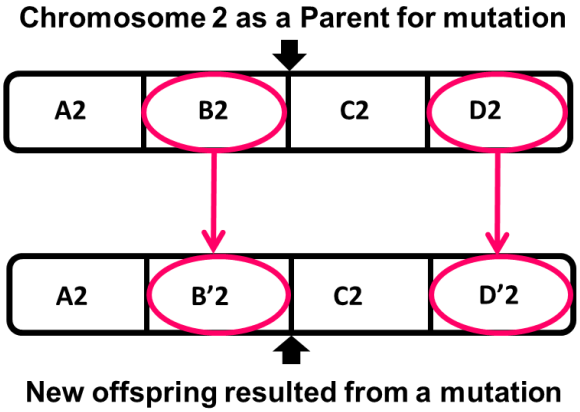


Figure 3.5. The general procedure of a fundamental mutation on two genes of an individual's chromosome

3.3.2 Improved GA Optimization Approach

In this chapter, we developed a GA-based optimization approach that consists of two well-designed GAs. The first GA is referred to as the *Initial-Population-Generator*, or *IPG-GA*, and is based upon the concept of the *Travelling Salesman Problem* (TSP) [37]. In TSP, the goal is to find the shortest cycle that passes through all of the nodes once and only once, i.e., the shortest *Hamiltonian cycle* [130]. Taking inspiration from the TSP, IPG-GA generates a set of Hamiltonian cycles as feasible but sub-optimal network

topologies. The generated network topologies are considered as the set of initial feasible solutions to the network topology design and routing problem. The structure of the IPG-GA chromosome is depicted in Figure 3.6. Here, the genes store the order of the selected nodes, which consequently represent the spans that form a Hamiltonian cycle as the network topology. Based on the installation cost of every selected span, the total topology cost of the network is calculated as the fitness function value of the individual. The details of the IPG-GA are discussed below, in Section 3.3.2.1.

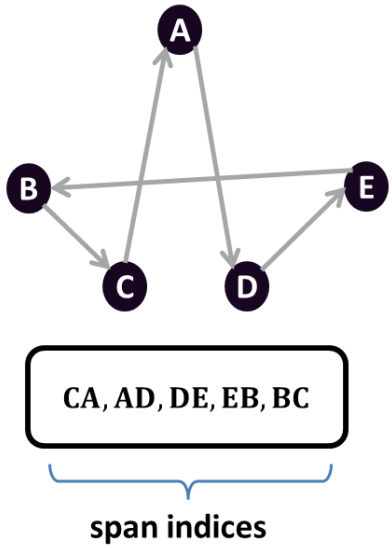


Figure 3.6. The structure of the chromosomes in IPG-GA demonstrates a genetically coded Hamiltonian cycle as the network topology.

Once a sub-optimal solution for the IPG-GA is achieved, a routing function is called to route all of the demands on the current network. The second GA, which we call the *Master-GA*, is responsible for simultaneously improving the topology of the network and routing the demands. The structure of the chromosome for the Master-GA is depicted in Figure 3.7. The genes encode the spans selected in the solution as binary values (1 if the span is selected, and 0 otherwise). A routing function routes all of the demands and

assigns the related working capacities on the spans encoded in the genes, and determines the total working capacity that arises on each span. As with the IPG-GA, the fitness function is the total cost of the individual, though that cost now also includes capacity costs. The details of the Master-GA are discussed below, in Section 3.3.2.2.

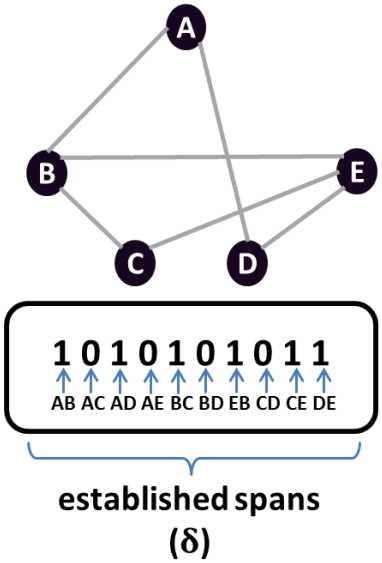


Figure 3.7. Structure of the chromosomes in Master-GA that demonstrates the complete topology of the network and assigned working capacities to the established spans in the network.

The high-level steps of the IGA algorithm, including both IPG-GA and Master-GA, are presented in Algorithm 3.

The proposed IGA algorithm includes two genetic algorithms: IPG-GA and Master-GA. The IPG-GA generates a set of sub-optimal Hamiltonian cycles as bi-connected network topologies. In other words, IPG-GA provides the Master-GA with a set of bi-connected network topologies that are of relatively good fitness. However, as the cyclic topology of networks does not necessarily represent the optimum network topology in terms of the network topology and routing cost, we implemented the Master-

GA to evolve the network topologies further. The detailed steps of the proposed IGA algorithm and its operators have been described in the following sections.

Algorithm 3. Proposed IGA Pseudocode including the IPG-GA and Master-GA

1. Initialize the population
2. Evaluate the members using the objective function
3. While not meeting the termination criteria:
    4. Select crossover parents
    5. Obtain the resulted offspring
    6. Evaluate the offspring
    7. Select mutation parents
    8. Obtain the resulted offspring
    9. Append the offspring to the current population
    10. Sort the population
    11. Truncate the population (maintain the same population size)

IPG-GA

12. Route the demands on the last population of solutions
13. While not meeting the termination criteria (reaching 20 iterations):
    14. Select crossover parents
    15. Apply the designed crossovers and obtain the offspring;
    16. Evaluate the offspring
    17. Select mutation parents
    18. Apply the designed mutations and obtain the offspring;
    19. Evaluate the offspring
    20. Append the offspring to the current population
    21. Sort the population;
    22. Truncate the population to maintain the same number of members

Master-GA

3.3.2.1 IPG-GA

As mentioned above, the well-known TSP seeks to find the shortest path in a network by visiting all the nodes once and only once; in network design and routing problems, the shortest path is often the lowest cost one. As we seek to minimize the total cost of

network design, the optimal solution would be the cheapest network topology. In the discussion above, we introduced the IPG-GA, which is responsible for generating a set of sub-optimal Hamiltonian cycles where optimality refers to the cost of the cycles. The IPG-GA starts with developing a set of Hamiltonian cycles among the nodes by generating 60 random ordered sets of all nodes resulting in 60 cycles in each generation. To generate new members for the next generation from the current one, IPG-GA benefits from two genetic operators: crossover and mutation. In each generation, 65% of the population undergoes mutation, and 65% undergoes crossover. The IPG-GA proceeds for 60 iterations, and as it proceeds iteration by iteration, the total cost of the cycles in the upcoming generations either will be the same as the current generation or will be less. Thus, the sub-optimal set of Hamiltonian cycles will be achieved in the last generation of the IPG-GA. The set of developed sub-optimal cycles represents the topology of the networks. However, the traffic demands have not been routed over the network, and thus, the working capacities have not been assigned to the networks' spans yet. For every cycle in the final generation of IPG-GA, we used an assignment algorithm that routes all of the traffic demands between their origin and destination nodes. For every traffic demand whose end nodes are directly connected by a single span in the network, a number of working traffic units equal to the traffic demand are assigned to that span. The other traffic demands' end nodes are on the cycle, but they are not directly connected by a single span in the network. In other words, these end nodes partition the cycle into two incomplete cycles, either of which can be considered a route for the demand. The traffic should be routed over the spans on either side of the cycle. To decide over which side of the cycle a demand should be

routed, we calculate the total capacity cost of traffic routing on the spans on either side and choose the one with the least total cost. Once for every cycle from the last generation of the IPG-GA, all of the traffic demands have been routed, we obtained an initial population to be fed into the Master-GA for further improvement in the topology design and routing cost. The pseudocode of IPG-GA has been illustrated in lines 3-12 of Algorithm 3.

3.3.2.1.1 IPG-GA Crossover

In IPG-GA, the chromosome structure is based on a random order of all of the nodes. Every two consecutive nodes represent a span. The set of all of the spans associated with the set of the random order of the nodes is stored as the genes of the chromosome. We employed two crossover functions to be applied to 65% of the current population in each iteration. First, we selected crossover parents using uniform random selection. Based on our pre-assessment analysis, both two-point and single-point crossovers generated diversity in the population. However, the two-point crossover was more effective compared to the single-point crossover. Thus, for every pair of selected parents, we applied either a single-point crossover with 0.25 probability or a two-point crossover with 0.75 probability.

In single-point crossover (depicted in Figure 3.8(a)), a nodal index was selected using a uniform random process to be the crossover point, where we partition the chromosomes of both parents into two parts. Two new offspring are generated by combining the first part of one parent with the second part of the other parent and vice versa.

Similarly, in a two-point crossover (depicted in Figure 3.8(b)), we select two crossover points (using the same approach as for single-point crossover) and partition the chromosomes of both parents into three parts. Two new offspring are subsequently generated by exchanging the middle part of one parent with the middle part of the other, and vice versa. In some cases, the crossover process creates an infeasible solution, where some nodes appear twice in a chromosome, and other nodes are absent. A repair mechanism follows the crossover to check for such incidents and swaps duplicate nodes with absent nodes. A schematic of this process is shown in Figure 3.8(b). As an example, let us consider the two parents represented in Figure 3.8(a), assuming the partitioning point is after the fifth member. The common members between the second part of the second parent and the first part of the first parent are 5 and 2. Similarly, the common members between the second part of the first parent and the first part of the second parent are 1 and 4. Considering the found duplicate members in the offspring, the first time members 5 and 2 appear in the first offspring, they are replaced by 1 and 4, respectively. Similarly, the first time that the members 4 and 1 appear in the second offspring, they are replaced by 2 and 5, respectively.

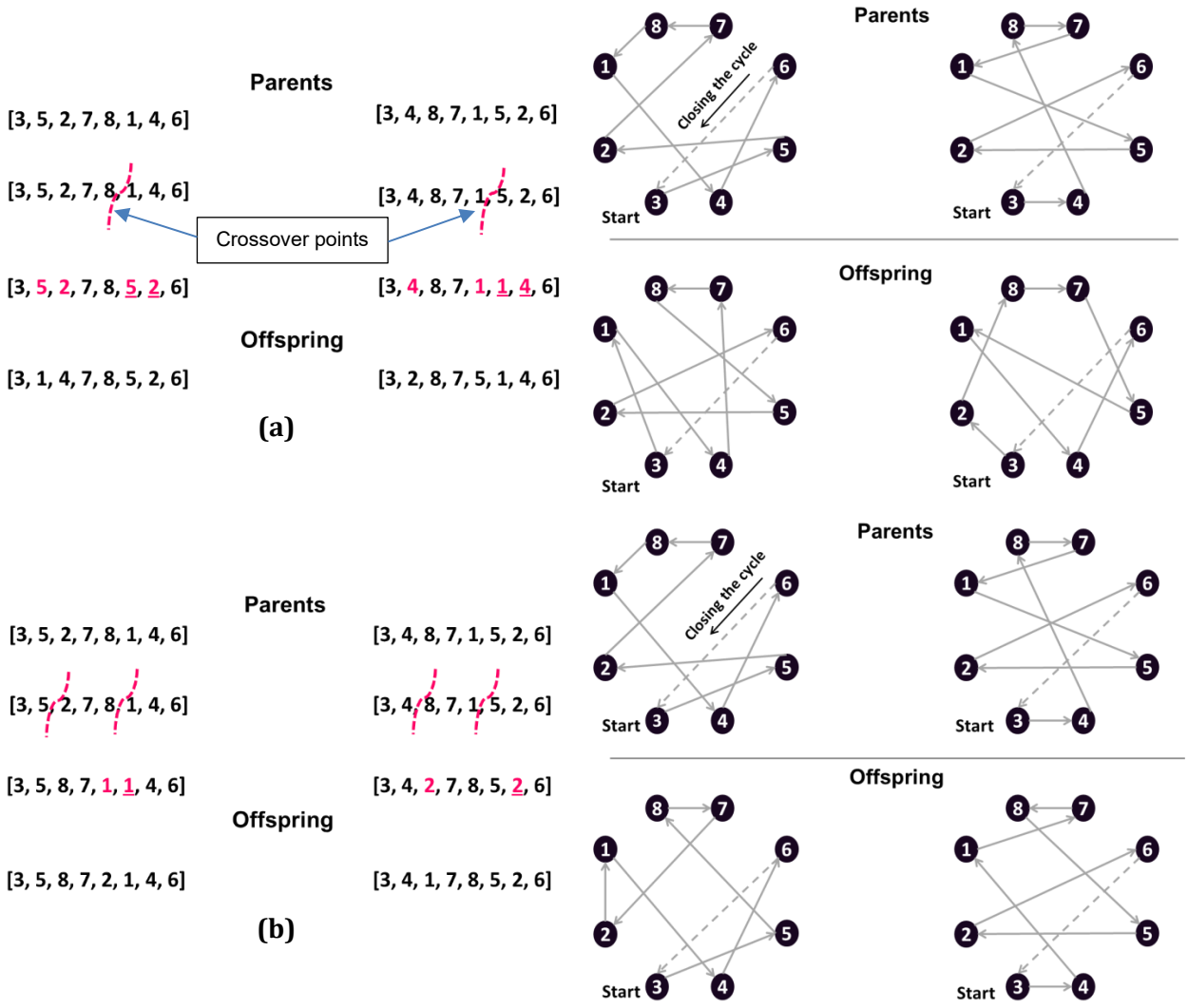


Figure 3.8. IPG-GA crossover operators the crossover points have been indicated by dashed red curve line; (a) Single-point, (b) two-point crossover.

3.3.2.1.2 IPG-GA Mutation

We utilized three different mutation functions: (1) two genes chosen by a uniform random process are swapped, (2) two genes are chosen by a uniform random process, and the entire sequence of genes between them (inclusive) is reversed, and (3) one gene chosen by a uniform random process is removed and inserted in a new location chosen by a uniform random process. Parent selection was made using the same approach described above for crossover. For each parent selected for mutation, we use

the first function with 0.25 probability, the second with 0.50 probability, and the third with 0.25 probability. The three mutation functions are illustrated in Figure 3.9(a) through Figure 3.9(c).

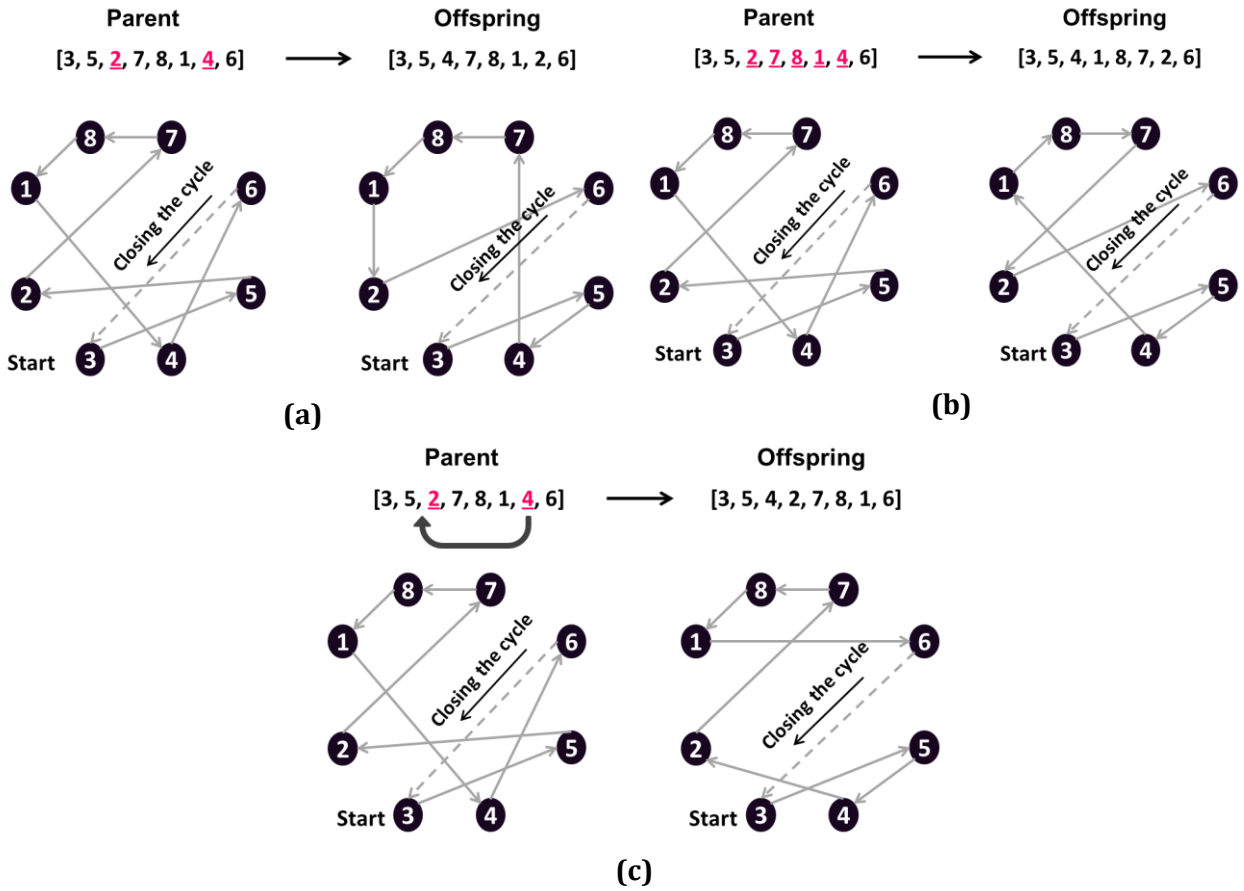


Figure 3.9. IPG-GA mutation operators; (a) swapping elements, (b) reversing a part of the chromosome, (c) moving an element from its position to a new position in the chromosome.

3.3.2.2 Master GA

The output of IPG-GA (i.e., its last generation) is a set of high fitness Hamiltonian cycles. Each of these cycles represents a feasible network topology that would satisfy the basic connectivity constraints of the FCR problem. However, these cycles will not necessarily include the optimal topology. Thus, the Master-GA is employed to improve this set of topologies and converge to a near-optimal solution. We chose the top 65% of

the chromosomes from the last iteration of the IPG-GA to be considered as the initial population for the Master-GA. The pseudocode of Master-GA is shown in lines 13-22 of Algorithm 3.

3.3.2.2.1 Master-GA-Assignment Function

In Master-GA, first, the topology of the network evolves using the genetic operators. Then using the *assignment* function, all of the demands are routed over the network's topology, and the required working capacities are allocated on the spans. We designed an improved Dijkstra's routing and assignment function to route the traffic demands and assign working capacities to spans based on the proposed routing operator in [100]. Given a set of traffic demands to be routed between their origin and destination nodes and the cost associated with routing the demands over every span in a network, Dijkstra's algorithm [41] finds the minimum-cost route for every demand between its origin and destination nodes. The *assignment* function employs an improved Dijkstra's algorithm to find the minimum-cost routes for every demand on the network. The cost of every route is determined based on the fixed cost of span establishment for every span along the route, plus the cost of transferring the traffic units on the spans along the route. Therefore, the cost matrix is comprised of both the span establishment and per-unit traffic routing costs, and once the demand is routed on the network, the fixed cost of the spans that form the route will be set to zero for the remaining demands; for already established spans, the cost matrix in Dijkstra's algorithm will only include the per-unit cost of routing the traffic.

In addition to the dynamic cost matrix for routing, the order of routing the demands have also been adjusted for routing. Demands have three components: 1)

origin node, 2) destination node, and 3) the units of the traffic demand (d^r). Here, they are sorted in descending order of their demand unit values; demand relations with the most demand units (i.e., with maximum d^r) are routed before those with fewer demand units.

3.3.2.2.2 Master-GA-Crossover

We employed two crossover functions to be applied to 55% of the current population in every iteration. The percentage of 55% has been selected based on a comprehensive set of analyses that showed us the best crossover rate is between 50% and 60%. A detailed description of our analysis has been provided in Section 3.3.2.3. For a crossover operation, we needed two parents to be selected from the population. For every parent to be selected, we performed a tournament selection with a tournament size of 10. According to our pre-assessment analysis, tournament selection resulted in better objective function values compared to random selection. In tournament selection, the best solution among a set of randomly selected solutions will be selected for crossover and mutation, and having both the randomness and elitism mechanisms can lead to a better result as opposed to only random selection. We repeated this process until 55% of the population was selected as crossover parents. For every pair of selected parents, we applied *master-crossover1* with 0.1 probability or *master-crossover2* with 0.9 probability. The master-crossover1 function takes two members of the population as parents and randomly removes 20% of the spans from both of them using uniform random selection. For each removed span, one span from the set of existing spans in the second parent was appended to the first parent such that the newly-appended span has one node in common with the removed span from the first

parent. The procedure of the master-crossover1 function has been illustrated in Figure 3.10. This crossover generates new offspring using new combinations within the already installed spans. Thus, it increases the diversity of the population within the visited areas of the search space. In other words, it promotes the exploitation ability of the search algorithm.

The master-crossover2 function takes two selected parents from the population and forms a set of spans that are absent in both of the parents (i.e., the set of all of the eligible spans except those in at least one of the parents). Then, using uniform random selection, master-crossover2 removes a random percentage of the spans between 50% and 90% of the parent's spans individually. We selected this range (50 and 90) based on our pre-assessment analysis, during which we found span removal percentages greater than 50 and less than 90 resulted in higher convergence rates. For each parent, from the set of absent spans, the master-crossover2 selects 10% of spans to be appended to that parent using uniform random selection. An illustration of master-crossover2 for one attempt of span removal and span appending process has been shown in Figure 3.11. In this example, the span between nodes 5 and 6 (i.e., span 56) and span 25 have been removed from parents 1 and 2, respectively. Then, from the set of absent spans, one span has been selected using uniform random selection to be appended to each offspring. As this crossover generates new offspring using absent spans (spans that were not installed in the selected parents yet), it increases the diversity of the population by exploring new areas of the search space.

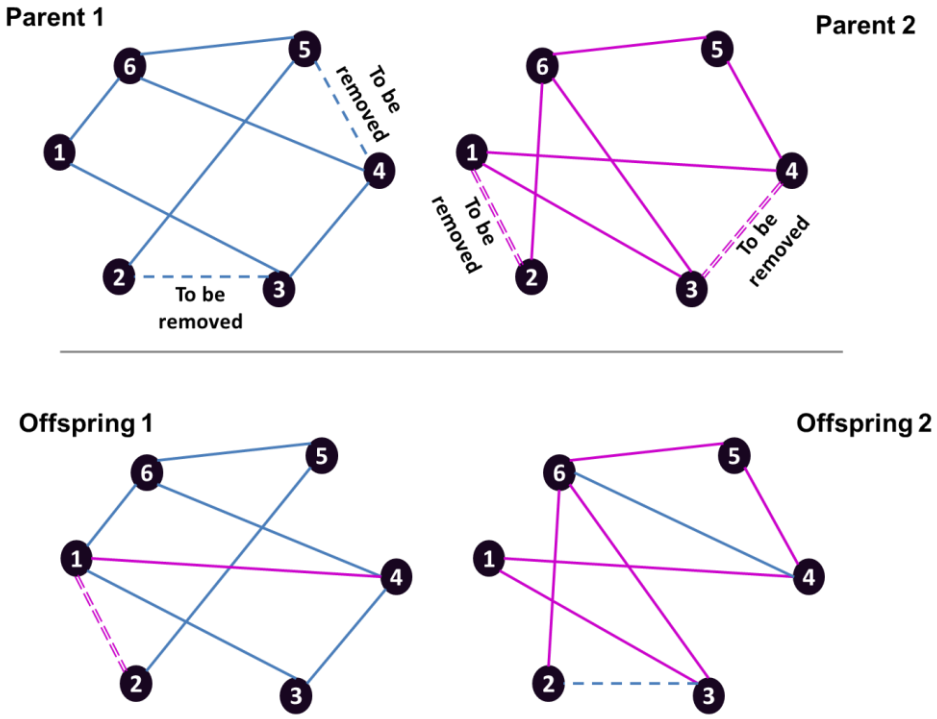


Figure 3.10. Master-GA, master-crossover1 function; Dashed spans in the parents are to be removed while the dashed spans in the offspring are inherited from the parents.

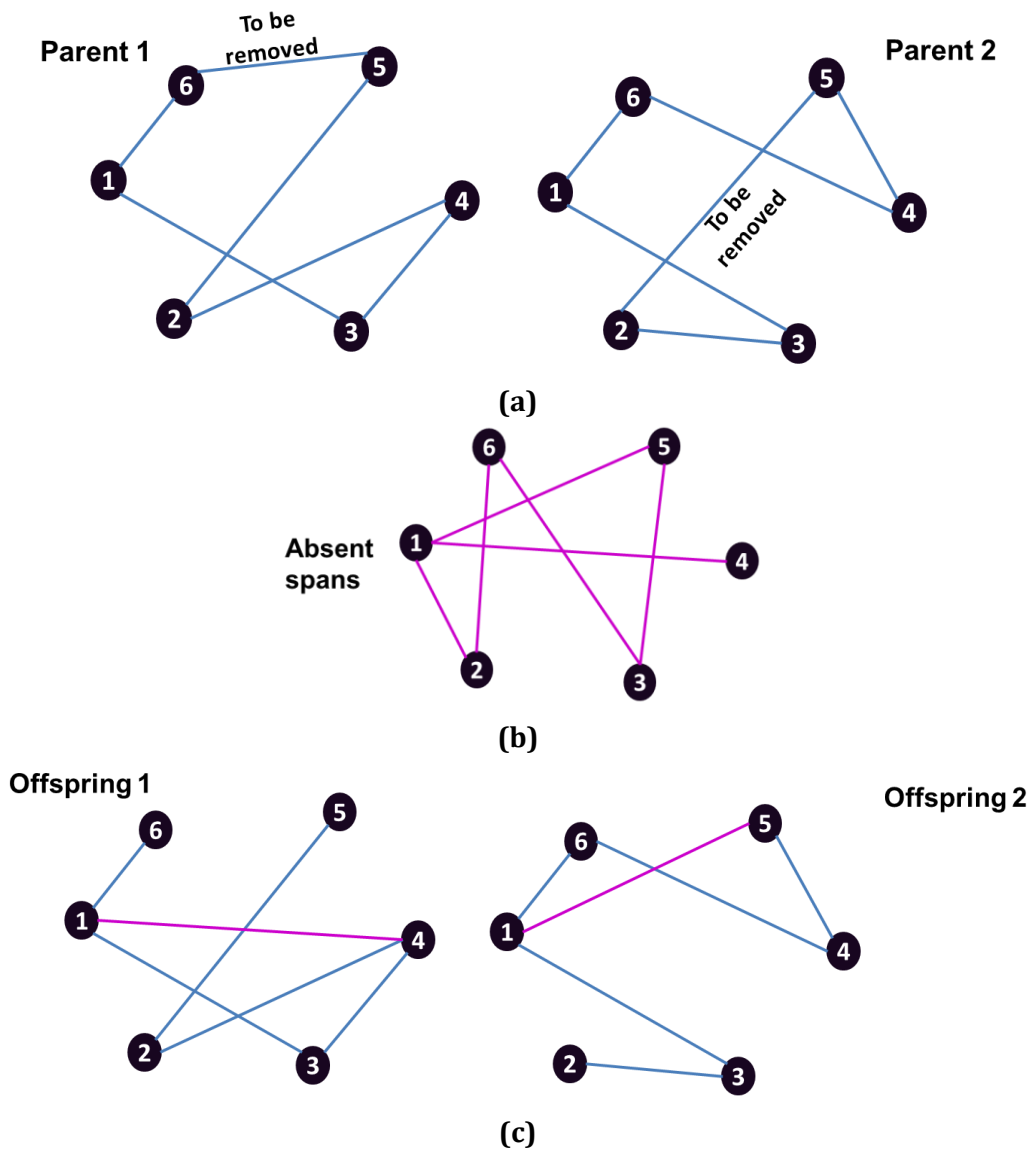


Figure 3.11. Master-GA, master-crossover2 function; (a) two parents with indicated spans to be removed, (b) the set of absent spans in both of the parents, and (c) the generated offspring.

3.3.2.2.2.1 Master-GA-Mutation

We utilized four different mutation functions:

(1) The most expensive span in the selected parent was replaced with another span with the least cost such that it has one end node in common with the replaced one. From the ascending ordered set of spans (i.e., sorted from the minimum to maximum

cost), the first span that has an end node in common with the removed span will be added to the network.

(2) Using uniform random selection, 40% of the spans in the network were selected. For every selected span, if the nodal degree of each end node of the span was greater than 2, that span was removed.

(3) From the set of existing spans in the network, a randomly selected percentage of the spans between 20% and 50% of the spans were removed using a uniform random selection process.

(4) From the set of absent spans in a selected parent, 20% of them were selected using uniform random selection and appended to the offspring.

Parent selection was done using the same approach described above for crossover. For each parent selected for mutation, we use the first function with 0.2 probability, the second with 0.3 probability, the third with 0.25 probability, and the fourth with 0.25 probability. Based on our pre-assessment analysis, we found the effects of the four mutation functions on the convergence rate of the algorithm almost the same with observing a small improvement from the second mutation function compared to the first one. The first two mutation functions are illustrated in Figures 3.12(a) and (b).

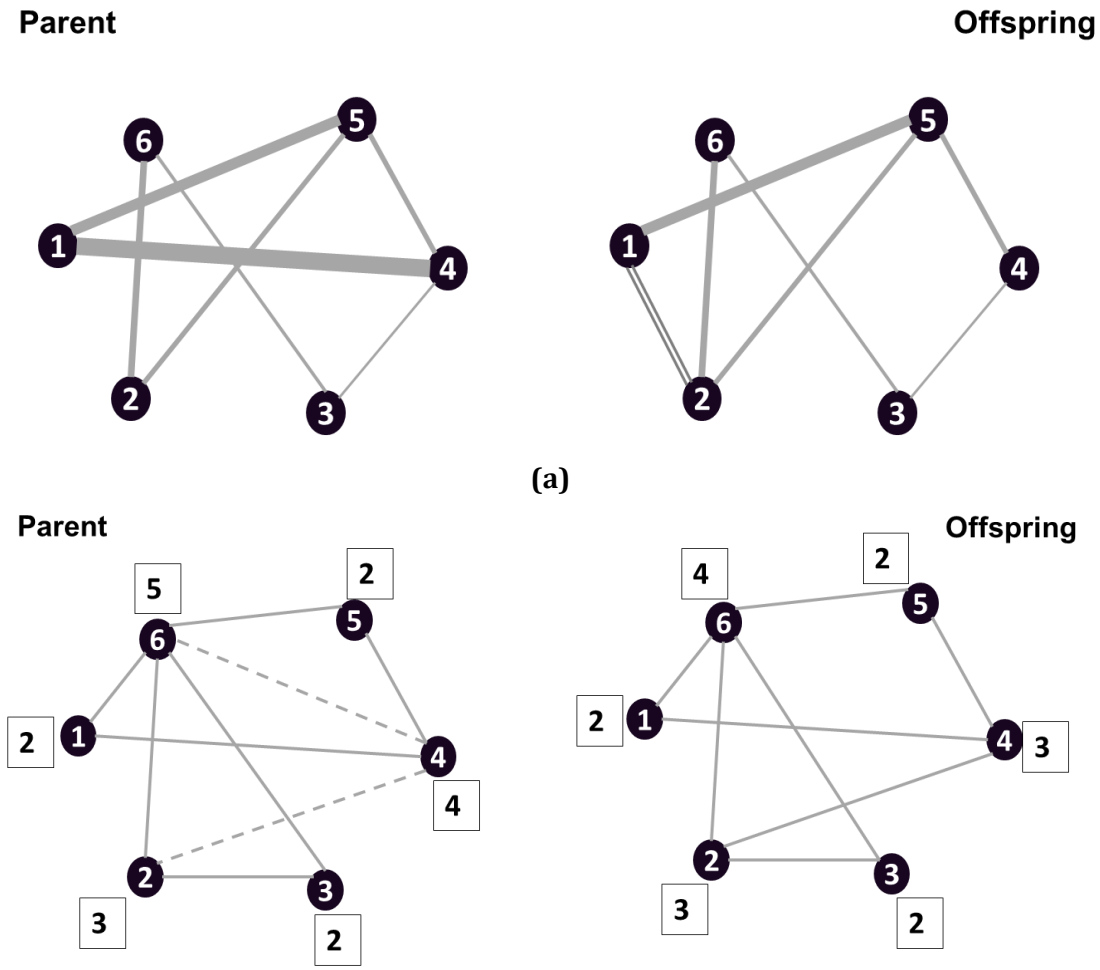


Figure 3.12. Master-GA mutations; (a) Cost-based mutation; the double-lined span is the newly installed span in the offspring, (b) Nodal degree-based mutation; the total nodal degree has been indicated next to every node, and the dashed spans are the ones with the highest nodal-degree end-nodes.

Although the mutation functions increase the exploration ability of the algorithm, there is a chance they would generate partitioned network topologies and hence, infeasible solutions. As most of the designed genetic operators try to keep the offspring feasible, the small chance of generating infeasibilities is associated with the randomness of selections. To avoid an unnecessary computational cost, the infeasible chromosomes did not go through the fitness evaluation process and were removed from the population. Moreover, as we initiated the Master-GA using a fully feasible population

from IPG-GA, in the worst-case scenario, even if all of the offspring are infeasible, we would still have a sub-optimal feasible solution to our problem.

3.3.2.3 Control parameters

There is a set of input parameters that directly control the efficiency of the algorithm. These parameters directly affect the processing time and the convergence rate of the algorithm. Setting up these parameters below a threshold can weaken their effects on the efficiency of the algorithm for converging toward the optimum solution while setting them up above that threshold can cause elongated processing times, which degrades the computational efficiency of the algorithm.

We performed a comprehensive set of analyses for tuning these control parameters for the presented IPG-GA and Master-GA separately. The parameters have been analyzed over 1440 separate runs (i.e., 20 runs per scenario) for a 20-node network. We analyzed a variety of crossover and mutation rates between 0.3 and 0.8. To select the best crossover and mutation rates, there must be a trade-off between the fitness value improvement and runtime increase. Our pre-assessment analysis showed that for every 10% increase in crossover and mutation rates of the IPG-GA, the average runtime increased by 2.6% and 2.1%, respectively. For example, by increasing the crossover rate of the IPG-GA from 0.6 to 0.7 when the mutation rate was fixed at 0.5, the runtime increased by 2.2%. Also, for every 10% increase in crossover and mutation rates of the Master-GA, the average runtime increased by 8.6% and 13.9%, respectively. Thus, we have to select the crossover and mutation rates that result in the best fitness value improvement while avoiding elongated runtimes.

The average fitness values for every combination of crossover and mutation rates of the IPG-GA are depicted in Figures 3.13 and 3.14. For fixed values of mutation rate, by increasing the crossover rate of the IPG-GA from 0.5 to 0.6, the average fitness value improves by 1.5%, while by increasing the crossover rate beyond 0.6, according to Figure 3.13, the fitness value improvements almost plateau (less than 0.5%). Moreover, for fixed values of crossover rate, by increasing the mutation rate of the IPG-GA from 0.5 to 0.6, the average fitness value improves by 0.3%, while by increasing the mutation rate above this value, the average fitness value improvement is less than 0.2%. Thus, based on the trade-off between the average fitness value improvement and runtime increase, the best threshold for the crossover and mutation rates in the IPG-GA was found between 0.6 and 0.7.

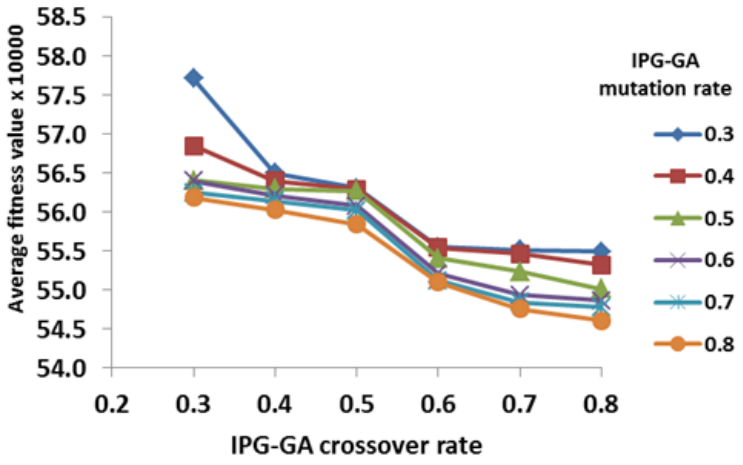


Figure 3.13. Average fitness value improvement with respect to various crossover rates of the IPG-GA, while the mutation rates are fixed.

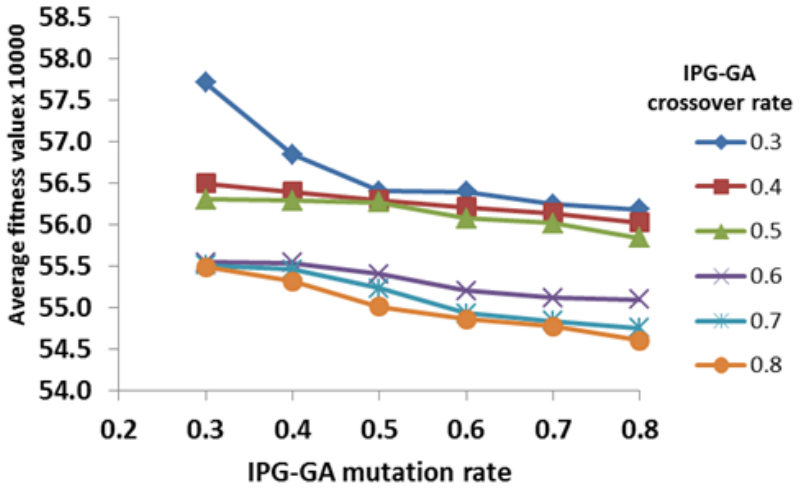


Figure 3.14. Average fitness value improvement with respect to various mutation rates of the IPG-GA, while the crossover rates are fixed

Similarly, the average fitness values for every combination of crossover and mutation rates of the Master-GA are depicted in Figures 3.15 and 3.16. For fixed values of the mutation rate, by increasing the crossover rate of the Master-GA from 0.4 to 0.5, the average fitness value improves by 0.2%, while with every 10% increase in the crossover rate beyond 0.5, the average fitness value improvement is less than 0.1%. Moreover, for fixed values of the crossover rate, by increasing the mutation rate of the Master-GA from 0.5 to 0.6, the average fitness value improves by 1.8%, while by increasing the mutation rate to higher values, the average fitness value improvement is less than 0.3%. Thus, for the Master-GA, the best threshold for the crossover rate was found between 0.5 and 06, while the best threshold for the mutation rate was found between 0.6 and 0.7.

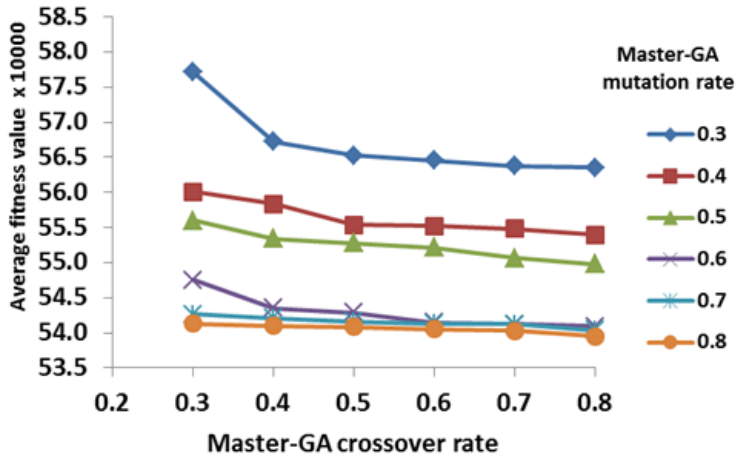


Figure 3.15. Average fitness value improvement with respect to various crossover rates of the Master-GA, while the mutation rates are fixed.

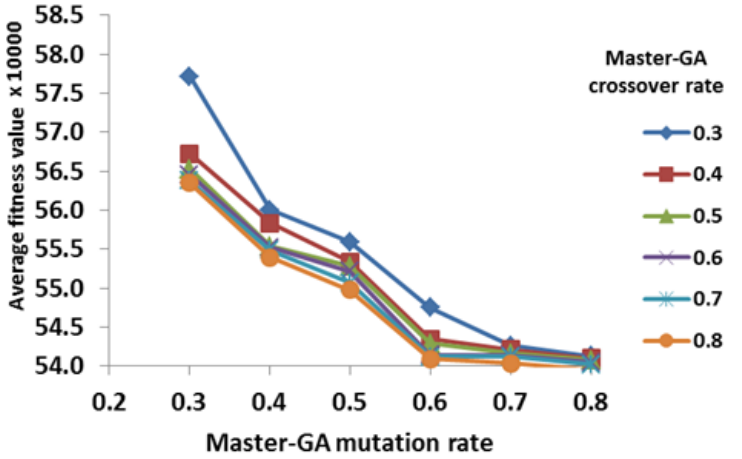


Figure 3.16. Average fitness value improvement with respect to various mutation rates of the Master-GA, while the crossover rates are fixed.

3.4 Results and discussion

3.4.1 Experimental setup

We now compare the results from the proposed IGA and those from the benchmark ILP model described by Equations (6) through (11) for the network topology design and routing problem. We used a Microsoft Windows Server 2012 R2 Standard x64-based PC, Intel(R) Xeon(R) CPU E5-2650 v3 running at 2.30 GHz with 128 GB RAM to solve the IGA in python 3.7 [73] and the ILP models using Gurobi 8.1.0 [60].

In this study, we employed 18 different test case networks ranging in size from 40 nodes to 150 nodes. Following some sources in the literature, we refer to the networks with 80 and 100 nodes as “medium-sized” and 120 and 150 nodes as “large-scale” networks [97]. The cost factors (i.e., the span establishment cost (F) and the unit cost of transferring the traffic over the spans (C)), the number of the traffic demands, and the magnitude of every traffic have been selected randomly using uniform random selection process. The upper bound and lower bound values of the unit cost were selected between 10 and 100. Similarly, the values of the span establishment cost were selected between 100 and 1000. Moreover, the demand magnitudes (d^r) were selected randomly with a magnitude between 1 and 100. The details of the test case configurations such as the number of nodes, number of possible spans, and number of traffic demands for every test case have been tabulated in Table 3.1. An example of the input file to the proposed algorithm is presented in Appendix A.

Table 3.1. Test case configurations

Test Case	Number of nodes |N|	$\|A\|$	$\|D\|$
1	40	780	400
2	40	780	288
3	40	780	353
4	60	1770	600
5	60	1770	448
6	60	1770	548
7	80	3160	800
8	80	3160	814
9	80	3160	758
10	100	4950	1117
11	100	4950	1042
12	100	4950	953
13	120	7140	1266
14	120	7140	1152
15	120	7140	1151
16	150	11175	1170
17	150	11175	1160
18	150	11175	1168

3.4.2 Validation of the results

We employed both the ILP model and the presented IGA for solving the network topology design and routing problem on various test cases. The IGA results such as runtimes and fitness values are obtained as the average of twenty sets of runs for each network, individually. The obtained results are tabulated in Table 3.2 where the normalized costs are calculated by dividing the IGA fitness (i.e., objective function) value by the ILP model fitness value. The normalized costs (IGA/ILP) represent the optimality of the IGA's solution, where the optimal (minimum) network design cost is obtained from the FCR problem's ILP model.

It can be seen from Table 3.2, that as the number of nodes in a network increases, the ILP runtime becomes progressively increased such that by doubling the

number of nodes of the test cases from 40 to 80, the average ILP runtime increases by almost 30 times. The ILP runtime for larger networks (e.g., test cases 10 to 18 with more than 100 nodes) is quite prohibitive. For instance, the ILP solver could not improve the found solution for test case 10 after more than 100 hours of running. However, for the same test case, the IGA can get to a reasonable optimality gap within a significantly shorter time (i.e., up to 40% better objective function value compared to ILP, within less than 5 hours). If a designer needs to solve the problem just once, then long runtimes might be of little consequence on a build that takes years. However, when network designers analyze various demand matrices and look at many multiple scenarios to determine which design will work best for every scenario, they can run the network design problem a great many times to determine the good configuration. As the design process of large-scale networks can take weeks or months for large scenarios, having a sub-optimal solution within reasonable processing time might be a good trade-off.

Based on the obtained results, it can be seen that the proposed methodology can outperform the ILP both in terms of runtime and objective function value, specifically for large-scale networks. The IGA's CPU process times vary between 0.1 hours and almost 11 hours. For networks with more than 80 nodes, the performance of IGA in terms of the processing time and the objective function value is 100% of the time more efficient than the ILP.

Figures 3.17 throgh 3.22 illustrate the performance of the proposed IGA for all of the test cases. The costs are normalized to the best cost achieved from the benchmark ILP model. Note that in the larger test cases, the first iteration of the IGA already outperforms the benchmark, though we continue iterating to achieve better results. In

Figures 3.17 through 3.22, the solid data points represent the average of the normalized network design costs over twenty sets of IGA runs. The error bars represent one standard deviation among the twenty sets of runs.

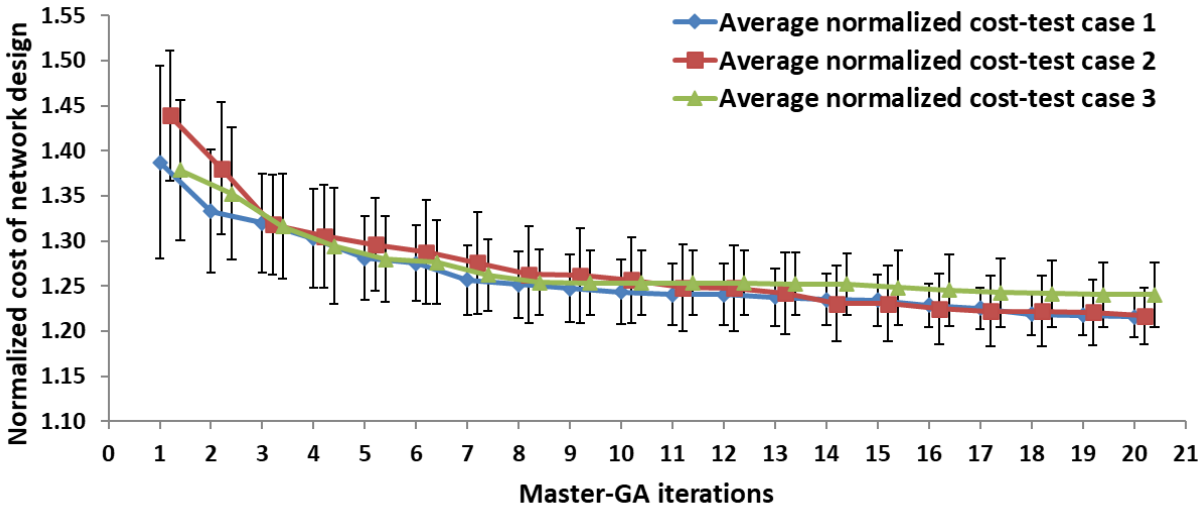


Figure 3.17. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations for test cases 1 to 3.

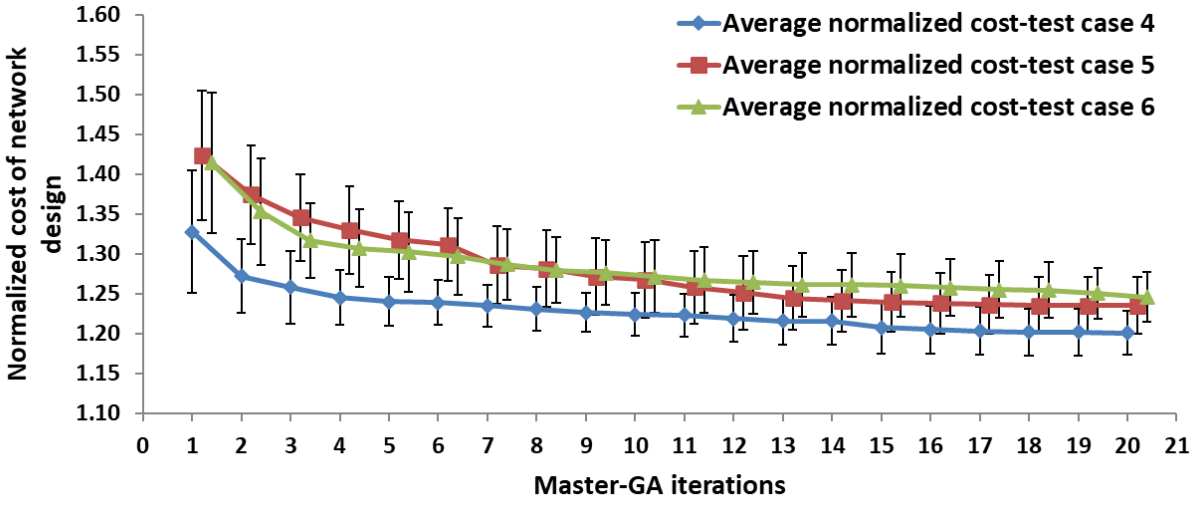


Figure 3.18. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations for test cases 4 to 6.

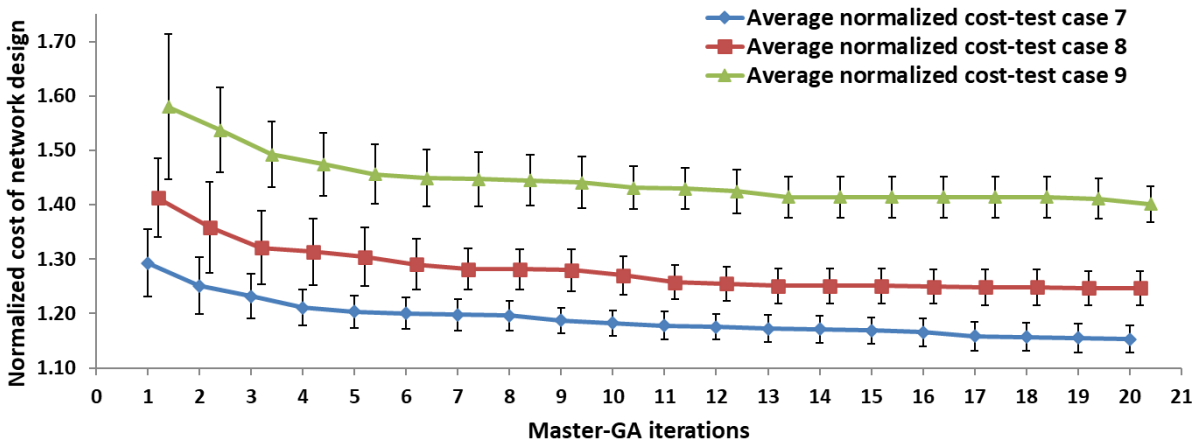


Figure 3.19. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations for test cases 7 to 9.

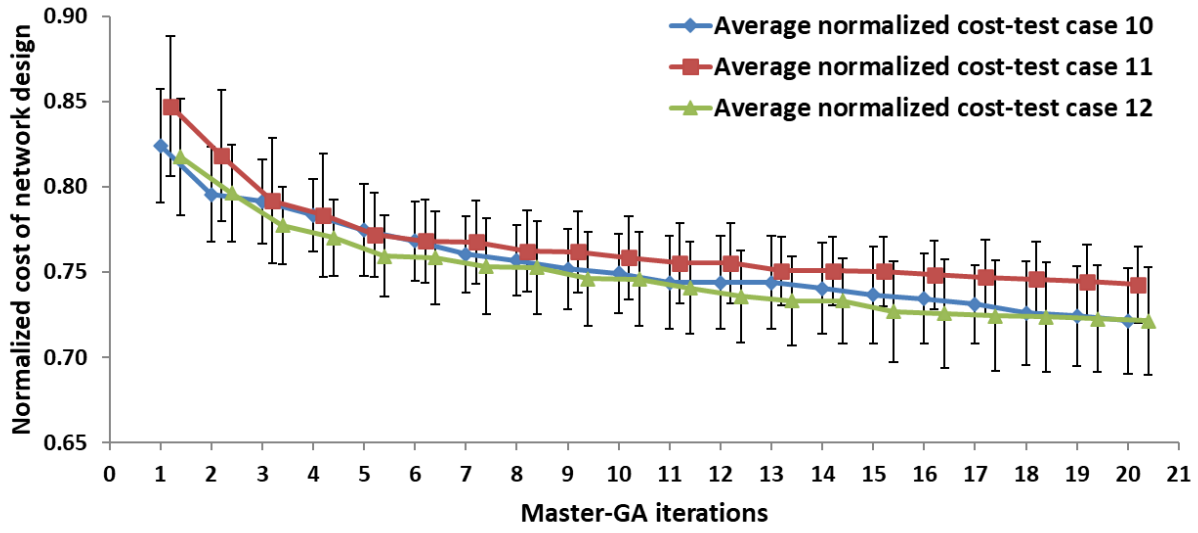


Figure 3.20. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations for test cases 10 to 12.

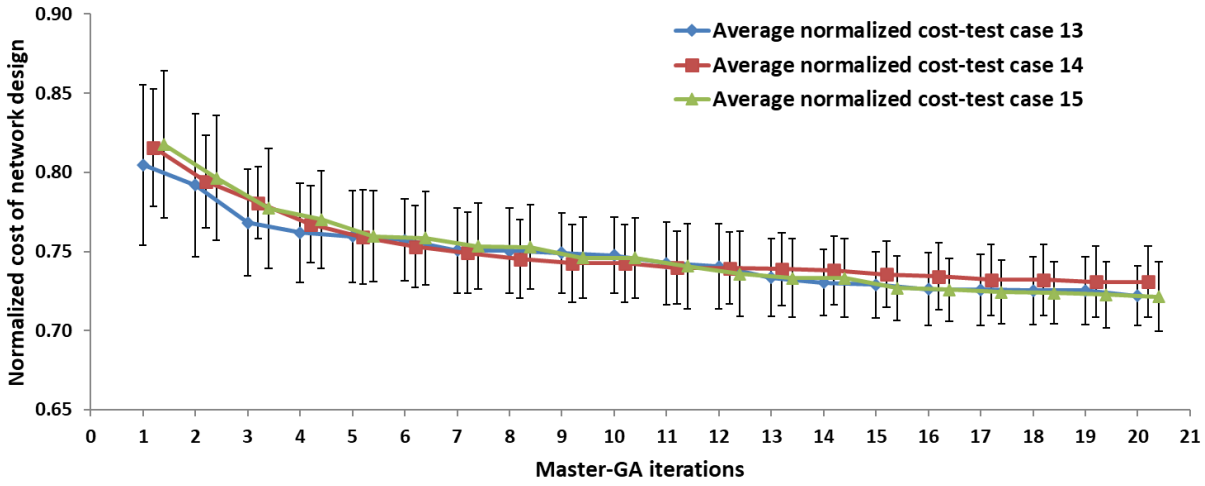


Figure 3.21. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations for test cases 13 to 15.

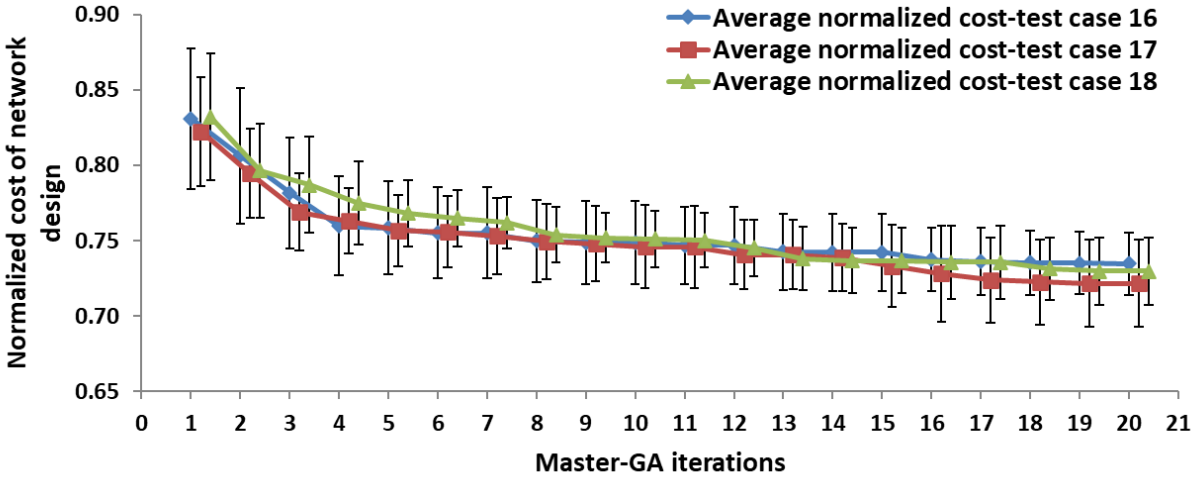


Figure 3.22. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations for test cases 16 to 18.

Table 3.2. Comparison between proposed IGA and ILP results (objective function value and processing times)

	ILP	IGA	
Test case	Runtime (h)	Runtime (h)	Normalized Cost
1	0.9	0.2	1.22
2	0.1	0.1	1.22
3	0.1	0.1	1.24
4	4.0	0.7	1.20
5	1.4	0.4	1.24
6	2.0	0.5	1.25
7	15.9	1.6	1.15
8	17.9	1.7	1.25
9	12.2	1.4	1.40
10	>40[4]	4.3	0.72
11	>40	3.9	0.74
12	>40	3.3	0.72
13	>40	7.6	0.72
14	>40	6.5	0.73
15	>40	6.4	0.72
16	>40	10.5	0.73
17	>40	10.5	0.72
18	>40	10.7	0.73

3.5 Conclusion

In this paper, we have presented an Improved GA-based algorithm for the large-scale network design and routing problem. In the presented algorithm, the concept of TSP has been employed to create a set of initial feasible and bi-connected solutions (i.e., network topologies). The set of sub-optimal initial feasible solutions has been created

[4] The ILP solver was not able to improve the found solution even after 100 hours of running.

using the IPG-GA algorithm. Then, a routing function routes all of the traffic demands on the network topologies that have been obtained from the last iteration of IPG-GA. The obtained population of network topologies is fed to the Master-GA for further topology and routing improvements in an iterative fashion. We have developed two sets of well-designed genetic operators (crossovers and mutations) for breeding new solutions from the initial ones. These operators try to maintain the feasibility and diversity of the solutions.

One of the main contributions of this research work is that the proposed IGA starts with a set of feasible solutions. Thus, even in the very first iteration of the algorithm, we have network topologies that are bi-connected. The bi-connectivity of the initial solutions is the result of their cyclic structure. Using TSP, a set of Hamiltonian cycles has been generated for the initial population of the IPG-GA. Although the initial solutions were feasible for this problem, they were not optimal. Therefore, using the IPG-GA, the set of initial feasible solutions was improved iteration by iteration to get a set of optimal cycles that can be used as sub-optimal network topologies for the Master-GA. As the Master-GA starts with a set of initial feasible solutions, the convergence rate of the algorithm will be more than the case the initial population is generated randomly. Moreover, as the initial feasible solution in this research is bi-connected, the constraints for employing the restoration mechanisms can be incorporated into the algorithm too.

We presented well-designed crossover and mutation operators that try to keep the solutions feasible and decrease the total network design cost while increasing the diversity of the populations. Moreover, we conducted a comprehensive set of experiments to tune the genetic operators' rates. The higher the genetic operators'

rates, the higher the diversity of the population and the higher the runtime. Thus, to reach a trade-off between the population diversity and runtime, a suitable rate for crossover and mutation rates in both the IPG-GA and Master-GA was obtained. The proposed algorithm resulted in higher fitness levels and shorter runtime for all test cases with 100 nodes and more.

Chapter 4 : Impact of Traffic Demand Distribution on Large-scale Network Topology Design

This chapter provides complementary results and discussions on the performance of the presented Improved Genetic Algorithm in Chapter 3. In this chapter, we extensively evaluated the performance of the presented Genetic Algorithm over a variety of large-scale test cases considering various traffic distributions to be routed[5].

4.1 Introduction

In this chapter, we employed the presented Improved Genetic Algorithm (IGA) to solve the previously-discussed Network Topology Design and Routing problem. We studied the effect of the routing order of the traffic demands on the large-scale network design process, with regard to the traffic distribution, while the network topology is variable. In other words, we studied the performance of the presented Genetic Algorithm under various traffic demands with different orders of the traffics to be routed. Analyzing the *clustered* (*hubbed*) and the *scattered* traffic distributions, the results showed that for clustered demand distributions, the total cost of the network design and routing is more sensitive to the order of routing the traffic demands compared to the scattered ones.

4.2 Methodology

Every traffic demand consists of three elements:

(1) origin node, from which the traffic has to originate and outward toward its destination,

[5] Portions of this chapter along with parts of Chapter 2 are slated for publication as a conference paper: S. Doostie, T. Nakashima-Paniagua, and J. Doucette, “An Improvement on the Network Topology Design and Routing Problem for Large-Scale Networks”, conference TBD.

(2) the destination node, to which the traffic has to arrive at, and

(3) the magnitude of the traffic demand (*d*) which represents the total amount of data or number of wavelengths that should be transferred from the origin node to the destination node.

Routing a traffic demand between its origin and destination nodes is equivalent to selecting a set of consequence spans that start from the origin node and end at the destination node. If any of the selected spans for the route have not been established in the network yet, there will be a span establishment cost (*F*) associated with that span. Also, the spans on the selected route will have to carry the traffic on them to transfer the traffic. So, there will be a variable cost associated with carrying one unit of the traffic demand on every span represented by C.

So, for the first routed demand, as we established a set of spans as parts of the route, we added their F values to the cost of the design and from this point forward, whenever we used any of these established spans, the cost associated with them would be only the cost of carrying demands ($C \times d$). Now, consider the routing process of the second traffic demand. As there is a set of spans already established in the network, the costs associated with those spans will be reduced to only the cost of routing the traffic demands on them (only $C \times d$). In other words, for the already established spans, $F = 0$. Therefore, the established spans (which now have less cost) have more probability to be selected for routing the upcoming traffic demands. On the other hand, if we had selected another traffic demand to be routed first, we would have another set of spans established on the network which would result in a different cost matrix for spans. As a result, we can say by routing each traffic demand, there is a tendency for upcoming

traffic demands to use the established spans as a part of their routes to achieve less cost (i.e., routing each demand generates a bias for other demands' routes selection). Thus, the order of routing the traffic demands has a direct effect on the converging rate of the total Network Topology Design and Routing problem cost and efficiency.

4.2.1 Experimental Setup

We sorted the set of traffic demands based on their magnitudes (*d*) in three different ways as ascending, descending, and random orders. For two patterns of traffic demands (clustered and scattered) we observed that the efficiency of the proposed GA can be affected by considering various orders of routing the traffic demands.

We coded the presented GA in Python 3.7 [73] and ran the GA on a Microsoft Windows Server 2012 R2 11 Standard, 128 GB RAM, x64-based PC, Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30 GHz. We tested the GA on ten different 60-node and ten different 100-node test-case configurations, which we designed for this research work. As discussed in the previous section, we seek to understand the effect of routing order of the traffic demands on the performance of the presented GA. In the first traffic distribution, which we call uniform random traffic distribution, each node as the origin node of traffic demand, was assigned a *uniform random* number of traffic demands between 5 and 10. In the second traffic distribution, which we call *hubbed* traffic distribution, 6 and 10 randomly selected nodes were considered as hub points for the 60-node and 100-node networks respectively. Every hub point in the 60-node networks was assigned a uniform random number of traffic demands between 40 and 50. Every hub point in the 100-node networks was assigned a uniform random number of traffic demands between 60 and 70. For simplicity, let us refer to the 60-node and 100-node

networks with hubbed traffic distribution as group 1 and group 2 of the test cases, respectively. Similarly, let us refer to the 60-node and 100-node networks with scattered traffic distribution as group 3 and group 4 of the test cases, respectively.

4.3 Results and discussion

The tabulated results in Tables 4.1 and 4.2 show the average Network Topology Design and Routing cost as the objective function value over 10 consecutive runs for each test case. In Table 4.1, the average cost for random, ascending, and descending order of the traffic demands for the hubbed traffic distributions have been tabulated from left to right for the 20 test cases in groups 1 and 2. Similarly, Table 4.2 provides the obtained results for the test cases in groups 3 and 4 when the scattered traffic distributions are applied. The objective value improvement has been calculated as the relative improvements when employing the descending order compared to the random and ascending order of traffic, respectively. These improvements have been tabulated in the last two columns of Tables 4.1 and 4.2.

Our analysis showed that in all of the cases, sorting the traffic demands before routing in a descending order based on their magnitude, resulted in the lower cost (lower objective function value), and sorting in an ascending order resulted in the highest cost. The relative objective function value improvement between ascending and descending orders was as high as 18% between the two scenarios for the test cases with a hubbed traffic distribution and 16% for test cases with scattered traffic distribution. The averages of objective function improvements for hubbed traffic distribution were 14%$\pm$2% and 17%$\pm$1% for test cases in groups 1 and 2, respectively.

Similarly, the averages of objective function improvements were 13%±1% and 15%±1% for test cases in groups 3 and 4, respectively.

Tabel 4.1.Total network design and routing cost for Clustered traffic distribution

Network Size			Traffic order			Total Objective Value Improvement%	
			Random	Ascending	Descending		
Test case	|N|	|S|	Mean Objective Value (*100k)	Mean Objective Value (*100k)	Mean Objective Value (*100k)	Descending to random comparison	Descending to ascending comparison
1	60	1770	59	66	56	5%	15%
2			61	64	58	5%	9%
3			56	64	55	2%	14%
4			59	64	56	5%	13%
5			55	62	52	5%	16%
6			53	59	51	4%	14%
7			61	64	56	8%	13%
8			61	66	55	10%	17%
9			63	70	59	6%	16%
10			60	67	57	5%	15%
11	100	4950	144	162	133	8%	18%
12			151	164	134	11%	18%
13			140	157	130	7%	17%
14			142	156	133	6%	15%
15			148	167	139	6%	17%
16			142	157	134	6%	15%
17			138	154	131	5%	15%
18			144	158	133	8%	16%
19			145	160	132	9%	18%
20			140	158	130	7%	18%

Table 4.2. Total network design and routing cost for Scattered traffic distribution

Network Size			Traffic order			Total Objective Value Improvement%	
			Random	Ascending	Descending		
Test case	|N|	|S|	Mean Objective Value (*100k)	Mean Objective Value (*100k)	Mean Objective Value (*100k)	Descending to random comparison	Descending to ascending comparison
21			67	74	65	3%	12%
22			65	72	62	5%	14%
23			65	73	64	2%	12%
24			64	71	61	5%	14%
25			63	69	59	6%	14%
26	60	1770	66	70	63	5%	10%
27			63	69	60	5%	13%
28			67	73	63	6%	14%
29			67	73	64	4%	12%
30			68	74	65	4%	12%
31			148	163	138	7%	15%
32			156	169	143	8%	15%
33			150	161	141	6%	12%
34			153	171	146	5%	15%
35	100	4950	151	165	139	8%	16%
36			149	165	142	5%	14%
37			146	163	137	6%	16%
38			147	165	139	5%	16%
39			153	163	142	7%	13%
40			145	164	138	5%	16%

Figures 4.1 through 4.4 depict a graphical representation of the objective function improvements for descending order of the traffic demands compared to the random and

ascending orders. Figures 4.1 and 4.2 represent the improvements for the test cases 1 to 20 with the clustered distribution of the traffics. Figures 4.3 and 4.4 represent the improvements for the test cases 21 to 40 with the scattered distribution of the traffics.

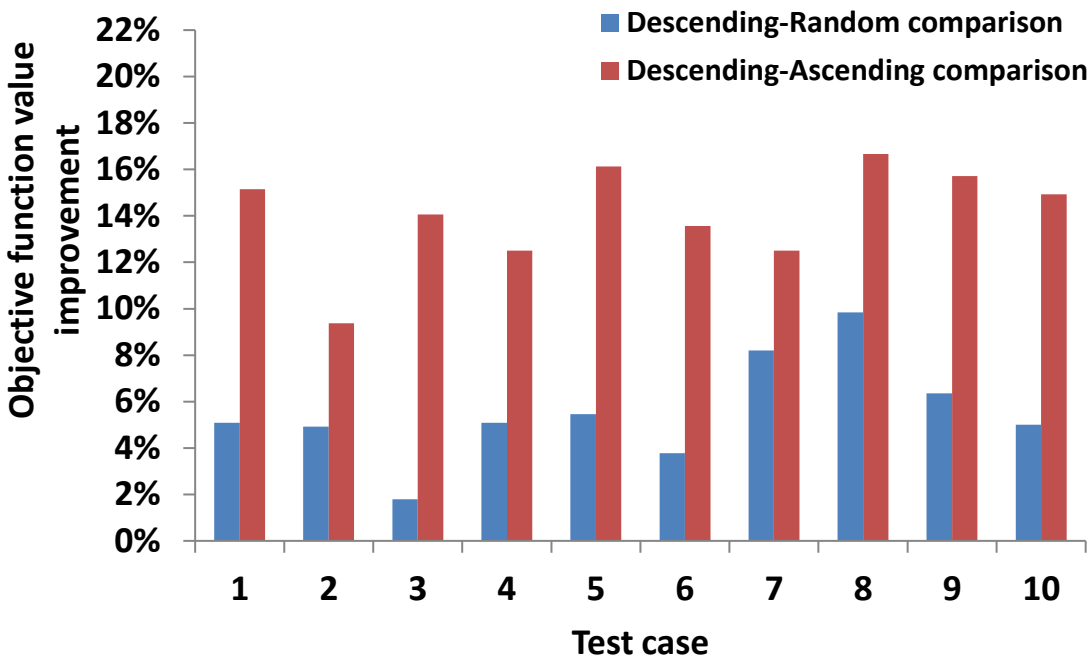


Figure 4.1. The objective function value improvements for descending order of traffic routing compared to random and ascending order for group 1.

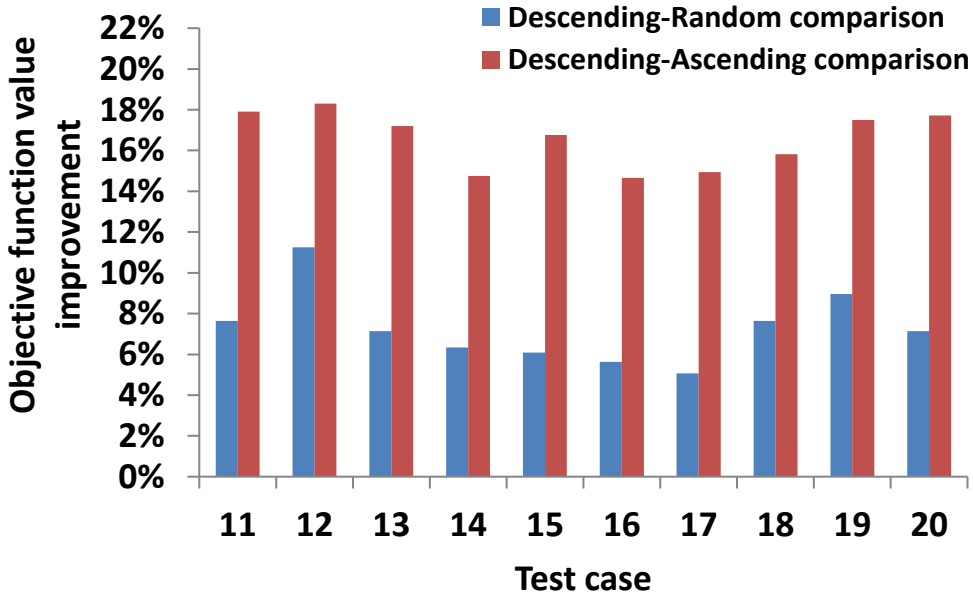


Figure 4.2. The objective function value improvements for descending order of traffic routing compared to random and ascending order for group 2.

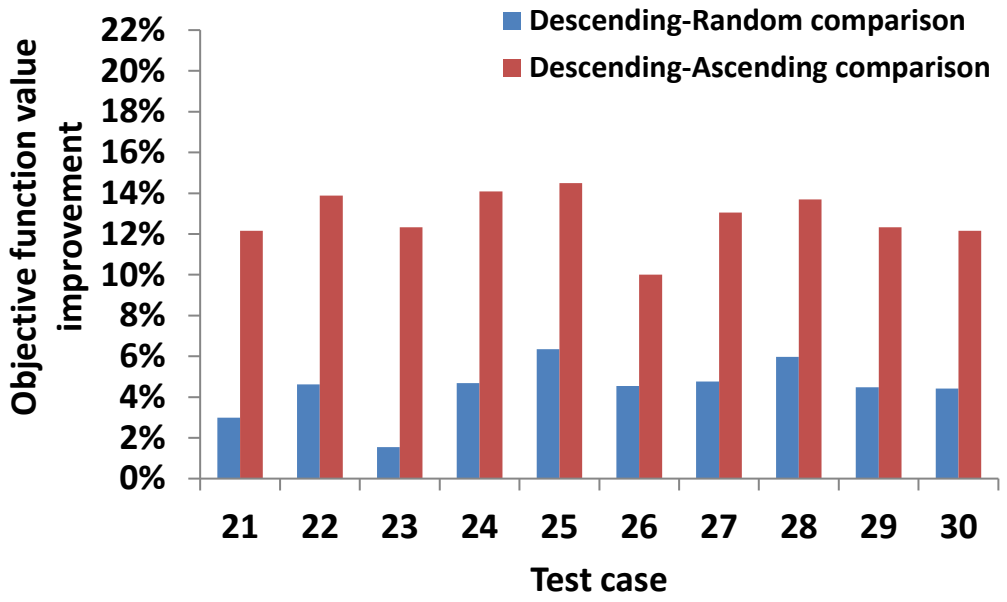


Figure 4.3. The objective function value improvements for descending order of traffic routing compared to random and ascending order for group 3.

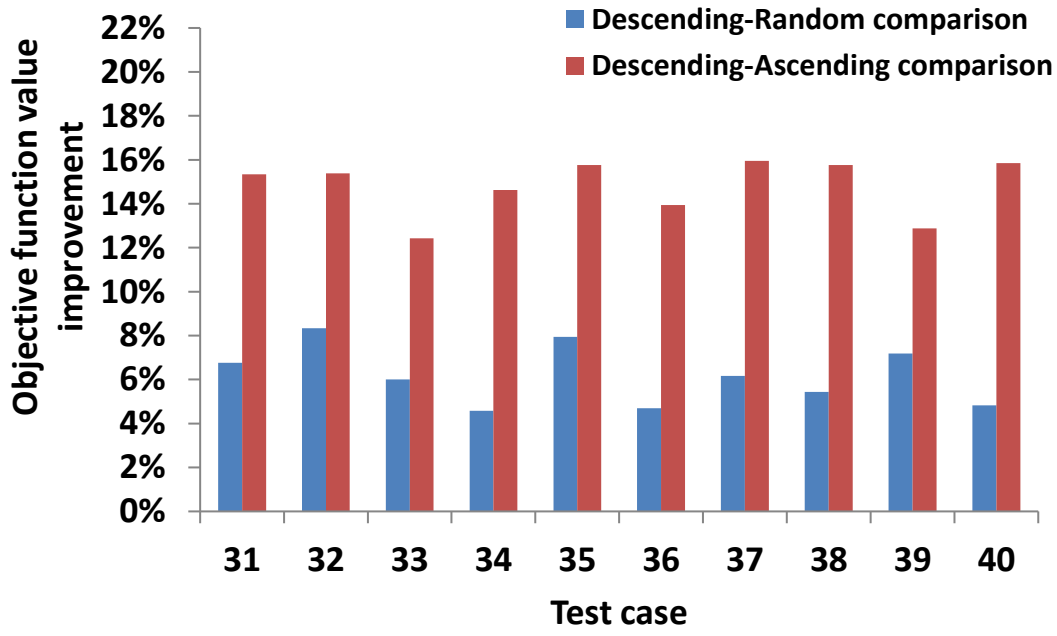


Figure 4.4. The objective function value improvements for descending order of traffic routing compared to random and ascending order for group 4.

4.4 Conclusion

On improving a GA-based network design and routing algorithm, this chapter provided an improvement based on the order of routing the traffic demands on the network considering various demand distributions. In this work, two general traffic distributions were studied over 40 large-scale networks. The results showed the effect of the order of routing the traffic demands on the networks with clustered traffics is more significant than in networks with the scattered distribution of traffic. The objective function value improvements for hubbed distribution of the traffics were up to 18% with an average of 14%$\pm$2% and 17%$\pm$1% for the test cases in groups 1 and 2, respectively. These improvements for test cases with scatter distribution of traffic were up to 16% with an average of 13%$\pm$1% and 15%$\pm$1% for groups 3 and 4, respectively.

Moreover, the average objective function value improvement increased with the network size. Our results showed that the average objective function value improvements were higher for groups 2 and 4 compared to groups 1 and 3 in both traffic distributions. For groups 2 and 4, the average objective function improvements over the two traffic distributions were 17% and 15%, respectively while these values were 14% and 13% for groups 1 and 3, respectively.

Chapter 5: Heuristic Approach for Survivable Large-scale Network Design Problem

Designing telecommunication networks with reliable infrastructure and high levels of performance that survive any failure is one of the most critical goals in the telecommunication industry. The overall network performance can be maintained by two means: (1) the functionality of the network's constitutive components (e.g., fiber optics and optical cross connectors) and (2) the network's ability to employ contingency plans (i.e., considering extra reserved capacity as the spare capacity to back up the affected traffic) in the event of the component failure to maintain the required functionality of the network. These contingency plans are known as "restoration mechanisms" that based on the available spare capacity in the network, provide new plans for routing the affected traffics[6].

Various failure patterns can disrupt the functionality of the networks such as single, dual, or multiple span-failure scenarios. Although single-span failures are the most common failure patterns, dual-span failures often occur as well. In this chapter, we present a novel approach for designing large-scale networks that are fully restorable in the event of any dual span-failure scenario using the span restoration mechanism. For simplicity, from now on we refer to "dual span-failure" as "dual failure".

An efficient deterministic approach for designing a dual-failure restorable network is by use of an integer linear programming (ILP) model. However, conventional ILP models lose their efficiency when being applied to large-scale networks. In this chapter,

[6] Portions of this chapter along with parts of Chapter 2 are slated for publication as a journal paper: S. Doostie, T. Nakashima-Paniagua, and J. Doucette, "A Novel Design Approach for Dual-Failure Span-Restorable Large-Scale Networks using Integer Linear Programming", journal TBD.

we designed a new ILP-based heuristic approach for solving the large-scale dual span-failure restorable network design problem. We used the previously developed ILP models as benchmarks for various network sizes. Our results showed that for large-scale networks, the previously-introduced ILP models encounter both computational memory limit and prohibitive runtime which make them inefficient approaches. Using the presented approach, we were able to determine sub-optimal solutions for large-scale networks considering full restoration against any dual-failure scenario where the exact methodologies were not able to find any solution after several days of runtime.

5.1 Introduction

Telecommunication networks have been extensively developed all around the world. Considering the continuing growth in demand for telecommunication services, the performance of telecommunication networks continues to be of the highest importance. A network's overall performance can be defined as the network's ability to provide the expected service in the expected time during the period that the network is intended to function. A network's quality of service can be secured through the network's ability to overcome inevitable accidents and failures using restoration mechanisms.

5.1.1 Network Structure

A network's structure consists of its underlying topology (nodes and spans) along with a detailed description of its constituent components. The topology of a network refers to the nodes and spans of the network and how they are connected. As mentioned earlier, in Chapter 2, the network's operation can be defined as the ability of the spans (e.g., optical fibers) to carry the traffic between two nodes. While the network is operating,

there could be failures happening at any time and anywhere in the network that causes service outages. In the same chapter, we described various restoration mechanisms that can be used to maintain the functionality of the networks in the event of such failures. In this chapter, we studied the Span Restoration (SR) mechanism for the full dual failure restorability of large-scale networks.

As mentioned earlier in Chapter 2, the optimal network design problem includes the optimal network topology design and optimal spare capacity placement to satisfy restorability constraints in the event of failures. One of the well-developed approaches for network design problems is Integer Linear Programming (ILP) [131]. There are various ILP formulations developed for the optimal design of restorable networks. One of the main ILP formulations is Mesh Topology Routing and Spare capacity (MTRS) [33] which can jointly optimize the network's topology and spare capacity allocation. This ILP becomes quite time-consuming even for medium-sized networks, thus, researchers separated the topology design and spare capacity allocation problems and presented individual ILPs for each one of them. One of the well-known ILP models for optimal network topology design problem is the Fixed Charge plus Routing (FCR) problem [32] which we dedicated the whole of Chapter 3 to it.

The approach introduced in Chapter 3 is focused on the routing of working traffic only and it does not study the restoration requirements such as spare capacity placement on the network. Reviewing the literature, to the best of our knowledge, the problem of large-scale network design considering full dual-failure span restoration has not been studied comprehensively. In this chapter, we studied the full dual-failure restoration of large-scale networks using the span restoration mechanism allowing any

required topology augmentation. We assumed the working traffics have already been allocated on the network, and we seek to place the minimum required spare capacity to protect the network against any dual failure among the working traffic. We presented an ILP model capable of solving the large-scale dual-span restorable network design problem. The presented approach is also applicable to other restoration mechanisms upon updating the restoration constraints and backup routes. To validate our results, inspired by the ILP model introduced in [33] as Reserved Network Fixed Charge plus Spare capacity problem (RN-FCS) for single failure span restorable networks, we developed the dual failure span restorable ILP as dual-failure RN-FCS for designing large-scale networks that are restorable against any dual failure.

5.1.2 Span Restoration Mechanism

Recall the span restoration mechanism that provides restoration of working traffic on failed spans. Span restoration provides the network with backup routes between the end nodes of the failed spans to carry the affected traffic between the end nodes of the failed span. As an example, consider the nine-node network in Figure 5.1, where span CD is a part of the route between nodes A and I and has failed. The span restoration mechanism restores the affected traffic on span CD (between nodes C and D) by rerouting the traffic through one or more possible backup routes between nodes C and D (e.g., CED, CHGD, or CHID).

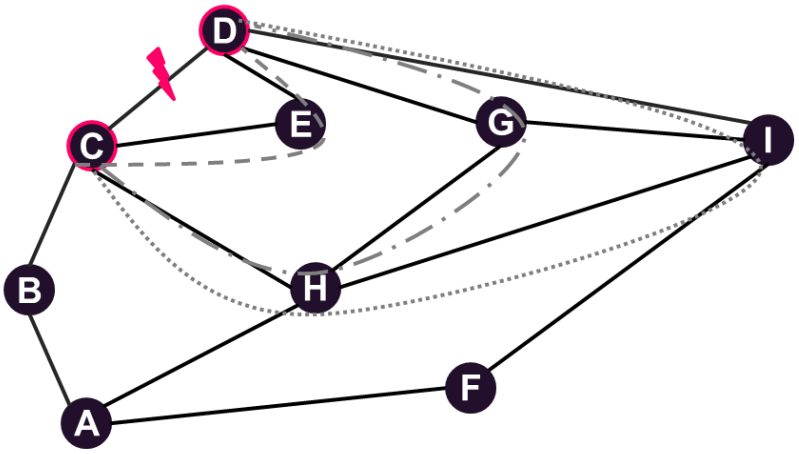


Figure 5.1. General Span Restoration mechanism. Upon the failure of the span CD, the SR mechanism provided three backup routes (demonstrated by dashed and dotted lines) to restore the span CD.

5.1.3 State of the Art

With a specific focus on the span restoration mechanism, several works studied single-failure span restoration in various network topologies via ILP models [132]–[134]. As a further step, one can consider more complex situations such as failure of shared-risk spans. The shared-risk spans are a set of spans in a network that have the same cause of failure. Thus, in the event of a failure, several spans may be affected simultaneously and therefore increase the complexity of the failure scenario. The authors in [135] and [136] studied the effect of dual-span failure under shared-risk spans on the spare capacity allocation of networks using the span restoration mechanism. The results presented in [135] showed that the existence of even two share-risk spans can significantly increase the required spare capacity network-wide.

In our work herein, we focus on dual-failure scenarios where two spans in a network fail simultaneously or overlap in time. Although the dual and multiple-failure scenarios are much more impossible than single failures, they do occur [105] and [123].

In [137], the authors studied and comprehensively compared various restoration mechanisms for restoring dual-failure scenarios. In general, full restoration of all the dual failure scenarios requires the network to be three-connected [138]; there should be at least three span-disjoint routes between every node pair for the network to be restorable. The authors in [139], presented an ILP model to determine the minimum spare capacity allocation considering dual-failure scenarios using partially disjoint paths. The authors in [44], [128], and [140] presented various ILP models for the analysis of the dual failure restoration of span-restorable networks considering a network with a given fixed topology. The authors in [44] presented an ILP model for minimizing the non-restored working capacity due to dual-failure scenarios in networks with fixed topologies while considering an upper bound for the cost of assigning spare capacity. They calculated the upper bound for the cost based on the minimum budget required for full single failure restoration of the network and determined the maximum restored working capacity for dual-failure scenarios.

5.1.4 Dual Failure Span Restoration

In this chapter, we focus on the full dual failure restorability of large-scale networks where each dual-failure scenario is defined as the simultaneous failure of two spans in the network. We depicted the basics of the SR mechanism in dual failure restoration in Figure 5.2. Assume spans CD and GH in the presented network fail simultaneously. The span restoration mechanism provides backup routes to restore the affected traffic on the failed spans. In this example, for simplicity, we assume a hop limit of four spans in every backup route, so the number of backup route choices would be limited. The affected working traffic on the span CD may be rerouted over the dashed-dotted route.

The affected working traffic on the span GH may be rerouted over one or several of the dashed routes. Considering restoration constraints and the objective to minimize the total cost of network design and sparing, the backup route selection may vary from network to network.

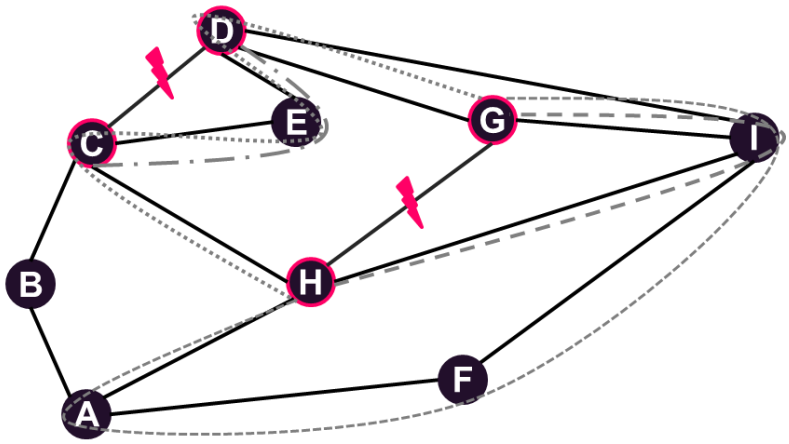


Figure 5.2. Dual failure Span Restoration mechanism with four hop limits for the backup routes; Spans CD and HG failed simultaneously. The backup routes for spans CD and HG have been shown with dashed-dotted and dashed lines, respectively.

During span establishment for routing the working traffic in the network design process, the resulting network may not be three or even bi-connected. And, as mentioned earlier in Section 5.1.3, the network should be at least three-connected to be fully dual-failure restorable. Thus, to make the network fully dual-failure restorable, in the presented model, we allowed for the necessary topology augmentation in the form of required span establishments. In the presented approach, the spare capacity is placed in the network for full dual failure restoration and new spans can be established in the network, to provide the required connectivity in the network.

The total number of dual failure scenarios (DFS) where we have $|S_{existing}|$ existing spans in a network, is $|S_{existing}| \times (|S_{existing}| - 1) / 2$. For instance, a network with 100

spans can experience 4950 different dual failure scenarios. As the dual failures are assumed to happen simultaneously, the order of the two failed spans does not matter. For instance, the failure scenario (CD, HG) is the same as (HG, CD). For every dual failure scenario, the eligible backup route for every span is a backup route that does not pass through the other failed span. Therefore, from the set of designed backup routes for every span, some of them might not be eligible for restoration, depending on the other failed span. For instance, considering the dual failure scenario (CD, HG) in Figure 5.2, the backup route CHGD for the restoration of span CD is not eligible anymore as it passes through the other failed span, HG. Therefore, the need to find eligible backup routes contributes to the complexity of the problem.

5.2 The Motivation for the Proposed Approach

In this section, first, we introduce an exact method for designing a full dual-failure span restorable network. Then, we introduce an improved version of the developed method that has higher efficiency when applied to medium-scale networks. Finally, we take a step further and develop a new ILP-based heuristic method that is more efficient for solving large-scale instances of the dual restorable network design problem. The followings are the detailed description of the three introduced models for full dual-failure span restorable network design.

5.2.1 Reserve Network Fixed Charge plus Spare Capacity Problem for the Restoration of Dual Failures (dual-failure RN-FCS)

We focus on full dual failure span restoration in large-scale networks. To have a valid and deterministic benchmark, we first designed an ILP model as a reserved network fixed charge plus spare capacity problem for the restoration of dual failures (dual-failure

RN-FCS), inspired by the ILP model introduced in [33]. Considering the set of spans with working traffic in the network, the proposed dual-failure RN-FCS places enough spare capacity on the network to restore working traffic affected by any dual-failure scenario. The goal is to minimize the total cost of additional span installation and spare capacity placement while providing 100% restoration for any dual-failure scenario among the spans with working traffic on them.

5.2.1.1 Notation

In this section, we introduce all the elements of the proposed dual-failure RN-FCS ILP model with their descriptions:

ij represents the span between nodes *i* and *j*.

Sets:

N is the set of nodes in the network.

S_{all} is the set of all of the possible spans in the network.

$S_{existing} \subseteq S_{all}$ is the set of existing spans in the network that carry working traffic.

$S_{absent} \subseteq S_{all}$ is the set of absent spans in the network that may/may not be selected for carrying spare capacities.

Parameters:

C_{ij} is the cost of carrying each unit of spare capacity on span ij.

F_{ij} is the cost of installing span ij in the network.

W_{ij} is the working traffic on span ij.

M is a very large positive number.

Decision Variables:

$s_{ij,mn}^{kp}$ is the amount of restoration flow routed over span kp when spans ij and mn are failed spans.

s_{ij} is the total spare capacity placed on span ij to support all of the restorations flows routed over ij.

$\hat{\delta}_{ij}$ is the binary decision variable for representing the existence of span ij in the network. $\hat{\delta}_{ij}$ is equal to 1 if ij has been installed in the network and 0 otherwise.

$$Minimize\ Total\ cost = \sum_{ij \in S_{existing}} (C_{ij} \times s_{ij}) + \sum_{ij \in S_{absent}} (C_{ij} \times s_{ij} + F_{ij} \times \hat{\delta}_{ij}) \quad (12)$$

Subject to:

$$\sum_{ik \in S_{all} |\ ij \neq mn \neq ik} s_{ij,mn}^{ik} = W_{ij} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (13)$$

$$\sum_{mk \in S_{all} |\ ij \neq mn \neq mk} s_{ij,mn}^{mk} = W_{mn} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (14)$$

$$\sum_{kj \in S_{all} |\ ij \neq mn \neq kj} s_{ij,mn}^{kj} = W_{ij} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (15)$$

$$\sum_{kn \in S_{all} |\ ij \neq mn \neq kn} s_{ij,mn}^{kn} = W_{mn} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (16)$$

$$\sum_{\forall pk \in S_{all} | \, k \notin \{i,j,m,n\}} s_{ij,mn}^{pk} - \sum_{\forall kp \in S_{all} | \, k \notin \{i,j,m,n\}} s_{ij,mn}^{kp} = 0 \quad (17)$$

$$\forall ij \in S_{existing}, \forall mn \in S_{existing}, \forall p \notin \{i,j,m,n\}$$

$$s_{kl} \geq s_{ij,mn}^{kl} \quad \forall ij \in S_{existing}, \forall mn \in S_{existing}, \forall kl \in S_{all} | ij \neq mn \neq kl \quad (18)$$

$$s_{ij} \leq \hat{\delta}_{ij} \times M \quad \forall ij \in S_{all} \quad (19)$$

$$\sum_{ij \in S_{absent}} \hat{\delta}_{ij} + |S_{existing}| \geq |N| \quad (20)$$

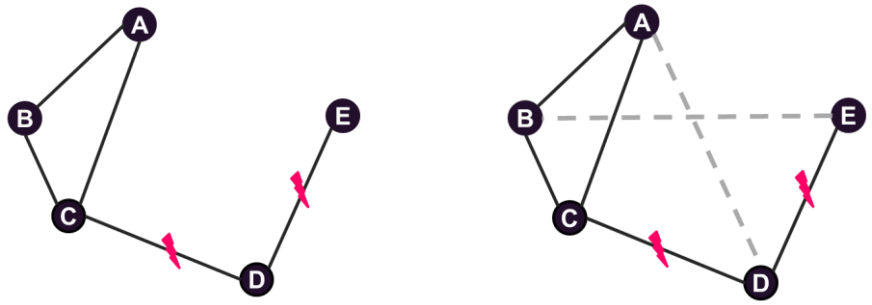
$$\sum_{j \in N | j \neq i} \hat{\delta}_{ij} \geq 2 \quad \forall i \in N \quad (21)$$

As discussed in Section 5.3.1, the objective function (Equation (12)) consists of two types of cost: (1) the cost of spare capacity placement for every span in the network, and (2) the cost of span installation for any span from the set of S_{absent} that has been newly established in the network. Constraints (13) and (14) guarantee enough spare capacity placement on the spans outgoing from the origin nodes of the failed spans to restore the affected working traffic. Constraints (15) and (16) make sure there is enough spare capacity placed on the spans incoming into the destination nodes of the failed spans to restore the affected working traffic. Constraint (17) is the transhipment constraint that for every node except the end nodes of the failed spans, ensures the incoming restoration flow is equal to the outgoing restoration flow. In another word, there is no production or consumption of restoration flow on the intermediate nodes in the network. Constraint (18) measures the required spare capacity on every span to

support all the restoration flows passing over that span. Based on the placed spare capacity on every span, constraint (19) determines whether or not a span needs to be installed. Constraints (20) and (21) are related to the topology of the network and establish a lower bound on the number of spans to be installed.

Topology-wise, in dual failure restorability, the network's topology needs to be three-connected. Thus, every node in the network must have a nodal degree of 3 or higher. So, the right-hand side of the constraint (21) should be 3. However, in this work, we were only concerned about the full dual failure restorability of the spans that carry working traffic. In another word, the dual-failure scenarios could only occur among the spans in the set $S_{existing}$. So, if there is a node that has only one span (from the set $S_{existing}$) attached to it, that node needs at least one other span to be incident on it to be able to restore the working traffic on the first span. Let's look at an example to illustrate this situation. Let us consider the network with five nodes in Figure 5.3-a where the five black solid spans represent the spans that carry working traffic and the two gray-dashed spans represent the newly established spans for placing spare capacity. We assumed the dual failure scenarios only occur for the solid black spans. As one of the dual failure scenarios, let's consider span ED and CD fail. To restore the affected working traffic on span ED, the network must have at least one span other than ED attached to node E and one span other than ED and CD, attached to node D. Thus, the spans BE and AD have been installed in Figure 5.3-b. Therefore, for node E with one span with working capacity attaching to it, the minimum nodal degree of 2 suffices. This condition happens for every node that has only one span from the set $S_{existing}$ connected to it. On the other hand, if a node has more than one span from the set $S_{existing}$ connected to it (e.g., nodes

A, B, C, and D in Figure 5.3-a), that node must have a nodal degree of 3 or higher. This justifies the nodal degree greater than or equal to 2 in constraint (21).



(a) (b)

Figure 5.3. In a dual failure scenario in a network with five nodes and five spans carrying working traffic (solid black lines), (a) assume spans ED and CD simultaneously fail, (b) applying span restoration mechanism, two new spans (dashed lines) were installed for placing spare capacity to support the affected traffic on spans ED and CD.

5.2.2 Proposed dual-failure RN-FCS with Reduced Search Space

We aim to reduce the computational complexity of the proposed dual-failure RN-FCS problem by reducing the size of the search space. In another word, we reduce the number of possible spans in the set S_{all} to make the search space smaller and thus decrease the computational complexity of the problem. The reason behind this action comes from the results we obtained from evaluating the performance of the dual-failure RN-FCS model in Section 5.3.1, where we can see that the proposed dual-failure RN-FCS ILP is not efficient when being applied to networks with 80 nodes and more.

5.2.2.1 Search Space Reduction by Removing Spans

We developed an algorithm named “Space Reduction” to eliminate a specific percentage of the high-cost spans from the set S_{all} while avoiding partitioning the graph.

For every network, we selected a set of less-valued spans to be removed from the search space. The selected spans are the ones with the longest lengths. For instance, a span that is connecting the two farthest nodes has a really low chance of being selected as a span on the optimal backup route because of its long length. In another word, as a span's installation cost is a multiplier of the Euclidean distance between its end nodes, the spans with the highest costs are longer and therefore have a smaller chance of being selected for a minimum cost network design solution.

5.2.2.2 Pseudocode of the Space Reduction Algorithm

Following is the pseudocode of the Space Reduction algorithm. First, we introduced the input and output parameters. Second, we explained the steps of the algorithm followed by detailing the designed functions within the algorithm.

Algorithm 4. Space Reduction Algorithm

Inputs

S_{all}: the set of all of the possible spans in the network.

A: the set of end-nodes of every span in S_{all}.

$S_{existing}$: the set of existing spans in the network that carry working traffic.

W: the set of working traffics on the spans in $S_{existing}$.

S_{absent}: the set of absent spans = S_{all} - $S_{existing}$.

SDel: the percentage of $|S_{absent}|$ to be deleted.

Num_Del: the number of spans in S_{absent} to be deleted.

Del_Index: the set of spans to be deleted from the network.

C: the set of costs of carrying traffic units on every span in S_{all}.

C_temp: a temporary variable to hold the costs of carrying traffic units on every span in S_{all}.

F: the installation cost of every span in S_{all}.

F_temp: a temporary variable to hold the installation costs of every span in S_{all}.

Dijkstra's algorithm [141]

Output

*Updated_* S_{all}: the reduced set of all the possible spans in the network.

Deleting Extra Spans:

1. *Num_Del* ← floor ($|S_{absent}| \times (SDel/100)$)
2. *Index_List* ← Sort *F* in Descending Order (*Index_List* stores the indices of the sorted spans)
3. *Index_List*(spans in $S_{existing}$)=[ ] (remove the indices of the spans in $S_{existing}$ from the *Index_List*)
4. *Del_Index* ← *Index_List*(1: *Num_Del*)
5. *F_temp*(*Del_Index*) = *M* (*M* shows a very large installation cost)
6. *C_temp*(*Del_Index*) = *M* (*M* shows a very large cost of carrying traffic units)
7. *F_temp*(spans in $S_{existing}$)=0
8. *F*(spans in $S_{existing}$)=0

Making sure the resulting network is at least two-connected:

9. Function *F_temp*, *C_temp* = Update_Cost(*F_temp*, *C_temp*, *Route_temp*)
   for *j* from 1 to length(*Route_temp*) – 1{
   F_temp(spans in $Route\_temp_j$) = ∞}

10. Function *F_temp*, *C_temp*, *Del_Index* = Add_New_Span(*F*, *C*, *F_temp*, *C_temp*, *Del_Index*)
    F_temp(spans in *Del_Index*(end)) = *F*(spans in *Del_Index*(end[7]))
    C_temp(spans in *Del_Index*(end)) = *C*(spans in *Del_Index*(end))
11. For *i* in $S_{existing}$ {
        Origin = *A*($S_{existing\ i}$, 1)
        Destination = *A*($S_{existing\ i}$, 2)
        Counter = 1
        While Counter <= 2
            Route_temp ← Dijkstra (*F_temp*, *C_temp*, Origin, Destination)
            If *Route_temp* ≠ ∅
                F_temp, *C_temp* = Update_Cost(*F_temp*, *C_temp*, *Route_temp*)
                Counter = Counter + 1
            Else
                F_temp, *C_temp*, *Del_Index* = Add_New_Span(*F*, *C*, *F_temp*, *C_temp*, *Del_Index*) }

Update the set of possible spans and their associated costs:

12. S_{all} (spans in *Del_Index*) = [] (remove the spans in *Del_Index* from S_{all})
13. S_{absent} (spans in *Del_Index*) = [] (remove the spans in *Del_Index* from S_{absent})
14. *F*(spans in *Del_Index*) = [] (removing the costs of the deleted spans from *F*)
15. *C*(spans in *Del_Index*) = [] (removing the costs of the deleted spans from *C*)

5.2.3 Span Restoration using Backup Routes: Introducing Backup Routes

According to the obtained results (see section 5.4.2), the Improved dual-failure RN-FCS ILP was not able to solve large-scale networks within reasonable process time. Thus, we decided to employ a different approach to solve the Reserved Network Fixed Charge plus Spare capacity problem. The main objective is to find a good but sub-optimal solution for the full dual failure restoration problem within a reasonable process time. We decided to decrease the complexity of the problem by changing the structure of the decision variables. In the dual-failure RN-FCS and Improved dual-failure RN-FCS

[7] The notation (end) refers to the last element of a vector. For instance: Vector(end)=100 means the last element of the Vector is equal to 100.

models, the integer decision variables are the amounts of spare capacity on every span. We introduce a new model where the decision variables are the amounts of restoration flows assigned to pre-enumerated backup routes for the restoration of any dual failure scenario.

We assumed that the network and a set of routed working traffic are given, thus, there are working capacities assigned to every span in the network. The network's topology may or may not be three-connected, as it may have not been designed for dual-failure restorability.

5.2.3.1 Notations

In this section, we introduce all of the elements of the presented ILP model with their descriptions:

Sets:

S_{all}: the set of all of the possible spans within the network.

$S_{existing} \subseteq S_{all}$: the set of existing spans in the network.

$S_{absent} \subseteq S_{all}$: the set of absent spans in the network.

BR_i : the set of eligible backup routes for the restoration of span i.

Parameters:

C_i : the cost of one unit of spare capacity on span i.

F_i: the cost of installing span i in the network.

W_i : the number of working capacity on span i.

M: a very large positive number.

Decision Variable:

s_i: the total number of spare capacities on span i.

$f_{br,i}^{i,j}$: the amount of flow routed on backup route br for the restoration of span i upon failure of spans i and j.

$\delta_{br}^{i,j}$: a binary variable that equals 1 if backup route br for the restoration of span i passes through span j.

$\hat{\delta}_i$: a binary decision variable that equals 1 if the span i from the set S_{absent} has been assigned any spare capacity for restoration and 0 otherwise.

5.2.3.2 Pre-enumerated Backup-Route Based Mathematical Model for the Dual Failure Span Restoration Problem (arc-path dual-failure RN-FCS)

As already stated, although the proposed dual-failure RN-FCS ILP model guarantees the optimal solution of the dual failure span restorable network design problem, its computational complexity (i.e., NP-hardness) [89], makes it intractable for large-scale network problems. The runtimes for various network sizes have been tabulated in Tables 5.2 and 5.3. Thus, we proposed an exact solution approach based on the ILP model in [44] for designing large-scale dual failure restorable networks able to find an optimal or sub-optimal solution within a reasonable computational memory and time. The ILP model of this approach is as follows in Equations (22) through (28). In this ILP, the objective is to minimize the total cost of the spare capacity allocation plus the cost of

installing new spans subject to eight sets of constraints to guarantee the full restorability of the network in the event of any dual failure scenario.

5.2.3.3 The General Algebraic ILP Model

$$Minimize\ Total\ cost = \sum_{i \in S_{existing}} (C_i \times S_i) + \sum_{i \in S_{absent}} (C_i \times S_i + F_i \times \hat{\delta}_i) \tag{22}$$

Subject to:

$$\sum_{br \in BR_i} f_{br,i}^{i,j} \geq W_i \quad \forall i \in S_{existing}, \forall j \in S_{existing} \mid i \neq j \tag{23}$$

$$\sum_{br \in BR_j} f_{\mathrm{br,j}}^{i,j} \geq W_j \quad \forall i \in S_{existing}, \forall j \in S_{existing} \mid i \neq j \tag{24}$$

$$s_k \geq \sum_{br \in BR_i} (f_{br,i}^{i,j} \times \delta_{br}^{i,k}) + \sum_{br \in BR_j} (f_{br,j}^{i,j} \times \delta_{br}^{j,k}) \tag{25}$$

$$\forall i \in S_{existing}, \forall j \in S_{existing}, \forall \mathrm{k} \in S_{existing} \mid i \neq j \neq k$$

$$f_{br,i}^{i,j} \leq M \cdot (1 - \delta_{br}^{i,j}) \quad \forall i \in S_{existing}, \forall j \in S_{existing}, br \in BR_i \mid i \neq i \tag{26}$$

$$f_{br,j}^{i,j} \leq M \cdot (1 - \delta_{br}^{j,i}) \quad \forall i \in S_{existing}, \forall j \in S_{existing}, br \in BR_j \mid i \neq i \tag{27}$$

$$s_k \leq \hat{\delta}_k \times M \quad \forall k \in S_{absent} \tag{28}$$

The network under study may not be three-connected, hence not dual-failure restorable. In that case, the network may need additional span(s) to be established to accommodate the required connection for allocating spare capacities. Since the

enumerated backup routes consist of spans from both $S_{existing}$ and S_{absent}, there might be spans along some of the enumerated backup routes that are a member of the set S_{absent}. If a selected backup route has one or more of such spans along, those spans will be established in the network and an installation cost (F) is added to the total cost.

Considering a very large constant *M*, constraints (26) and (27) select eligible backup routes for the restoration process of every dual failure scenario. Constraint set (28) ensures that if there is an absent span in the network with a non-zero spare capacity assigned to it, it will be considered for the installation cost in the objective function.

The efficiency of the proposed ILP model can be analyzed concerning the optimality gap of the final solution and the duration of the computational time using available CPU and RAM. Considering the NP-hardness of the problem, the dual-failure RN-FCS ILP model cannot be solved as efficiently as needed for even medium-size networks. On the other hand, the introduced ILP finds a sub-optimal solution in a reasonable computational time. Also, the introduced approach provides the operator with the opportunity to select elite spans for the set of backup routes. Being able to manually select specific spans as a part of specific routes for transferring the traffic, allows the network operator to tailor the network to suit best the dynamic conditions and can be of crucial importance [104]. Thus, any alternative according to the present conditions and limitations of the network can be easily implemented in planning the spare capacity allocation.

5.2.3.4 Backup Route Generation (Enumeration)

To provide full dual failure restorability, the network needs to be three-connected; however, the input network's topology may not be three or even bi-connected. In other words, in the network under study, there may not be three disjoint routes between the end nodes of every span with working traffic. To overcome such connectivity deficiencies in the network, we need to establish new spans. The process of new span establishment is done by employing backup routes that may consist of spans from both $S_{existing}$ and S_{absent}. Therefore, if there is a span(s) along the selected backup routes that belong to the set of absent spans, it will be established in the network and contribute to the connectivity of the network topology.

We are inspired by Suurballe's and Dijkstra's algorithms [41], [141], and [142] to find the set of shortest disjoint backup routes between the end nodes of every span in the set $S_{existing}$. In our study, the span installation cost and the cost of carrying spare capacity on every span are correlated with the length of the span. Therefore, by referring to the shortest route, we mean the lowest cost route. In Suurballe's algorithm, first by employing Dijkstra's algorithm, we find the shortest route between the end nodes of the first span. This route is the first backup route. Second, to find a disjoint backup route from the first backup route, we eliminate all of the spans of the first backup route from the set of all of the spans. Then using Dijkstra's algorithm, we find the second shortest backup route that is disjoint from the first one. Up to this point, we find a pair of shortest disjoint backup routes between the end nodes of the first span in $S_{existing}$. We repeat this process for all the other spans in $S_{existing}$ one by one, to find the first pair of disjoint backup routes for every span in the set $S_{existing}$. However, only the shortest pair

of disjoint backup routes is sufficient enough to find a feasible solution to the problem; it is not necessarily a good solution. To grow the pool of backup routes and provide a diverse set of backup routes to choose from, we enumerate several disjoint backup route pairs for the restoration problem to choose from.

To find more pairs of disjoint backup routes between the end nodes of the spans, we use the original order of spans in $S_{existing}$ to proceed. Before generating the second pair of disjoint backup routes, we update the cost matrix in Dijkstra's algorithm. Updating the cost matrix in Dijkstra's algorithm is equivalent to removing the span installation cost (F) of the spans used in the first pair of backup routes from the total cost of those spans. Afterwards, for finding the next pairs of disjoint backup routes, considering the first span in $S_{existing}$, we eliminate one span from its first and second backup routes and then using Dijkstra's algorithm we find the third backup route. For the fourth backup route to be fully disjoint from the third one and also be distinct from the first and second one, we eliminate all of the spans used in the third backup route and one span from the first and one from the second backup routes. In this algorithm whenever we find a pair of backup routes for all of the spans in $S_{existing}$, the cost matrix is updated to count for the newly-used spans from set S_{absent}. Thus, the order of the spans in $S_{existing}$ will not affect the routing process. In other words, this method of updating the cost matrix causes the algorithm to become independent of the order of spans in $S_{existing}$. This independence of the order of members in $S_{existing}$ reduces the complexity of the solution process. The pseudocode of the backup route generation algorithm is shown in Algorithm 5.

Algorithm 5. Disjoint backup route generation algorithm

Inputs

A: the set of end-nodes of every span in S_{all}.

$S_{existing}$: the set of existing spans in the network that carry working traffic.

PBR: number of backup route pairs for every span in $S_{existing}$.

C: the set of unit costs of carrying traffic on every span in S_{all}.

F: the set of installation costs of every span in S_{all}.

F_temp: a temporary set of installation costs for every span in S_{all}.

Dijkstra algorithm [141]

Output

BR: the set of the entire generated backup route pairs for every span in $S_{existing}$.

Finding disjoint backup routes:

1. For *ii* from 1 to *PBR* {
    2. For *i* from 1 to length ($S_{existing}$) {
        F_temp ←*F*
        Origin=A ($S_{existing\ i}$, 1)
        Destination=A ($S_{existing\ i}$, 2)
        k=1
        3. While $k \leq 2${
            If *ii* == 1 AND *k* ==1{
                BR (*i*, 1) ← Dijkstra (*F*, *C*, *Origin*, *Destination*) }
            Else {
                If length (BR_i) == odd {
                    4. For *j* in BR_i (*end*) {
                        $F\_temp_j = \infty$}
                    BR_i *(end+1)* ← Dijkstra (*F_temp*, *C*, *Origin*, *Destination*) }
                Else {
                    5. For *j* from 1 to length (BR_i){
                        alpha = a random element from $BR_i(j)$
                        F_temp (spans in *alpha*) = ∞
                        }
                    BR_i (*end+1*) ← Dijkstra (*F_temp*, *C*, *Origin*, *Destination*)}}}}
        6. For *i* from 1 to length ($S_{existing}$) {
            7. For *j* in BR_i (1) {

F_j =0}
8. For j in BR_i (2) {
F_j =0} }

Having control over the selection of the backup routes provides a desirable level of flexibility in the network design process [104]. Generating the backup routes can be modified as much as the operator advises. For instance, based on the circumstances, some spans might be preferable to be used for sparing, or the cost associated with some other spans may dynamically change over time. In those scenarios, the operator can easily substitute one or several backup routes with the preferable ones. Also, in some applications, the increase in the number of hops in the backup routes can degrade the quality of the rerouted signal [50] and [128]. Thus, there could be a hop limit constraint enforced to the backup route generation process in the presented approach to meet the client's needs.

5.2.3.5 Experimental Setup

The analyzed test cases cover a variety of network sizes as small as 20 nodes up to 200 nodes. The networks' specifications have been detailed in Table 5.1. The arrangement of the nodes in a network has been selected randomly and the unit cost of spare capacity (C) for each span has been calculated as the Euclidean distance between the end nodes of the span. The cost of span establishment (F) is considered as a factor of (C), and various F/C ratios between 20 and 500 have been considered in [106]. In this study, the F/C ratio is set at 100 in all of the test cases.

Table 5.1. The specifications of the test cases

Test case	$\|N\|$	$\| S_{all} \|$	$\| S_{existing} \|$
1	20	190	27
2	40	780	49
3	80	3160	85
4	120	7140	133
5	140	9730	152
6	200	19900	227

All the experiments were run on a Windows Server 2012 R2- 128 GB RAM- Intel® Xeon® CPU E5-2650 v3 @ 2.30GHz (2 processors). All of the models were implemented in AMPL (20150327) [59] and we used Gurobi 8.1.0 [60] as the solver that used 32 parallel threads. The algorithm for generating backup routes was implemented in MATLAB R2020a [143]. The followings are the results of the proposed ILP-based heuristic approach compared to the dual-failure RN-FCS and Improved dual-failure RN-FCS as benchmarks for dual failure span restorable large-scale network design problem.

5.3 Results & Discussions

5.3.1 Experimental Results

The obtained results from the three discussed models have been presented in the following subsection of this section.

5.3.1.1 Dual-failure RN-FCS ILP Results

According to the results tabulated in Table 5.2, it can be seen that the presented dual-failure RN-FCS ILP is quite efficient when used to solve small test cases. However, its performance in terms of the processing time and optimality of the solution decreases drastically with an increase in the network size. Although the main contribution of this work is focused on the large-scale networks' restorability, we report the results for small networks as well to provide an exact point of reference for our comparisons and validate our results in small test cases.

Table 5.2. The total cost of Fixed-Charge Sparing using dual-failure RN-FCS ILP model for full dual failure restoration

Test Case	Network Size		Network Cost ($\times 10^6$)	Process Time (hours)
	$\|S_{all}\|$	$\|N\|$		
1	190	20	4.5	0.26
2	780	40	5.4	5.05
3	3160	80	N/A	72.0
4	7140	120	N/A	N/A
5	9730	140	N/A	N/A
6	19900	200	N/A	N/A

Our simulations showed that for networks with 80 nodes and more, the dual-failure RN-FCS is not able to find any solution after more than three days of running. Moreover, for the networks with 120 nodes and above, the used computer with 128 GB of RAM ran out of memory. Due to such difficulties, we implemented dual-failure RN-FCS with reduced search space to try and solve the problem for bigger network instances.

5.3.1.2 Dual-failure RN-FCS with Reduced Search Space Results

Using the Space Reduction algorithm, we managed to improve the performance of dual-failure RN-FCS ILP for larger networks. In the dual-failure RN-FCS model with reduced search space, unlike the dual-failure RN-FCS model where every span from the set S_{all} can be selected as a part of the restoration solution, a limited set of spans are eligible to be used as a part of the restoration solution. In other words, using Algorithm 4, by selecting only a valuable subset of the set S_{all}, we reduce the search space and increase the performance of the solution approach. We studied the effects of reducing 70%, 80%, and 90% of the possible spans, on the dual-failure RN-FCS solution performance. The total network costs and the process times for the reduced search spaces have been tabulated in Table (5.3). The search space reduction percentages have been mentioned at the top of each column in Table 5.3.

Table 5.3. The total cost of Fixed-Charge Sparing using dual-failure RN-FCS ILP model for full dual failure restoration, considering full and reduced search space

Test Case	Network Size	Search Space Reduction%							
		0%		70%		80%		90%	
	|N|	Network Cost ($\times 10^6$)	Process Time (hours)	Network Cost ($\times 10^6$)	Process Time (hours)	Network Cost ($\times 10^6$)	Process Time (hours)	Network Cost ($\times 10^6$)	Process Time (hours)
1	20	4.5	0.26	4.9	0.003	5.1	0.001	5.1	0.001
2	40	5.4	5.05	5.4	0.13	5.4	0.06	6.1	0.02
3	80	N/A	72.00	7.0	45.92	7.0	32.03	7.2	1.49
4	120	N/A	N/A	N/A	N/A	N/A	> 72.00	21.3	72.00
5	140	N/A	N/A	N/A	N/A	N/A	N/A	N/A	> 72.00
6	200	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

According to the obtained results, reducing the search space by removing the most expensive spans from the search space, resulted in a decrease in the processing time while it did not cause any significant increase in the total network cost. In other words, the existence of the most expensive spans in the networks does not play a significant role in finding the optimal solution, while it causes high levels of computational complexity. Therefore, eliminating those spans from the search space is beneficial for finding optimal solutions within less computational complexity. Using the dual-failure RN-FCS model with reduced search space, we were able to find a sub-optimal solution for networks with 80 and 120 nodes. However, the performance of dual-failure RN-FCS

started to decrease for larger networks. For a network with 140 nodes, even by reducing the search space by 90% and using 80% of the RAM, the dual-failure RN-FCS was not able to find any solution after almost 4 days of running.

5.3.1.3 Span Restoration using Backup Routes Results

Employing the proposed Backup Route-based ILP model (arc-path dual-failure RN-FCS) for designing the full dual failure span restorable network, we were able to find sub-optimal solutions for large-scale networks where even the dual-failure RN-FCS with reduced search space was not able to find any solution. Table 5.4 represents the obtained results showing the solution performance and total process times.

In every test case, we calculated the optimality gap of the Backup Route-based ILP model according to the best results obtained from the dual-failure RN-FCS and Improved dual-failure RN-FCS models. It can be seen that by increasing the number of backup route pairs from an average of 5 to 10 per span in $S_{existing}$, the objective function value improved by up to 22% for large-scale networks. The average of 5 and 10 backup routes per span in $S_{existing}$ were enumerated from a reduced search space where we removed 90% and 80% of the longest spans from the set S_{all}, respectively. Therefore, if for some of the spans, there were less than 5 or 10 pairs of backup routes found, the backup route generator compensated by finding more pairs of backup routes for other spans in the network. Thus, in Table 5.4, we indicated the number of backup routes as an average of 5 and 10 pairs per span.

For networks with 140 nodes and more, we were not able to find any solution using the dual-failure RN-FCS or Improved dual-failure RN-FCS models. However, the

proposed Backup Route-based ILP model was able to find sub-optimal solutions. The improvement in the objective function value using the proposed model was 11% and 17% for 140 and 200 node networks, respectively.

Table 5.4. The total cost of Fixed-Charge Sparing using the Backup Route-based ILP model for full dual failure restoration

Test Case	Network Size	Backup Route-based ILP				
		$\overline{5}$ PBR		$\overline{10}$ PBR		Optimality Gap Improvement %
	$\|N\|$	Process Time (hours)	Optimality Gap %	Process Time (hours)	Optimality Gap %	
1	20	0.0003	23.90	0.001	12.33	48.41
2	40	0.004	26.61	0.02	18.13	31.87
3	80	0.02	40.12	1.40	31.93	20.41
4	120	0.14	38.39	1.54	29.90	22.12
5	140	0.58	N/A	11.15	N/A	N/A
6	200	0.80	N/A	27.61	N/A	N/A

5.4 Conclusion

This chapter studied the large-scale dual-failure span restorable network design problem. The dual failure scenarios may not be as frequent as single failure scenarios; however, in vital network services, even a small chance of service outages can be catastrophic. Therefore, having a network that is immune to any dual failure scenario is important. As already stated, several ILP models were presented to solve the dual failure span restoration problem, however, they encounter memory limits and prohibitive

runtimes for the solution of large-scale networks. In this chapter, we employed a new ILP model based on pre-enumerated backup routes to design large-scale dual-failure restorable networks. The presented model can be applied to any network carrying working capacity, regardless of the network's connectivity (e.g., biconnected, three-connected, etc.). We introduced a heuristic-based algorithm for the pre-enumeration of elite backup routes for the restoration of network spans.

We applied the three ILP models (dual-failure RN-FCS, dual-failure RN-FCS with reduced search space, and the ILP model based on the proposed heuristic backup route enumeration) on seven test cases with various sizes from 20 up to 200-node networks. Our simulations showed the performance of the previously introduced models becomes quite poor when designing large-scale networks. In the case of networks with 140 nodes and more, the previously developed models using powerful solvers such as Gurobi, were not able to find any solution while the introduced model was able to find a sub-optimal solution within reasonable processing time. Moreover, in this approach, by being able to implement backup routes, the network operator benefits from improvising in various conditions. Having control over the restoration backup routes can be advantageous if there are specific constraints on the hop limits of the routes or if there are some preferable spans that must/ must not be used for restoration purposes. Another advantage of the presented model that helps with decreasing the complexity of the solution process, is the decision variable types. In the introduced model, the decision variable translates into choosing a backup route from the set of given backup routes, while in the dual-failure RN-FCS model, the decision variables have to build up the backup routes for every span as a part of the solution. This change in the structure

of the decision variables made the solution process easier and as a result, our model required up to 70% less computational memory compared to the state-of-the-art models.

Chapter 6 : GA-based Approach for Survivable Large-scale Network Design Problem

With the rapid global development of telecommunication networks, their performance has become important more than ever. One of the contributing factors to the network performance is the employed survivability mechanism. Networks are usually equipped with the ability to survive single failures by restoring the affected traffic. However, higher levels of network robustness require restoration against more complex failure scenarios. Higher restoration ability can be interpreted as the restoration of two or more simultaneous failures in the network. In this chapter, we focused on the span restoration mechanism as one of the most capacity-efficient restoration mechanisms and we proposed an approach for full restorability against any dual-failure scenario in a large-scale network using a Genetic Algorithm. For our approach to be applicable to any step of the network design process, we allowed for the installation of new spans if necessary, by considering a variable topology for the networks under study. We consider the total cost of the dual-failure restorable network design as the span installation cost and spare capacity placement cost. We validated our results with previously developed ILP models as benchmarks and achieved an optimality gap of 28.89% ± 2.1 in large-scale network design costs in up to 96.95% ± 1.8 shorter processing times[8].

6.1 Introduction

In Chapter 2, we provided a detailed introduction to telecommunication networks. These networks provide the infrastructure for so many critical systems such as the health care

[8] Portions of this chapter along with parts of Chapter 2 are slated for publication as a journal paper: S. Doostie, T. Nakashima-Paniagua, and J. Doucette, “A Novel Genetic Algorithm-based Approach for Design of Dual-Failure Span-Restorable Large-Scale Networks”, journal TBD.

system, rail and air traffic control system, and other IT systems [144]. We also mentioned that to establish networks with adequate levels of service quality, the network should be resilient to various kinds of failures. The failures may have various sources, causes, or frequencies; however, by employing proper protection and restoration mechanisms in the networks we can recover the network from failures with minimum cost and delay time in its operations [145]. In this chapter, we focused on the span restoration mechanism [146]–[148] and a detailed description of this mechanism has been described in Chapter 2. Span restoration mechanism provides efficient sharing of spare capacities [148] while it provides better control over the backup paths [134] and thus can have a better demonstration of our proposed heuristic-based approach.

A network can encounter various component failure scenarios such as span or node failures in various patterns. One of the most frequent failure scenarios in telecommunication networks is span failure scenarios which we can categorize into single and multiple span failures at a time. In this chapter, we focused on the dual failure scenarios meaning two spans may fail simultaneously in a network. Employing the span restoration mechanism, the failed span will be restored over the backup routes between its end nodes.

Studying dual failure scenarios, simultaneous failures of two spans have a more complex structure compared to single failures. As any two working spans (i.e., spans that carry working traffic) can be failed simultaneously, a network requires two sets of backup routes between the end nodes of both of the failed spans which none of the backup routes should pass through the other failed span. In another word, the network

should be three-connected with regard to every working span. Figure 6.1 illustrates the basic schematic of the dual-failure span restoration mechanism. Considering spans CE and BF are failing at the same time. In this case, neither the backup routes for span CE are eligible to pass through span BF, nor the backup routes of span BF are allowed to pass through span CE. For instance, the backup route BAF is an eligible backup route for failed span BF but the backup route BCEF is not.

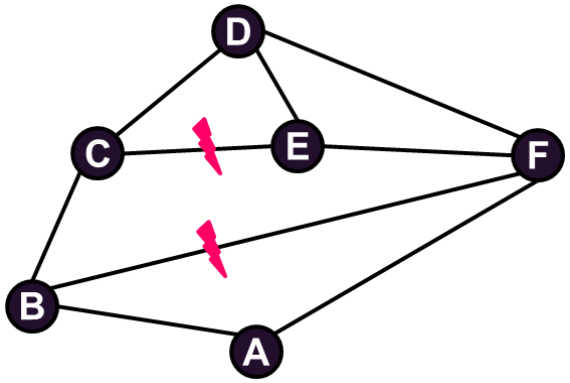


Figure 6.1. Basic Schematic of dual failure SR mechanism. Consider the failure of spans CE and BF are happening simultaneously.

Although the dual-failure scenarios are not as frequent as single failures, their occurrence rate increases with the increase in the network size [149]. There are three reasons that we believe the dual-failure restorability is of crucial importance for large-scale networks: (1) In large-scale networks, spans (e.g., optical fibers) based on their physical and technical specifications can transfer high volumes of traffic [150] and the more the amount of traffic on a span, the higher the probability of the span becoming failed. (2) In the event of single-span failure, if the repair and maintenance times are long [151], before fixing the failed span, another span in the network might fail thus; a simple single failure scenario can turn into a dual failure scenario. (3) Depending on how critical are the network services and how expensive (socially and financially) can

the service outages be, a network operator may prefer to have full restorability against any dual failure scenario at all times.

6.2 Related Work

As discussed earlier, the network design problem consists of the network topology design, routing, and spare capacity allocation problems. The authors in [33] and [81] studied the complete problem of mesh topology routing and sparing (MTRS) as a unified integer linear programming model [131]. Given the coordination of nodes in the network and the traffic demand matrix, the MTRS model is guaranteed to find the optimal solution to the complete problem of network topology design, routing, and spare capacity allocation. However, in terms of the processing time and the computational difficulty, it is not efficient even for medium-sized networks. Thus, they presented an ILP-based heuristic approach to divide this problem into three sub-problems and tackle each one of them separately. They introduced a 3-step ILP model that first solves the topology and routing problem that is being known as fixed charge plus routing (FCR) problem [32]. The result of the first step in [33] was a network with installed spans and allocated working capacities to accommodate the traffic demand matrix. In the second step, using the obtained topology and determining working routes from the first step, they solved the spare capacity allocation problem for restoration purposes. Thus, the network topology could be augmented by installing additional spans to make the network two-connected (i.e., physically restorable) and spare capacities were placed on the spans for restoring every single failure scenario happening for every working span. Finally, using the obtained topology from the second step, they solve the complete problem only at this step; they considered the installed spans from the first two steps

only as the possible spans for the network to reduce the complexity of the solution and search space. Moreover, the authors in [152] presented a 4-step ILP-based heuristic for the problem of multi-period survivable network design augmentation in which they focused on the SBPP mechanism.

Herein, we studied the full survivability of large-scale networks while minimizing network cost by minimizing the required spare capacity placement and new span installation. The authors in [148] Introduced an ILP model for minimizing the required spare capacity for span restorable networks considering single failure scenarios and various spectrum conversion schemes. The authors in [153] presented ILP models for the full and partial dual-failure restorability of networks, considering fixed topologies. They performed an extensive analysis of the partial dual-failure restoration and observed that partial restorability requires significantly lower levels of spare capacity.

Inspired by the research work presented in [33], we recently introduced two different ILP models for the spare capacity allocation problem for full restorability against dual-failure scenarios, considering the working traffic has been routed and the network topology can be augmented to satisfy the restoration requirements in Chapter 5.

The authors in [154] presented an iterative heuristic approach for a survivable network design problem in which they employed tabu search to find disjoint routes between the end nodes of the traffic demands and assign relays on their optimal position along the found paths.

In this chapter, we propose a heuristic approach for the problem of dual failure restoration of large-scale networks and we design a genetic algorithm (GA) [155] to

solve the dual failure restorable large-scale network design equipped with a span restoration mechanism. We select GA as our approach because of its higher performance proficiency and flexibility compared to other evolutionary approaches [113], [156]. Among all of the well-known evolutionary algorithms, GA has been noticed extensively because of its ability for robust search. The details of the genetic algorithm are mentioned in Chapter 2.

In this chapter, we introduce a heuristic approach based on a GA for designing dual-failure span-restorable large-scale networks considering variable network topology.

6.3 Dual-failure RN-FCS ILP Model

The previously presented ILP model named reserved network fixed charge plus spare capacity problem for the restoration of dual failures (dual-failure RN-FCS) in Chapter 5 determines the optimal spare capacity placement on the network while minimizing the total cost of spare capacity placements and new span installations. The complexity of this problem is NP-hard, thus specifically for large-scale networks, it may not be efficient. Employing the available resources, this ILP encounters memory limits and prohibitive runtimes for designing networks with 80 nodes and more. In Chapter 5, we presented an improved version of the dual-failure RN-FCS ILP model to mitigate the computational complexities by a trade-off between the optimality and computational complexity of the solution. We also introduced another ILP model based on the pre-enumeration of backup routes for dual failure span restorable network design problem, while the topology can be dynamically changed. That model can find a sub-optimal solution very fast with small computational infrastructure requirements. However, the pre-enumeration of backup routes of the restorable spans limits the solution to only the

found backup routes. Thus, there will be some parts of the search space that will not be searched for possibly good backup routes and the global optimum solution may not be achieved.

6.3.1 The Mathematical Formulation of the Problem

Dual-failure RN-FCS aims to minimize the total cost of spare capacity allocation and new span installation subject to the set of constraints that guarantee the full restorability against any dual failure scenario among working spans. Following is the description of the sets, parameters, and decision variables used in the ILP model based on Chapter 5. Equations (29) through (38) are adopted from Chapter 5.

Sets:

N: the set of the fixed nodes of the network.

S_{all}: the set of all of the spans that are or can be installed in the network.

$S_{existing}$: the set of installed spans that carry working traffic and will be considered for restoration.

S_{absent}: the set of $S_{all} - S_{existing}$ (possible spans for new installments)

Parameters:

C_{ij}: the cost of spare capacity placement on the span from node i *to* j.

F_{ij}: the cost of span establishment from node i *to* j.

W_{ij}: the amount of working traffic on span from node i *to* j.

M: a big positive number.

Decision Variables:

$s_{ij,mn}^{kp}$: the amount of spare capacity allocated to the span from node k *to* p when spans from node i *to* j and m *to* n are failed.

s_{ij}: the accumulated spare capacity allocated on span from node i *to* j such that it is enough to support all of the dual failure scenarios.

$\hat{\delta}_{ij}$: the binary variable that equals 1 if the span from node i *to* j from the set S_{absent} was installed for spare capacity placement, and zero otherwise.

$$Minimize\ Total\ cost = \sum_{\forall ij\ \in S_{existing}} (C_{ij} \times s_{ij}) + \sum_{\forall ij \in S_{absent}} (C_{ij} \times s_{ij}\ +\ F_{ij} \times \hat{\delta}_{ij}) \quad (29)$$

Subject to:

$$\sum_{\forall ik \in S_{all} |\ ij \neq\ mn \neq ik} s_{ij,mn}^{ik} = W_{ij} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (30)$$

$$\sum_{\forall mk \in S_{all} |\ ij \neq\ mn \neq mk} s_{ij,mn}^{mk} = W_{mn} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (31)$$

$$\sum_{\forall kj \in S_{all} |\ ij \neq mn \neq kj} s_{ij,mn}^{kj} = W_{ij} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (32)$$

$$\sum_{\forall kn \in S_{all} |\ ij \neq\ mn \neq kn} s_{ij,mn}^{kn} = W_{mn} \qquad \forall ij \in S_{existing}, \forall mn \in S_{existing} \quad (33)$$

$$\sum_{\forall pk \in S_{all} |\ k \notin \{i,j,m,n\}} s_{ij,mn}^{pk} - \sum_{\forall kp \in S_{all} |\ k \notin \{i,j,m,n\}} s_{ij,mn}^{kp} = 0$$

$$\forall ij \in S_{existing}, \forall mn \in S_{existing}, \forall p \notin \{i, j, m, n\}$$

(34)

$$s_{kl} \geq s_{ij,mn}^{kl} \quad \forall ij \in S_{existing}, \forall mn \in S_{existing}, \forall kl \in S_{all} | ij \neq mn \neq kl \tag{35}$$

$$s_{ij} \leq \hat{\delta}_{ij} \times M \quad \forall ij \in S_{all} \tag{36}$$

$$\sum_{\forall ij \in S_{absent}} \hat{\delta}_{ij} + \left| S_{existing} \right| \geq |N| \tag{37}$$

$$\sum_{\forall j \in N | j \neq i} \hat{\delta}_{ij} \geq 2 \quad \forall i \in N \tag{38}$$

As discussed in Chapter 5, the dual-failure RN-FCS's objective (Equation (29)) is to minimize the total cost of spare capacity allocation and possibly-required span establishment for full restoration against any dual failure scenario. The constraints from equations (30) to (33) guarantee enough spare capacity placement on the spans originated from and terminated at any origin or destination node of failed spans in every dual failure scenario. Equation (34) is the mathematical representation of the transhipment constraint for the intermediate nodes. Equation (35) guarantee the allocation of enough spare capacity on every span, considering all of the dual failure restoration capacities. Equation (36) tunes the $\hat{\delta}_{ij}$ values based on the existence of spare capacity on the spans from the set S_{absent}. Equations (37) and (38) are additional constraints that only describe the topology and environment of the network.

6.4 Proposed Genetic Algorithm for Survivable Network Design

In this chapter, we propose a GA based on the main characteristics of the dual-failure RN-FCS ILP model. Unlike the introduced model in Chapter 5, the proposed GA herein is not bound to a limited set of pre-enumerated backup routes and it can explore the search space to find various backup routes for the restoration of any dual failure scenario in the network. As an input to this problem, we have a set comprising of all of the dual-failure scenarios that we want to restore one by one and allocate a set of spare capacities on the network to restore the working traffic upon simultaneous failure of two working spans. In this context, the order of restoration of the dual failure scenarios is important. Let's say we have a network of five nodes and six working spans as shown with solid lines in Figure 6.2(a). Let us consider the simultaneous failure of spans AB and AE as the first dual-failure scenario to analyze. Since node A will become disconnected from the network, the network needs at least another span to be joined to node A. Thus, we assume the span AC has been selected to be installed in the network for the restoration of spans AB and AE. Taking into account the newly installed span AC, we can use backup routes ACB and ACE for the restoration of spans AB and AE respectively, as illustrated in Figure 6.2(b). In the next step, let's consider the next possible dual-failure scenario as the simultaneous failure of spans CB and CE. Considering the spare capacities that had been placed for the previous dual-failure scenario, we have some spare capacities already allocated to spans CB, AC, and CE. Thus, upon failure of spans CB and CE, the previously placed spare capacity on span AC can be utilized without additional costs. So, the backup routes for the restoration of spans CB and CE could be CAB and CAE, respectively as illustrated in Figure 6.2(c).

Now consider the situation where first we analyze the failure of spans CB and CE as illustrated in Figure 6.2(d). In this case, the network's topology is connected and there is no need for a new span installation. The backup routes for the restoration of spans CB and CE could be CDEAB and CDE, respectively where there is enough spare capacity placed on them. Now, let us consider the second dual-failure scenario is the simultaneous failure of spans AB and AE. This dual-failure scenario makes the network disconnected, thus requiring a new span installation. Assuming we installed span AD, by considering the previously allocated spare capacities on the network (on spans CD, DE, EA, and AB) and assigning enough spare capacity to span BC, we can restore spans AB and AE through backup routes ADCB and ADE, respectively as illustrated in Figure 6.2(e). One can see the difference between the final protected networks in figures 6.2(c) and 6.2(e) where we considered only two dual-failure scenarios (AB, AE) and (CB, CE) in different orders. Thus, analyzing every dual-failure scenario will change the available spare capacities in the network, which can change the future decisions for the restoration of the upcoming dual-failure scenarios. Here, we showed the difference that the order of analyzing the dual-failure scenarios can make in the final designed network. In the presented GA, we benefited from the iterative aspect of the algorithm to evaluate a vast variety of possible combinations of the dual-failure scenario orders in a network design problem. The details of our GA structure and designed genetic operators have been explained in the next section.

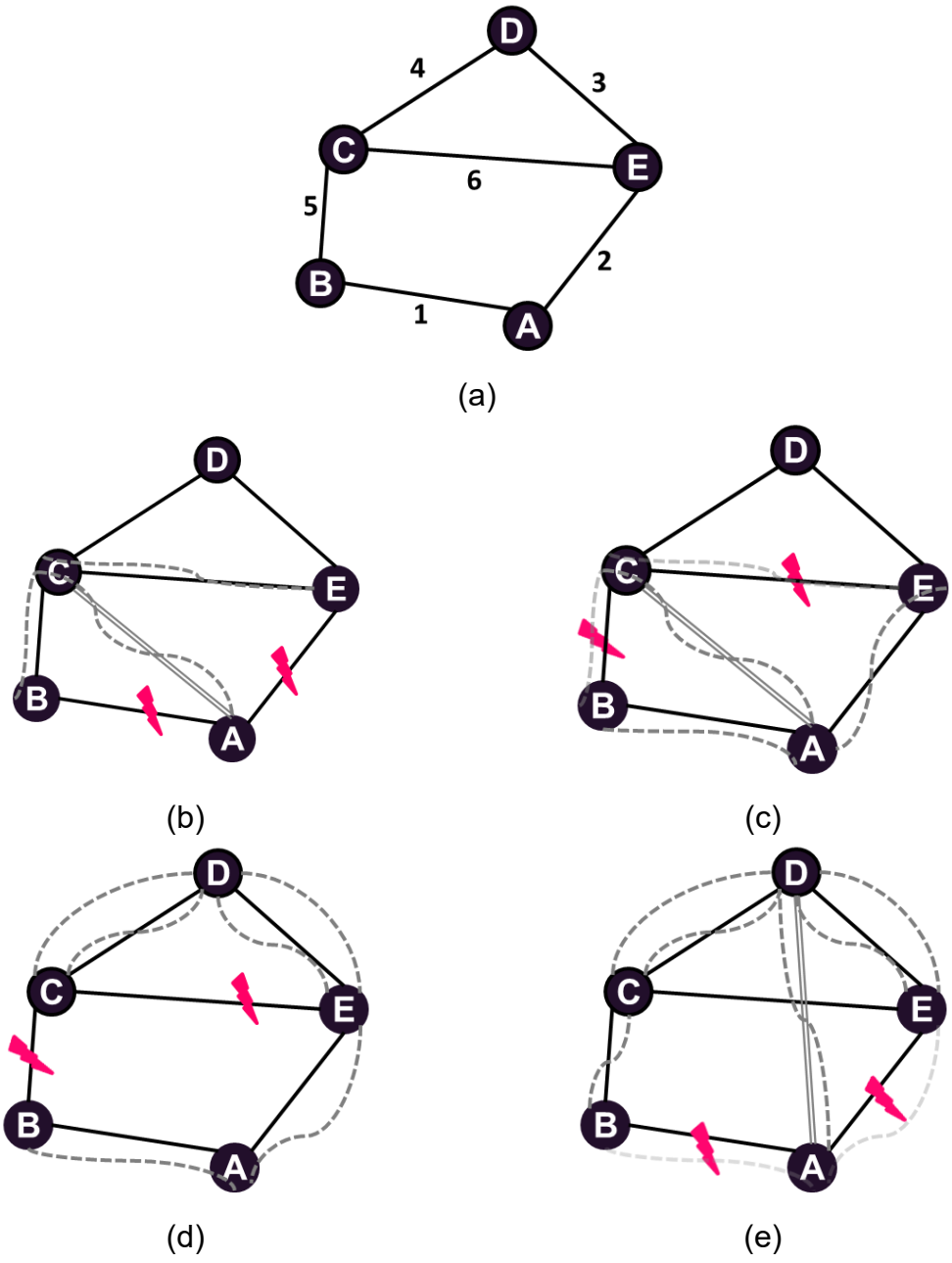
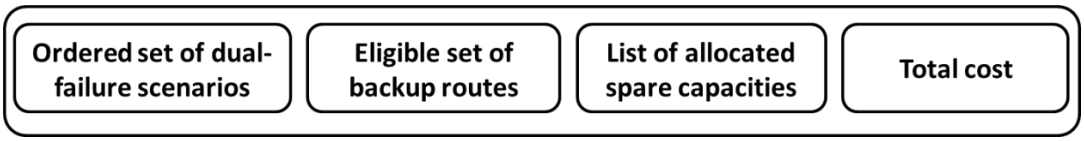


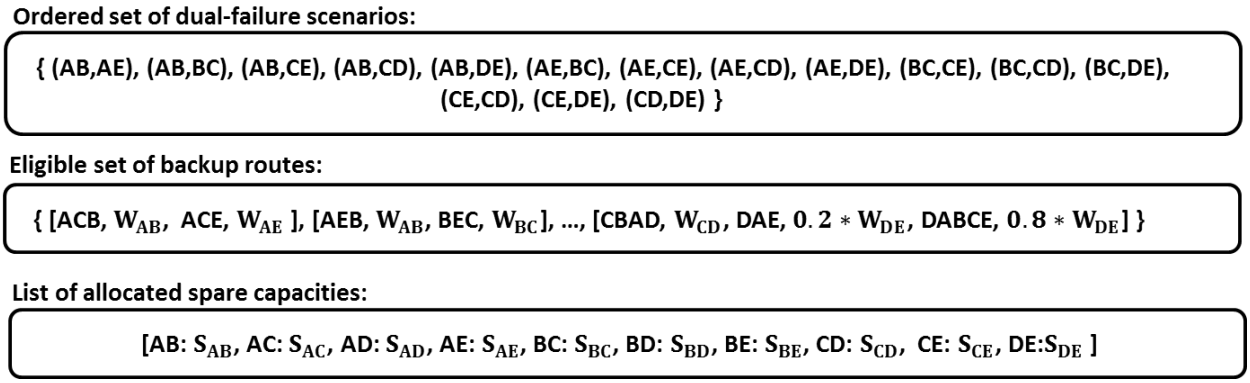
Figure 6.2. The effect of the order of the dual-failure scenarios to be evaluated on a network with five nodes and six working spans: considering dual-failure scenarios (AB, AE) and (CB, CE) based on the network in (a). In (b) first (AB, AE) was evaluated, thus AC was installed. Then in (c), (BC, CE) was evaluated. In (d), first (CB, CE) was evaluated then in (e), (AB, AE) was evaluated.

6.4.1 The Building Blocks of Survivable Network Design Genetic Algorithm

We studied the problem of full dual failure restorable network design in which the solution sought is a minimum cost connected network topology with a set of spare capacities that will restore all the dual-failure scenarios that may happen. The structure of the chromosome in our GA comprises one main gene to which we apply the genetic operators and three calculating genes. The main section contains the ordered set of all of the dual-failure scenarios and the calculating sections include: (1) the backup routes for restoring the working traffic on every span in every dual-failure scenario, (2) the total amount of spare capacity assigned to the spans, and (3) the total cost of spare capacity allocation in addition to the possible cost of new span installation. Figure 6.3(a) demonstrates the general structure of the presented chromosome. Considering the network in Figure 6.2, the details of the proposed chromosome structure have been demonstrated in Figure 6.3(b).



(a)



(b)

Figure 6.3. (a) The general structure of the chromosome of the proposed GA; (b) the detailed structure of genes of a chromosome based on the network in Figure 6.2.

To generate the initial population, we built the initial chromosomes with a different ordered set of dual-failure scenarios to generate various feasible solutions to the problem. The size of the ordered set of all possible dual-failure scenarios can be calculated by $|S_{existing}| \times (|S_{existing}| - 1) / 2$. Therefore, for a network with only 100 working spans, the total number of dual-failure scenarios would be 4950, meaning there would be 4950 various pairs of spans to be considered for dual-failure restorability. Considering a dual-failure scenario as a tuple of two working spans being failed simultaneously, we included a variety of dual-failure scenario orders in our initial population such as: (1) first we paired the first span in $S_{existing}$ with all the other upcoming spans in that set, then we tried the same matching with the second span, and so on. The schematic of this ordered dual-failure scenario has been depicted in Figure 6.4(a). (2) We paired every span in $S_{existing}$, with its consecutive span that comes after the first one. In doing so, we skipped some dual-failure scenarios, so we inserted the

missing dual-failure scenarios at the end of this ordered dual-failure scenario (Figure 6.4(b)). Also, we generated another ordered dual-failure scenario using this procedure with a shuffled $S_{existing}$. (3) We generated an ordered set of dual-failure scenarios using the inverse procedure in (2), so the first dual-failure scenarios would include the last spans in $S_{existing}$ (Figure 6.4(c)). The other ordered dual-failure scenarios have been generated using uniform random sampling from the first chromosome (i.e., a uniform random sample from the first ordered set of all of the dual-failure scenarios). In Figures 6.4(a) to 6.4(c), for simplicity we used numbers from Figure 6.2 to refer to the existing spans, thus the set $S_{existing} = \{1, 2, 3, 4, 5, 6\}$. Moreover, we reordered one of the randomly selected ordered dual-failure scenarios according to the total working traffic of its spans. In another word, we ordered the dual-failure scenarios descending based on the total working traffics of the two working spans in every dual-failure scenario so that the first dual-failure scenario would have the highest total working traffic to be restored and the last dual-failure scenario would have the least.

Ordered set of dual-failure scenarios = {(1,2),(1,3),(1,4),(1,5),(1,6),(2,3),(2,4),(2,5),(2,6),
(3,4),(3,5),(3,6),(4,5),(4,6),(5,6)}

(a)

Ordered set of dual-failure scenarios = {(1,2),(2,3),(3,4),(4,5),(5,6),(1,3),(1,4),(1,5),(1,6),
(2,4),(2,5),(2,6),(3,5),(3,6),(4,6)}

(b)

Ordered set of dual-failure scenarios = {(5,6),(4,5),(3,4),(2,3),(1,2),(1,3),(1,4),(1,5),(1,6),
(2,4),(2,5),(2,6),(3,5),(3,6),(4,6)}

(c)

Figure 6.4. (a) Represents the first method for generating an ordered set of dual-failure scenarios; (b) depicts the second method for generating an ordered set of dual-failure scenarios; (c) shows the third method that is the inverse of the second method.

6.4.1.1 *HeuristicRouting* Function

Every dual-failure scenario consists of two working spans that must be restored upon their simultaneous failure through other spans. The best set of backup routes for every span would be those that incur less cost for designing the network. Notably, the total cost of every backup route is defined as the cost of the spare capacity allocation on backup route spans plus the cost of new span installation (where applicable). To find the best set of backup routes for both spans in all of the dual-failure scenarios, we introduce a novel *HeuristicRouting* function. This function defines how many/which backup routes must be used to restore a failed span as well as how much of the required spare capacity should be assigned to each of the selected backup routes.

Before describing the *HeuristicRouting* function, we make several observations: (1) The spare capacities that have been placed on one backup route for restoring a failed span cannot be used for the restoration of the other span in the same dual-failure scenario, i.e., the network must be able to restore both spans in a dual-failure scenario at the same time. (2) The spare capacities that have been allocated for the restoration

of spans in every dual-failure scenario can be used for the restoration of spans in another dual-failure scenario, and once a new span is installed as a part of a backup route, the cost of installation for that span will be zero for other backup routes. (3) The developed *HeuristicRouting* function analyzes one dual-failure scenario at a time and always the first span in the dual-failure scenario before the second one.

To calculate the second section of every chromosome (i.e., the eligible set of backup routes and the number of the spare capacity assigned to each route), we applied the *HeuristicRouting* function to all of the dual-failure scenarios one by one. The output of the *HeuristicRouting* function is the set of best backup routes and the number of spare capacities allocated to their spans for full dual failure restoration of the network.

As the selected backup routes for different dual-failure scenarios can share their allocated spare capacities, the order of analyzing (1) dual-failure scenarios and (2) spans in a particular dual-failure scenario can change the best backup route for the next dual-failure scenarios, i.e., the order of the dual-failure scenarios and the order of the spans in every dual-failure scenario can change the output of the *HeuristicRouting* function. Notably, in a network where $|S_{existing}|$ spans carry working traffic, the number of dual-failure scenarios is $|S_{existing}| \times (|S_{existing}| - 1) / 2$. Thus, the length of the ordered set of dual-failure scenarios in a chromosome can be in the order of thousands for large-scale networks. However, after evaluating some dual-failure scenarios, the allocated spare capacity will be enough for the restoration of the upcoming dual-failure scenarios, and as a result, the effects of the order of the upcoming dual-failure scenarios on the output of the *HeuristicRouting* function will be negligible. This is why genetic operators

mostly were focused on the early parts of the chromosomes and generated new chromosomes by changing the orders of the early dual-failure scenarios.

6.4.1.1.1 *HeuristicRouting* Function Building Blocks

The *HeuristicRouting* function evaluates dual-failure scenarios as the first section of a chromosome one by one. In this function, we employed Dijkstra's algorithm [41] and [37], to find the least-cost path between the end nodes of a failed span. The cost matrix for Dijkstra's algorithm was obtained as new span installation cost plus spare capacity allocation cost for every span based on the amount of restorable traffic, i.e., the cost matrix was calculated as $F_{ij} + C_{ij} \times WT_{mn}$ for every span ij in the network and every span mn with restorable working traffic WT_{mn}. Also, as each new dual-failure scenario can use the allocated spare capacities from the previously evaluated dual-failure scenarios, we designed a *CostReduction* function (presented in Algorithm 6) to reduce the routing cost of spans with installed spare capacity proportional to the amount of the installed spare capacity. This feature allows for sharing the spare capacity among the backup routes of various dual-failure scenarios which we know will not occur simultaneously.

Algorithm 6. The pseudocode of the *CostReduction* function.

Inputs:

WT: working traffic to be restored

AvailableS: the set of spans with their previously allocated spare capacities from the previous dual-failure scenarios

CostD: the set of cost coefficients for Dijkstra's algorithm

C: the set of spare capacity placements' cost

Output:

CostD: the updated list of cost coefficients for Dijkstra's algorithm

CostReduction:

1. for i from 1 to length (*AvailableS*) {
2. if $AvailableS_i \geq WT$
3. $CostD_i = 0$
4. else
5. $CostD_i = CostD_i - AvailableS_i \times C_i$ }

While the *CostReduction* encourages the new backup routes to use the previously allocated spare capacities, there could be cases where the amount of working traffic that requires restoration, is greater than the available spare capacities on the spans of the found backup route. Thus, we designed the *MustUseSpare* function (presented in Algorithm (7)) that finds as many backup routes as possible for the restoration of the failed working traffic using only the previously allocated spare capacities, i.e., the *MustUseSpare* function finds backup routes with zero cost for the restoration of the current failed working traffic, if available.

To this end, the *MustUseSpare* function (Algorithm 7) first, sets the costs of spans with zero allocated spare capacity to "infinity." This prevents Dijkstra's algorithm to find a backup route that has even one span with zero spare capacity on it. Second, the costs of spans with a non-zero spare capacity are updated using the *CostUpdate*

function (embedded in Algorithm 7) such that the total cost of routing the failed working traffic over these spans is reduced proportionally to the amount of available spare capacity on them. Then, Dijkstra's algorithm tries to find a backup route for the restoration of all or parts of the failed working traffic, depending on the availability of the spare capacity. To achieve this, first, the minimum amount of spare capacity available on the spans of the found backup route is assigned to a variable named *TValue*, i.e., *TValue* represents the proportion of working traffic that can be routed free of cost over the found backup route using only the previously allocated spare capacities. If *TValue* is greater than or equal to working traffic, that means the working traffic is fully restored with the found backup route, if *TValue* is less than working traffic, means that only a portion of the working traffic is restored. In the latter case, we update the amount of available spare capacities and repeat this process for the remaining working traffic that has not been restored yet (i.e., initial working traffic minus *TValue*).

As the order of restoring spans in a dual-failure scenario could change the total spare capacity allocation, we used different protocols to improve the routing process. After routing the spans in a dual-failure scenario, there could be two scenarios: (1) if the summation of the remainder of working traffics (WT_{11} and WT_{12} for the first and second spans, respectively) is zero or only the WT_{12} is equal to zero, which means we used up all the available spare capacities in the network and there is no room for further improvement of the backup routes. Thus, the found backup routes and updated available spare capacities are finalized and we can proceed to the next dual-failure scenario; (2) if the WT_{12} is non-zero, we reverse the order of the routing, i.e., the second

span of the dual-failure scenario is evaluated first. The change of routing orders can improve the usage of the available spare capacities.

When the reverse routing order is evaluated, we compare the total remainder of working traffic in both routing orders. Then, one of the following sub-scenarios could happen: (2-1) the total remainder of working traffic in the reverse order (WT_{21} and WT_{22} for the second and first spans, respectively) is zero. In this sub-scenario, the found backup routes and updated available spare capacities using the reverse order are finalized and then we can proceed to the next dual-failure scenario; (2-2) WT_{21} is non-zero. In this sub-scenario, we compare the total remainder working traffics in both original and reverse orders of routing and select the order in which the total remainder working traffic is less; (2-3) the remainder of the working traffic of the second routed span in both original and reverse orders are non-zero, i.e., (WT_{11} = 0 and $WT_{12} \neq 0$) or (WT_{22}= 0 and $WT_{21} \neq 0$). In this sub-scenario, we further categorize the possibilities based on the number of shared spans between the two sets of backup routes of the two failed spans in a dual-failure scenario.

Algorithm 7. Pseudocode of the MustUseSpare and CostUpdate functions for analyzing the kth dual-failure scenario and jth span in that dual-failure scenario.

Input:

BR: the set of backup routes for all of the dual-failure scenarios

Sol_temp: the set of spans with the installed spare capacities during the evaluation of the previous dual-failure scenarios. These spare capacities are now available to be used for the restoration of the current dual-failure scenario's spans.

WT_j: the amount of working traffic on the j^{th} span in a dual-failure scenario to be restored

C: the set of spare capacity placements' cost

CostD: the set of cost coefficients for Dijkstra's algorithm

Output:

Route_temp: the set of temporary backup routes

***MustUseSpare* function:**

```
1. for j from 1 to 2 {
2.     flag = 0
3.     while flag == 0 {
4.         for i from 1 to length(Sol_temp) {
5.             if Sol_temp_i == 0
6.                 CostD_i = ∞
7.             else if Sol_temp_i ≤ WT_j
8.                 CostD_i = CostD_i – C_i × Sol_temp_i
9.         Route_temp←Dijkstra (CostD)
10.        if Route_temp == ∅
11.            flag = 1
12.        else
13.            TValue = minimum (spare capacities on the spans of Route_temp)
14.            if  Tvalue ≥ WT_j
15.                append(Route_temp) to BR_j with magnitude (WT_j)
16.                UpdateAvailableSpare
17.                WT_j = 0
18.                flag = 1
```

CostUpdate function-lines 4-8

19.	else
20.	append(*Route_temp*) to BR_j with magnitude (*TValue*)
21.	*UpdateAvailableSpare*
22.	$WT_j = WT_j - Tvalue$

First, we identify the set of shared spans between backup routes of both spans in a dual-failure scenario: *INTS1* represents the shared spans between the backup routes of the first and second routed spans in the original dual-failure scenario order. Similarly, *INTS2* represents the shared spans between the backup routes of the first and second routed spans in the reversed dual-failure scenario order. Our goal is to find backup routes with unique spans for each failed span in a dual-failure scenario, based on the fact that the used available spare capacities for building a backup route for a failed span cannot be used to build a backup route for the other failed span. We aim to reduce the chance of using up all the available spare capacities that could be used for finding backup routes for either of the failed spans toward building the backup routes for the first routed span only, while it can be restored through other available spare capacities that cannot be used for the second routed span. To do so, first, we selected the set between *INTS1* and *INTS2* that has the least number of members (i.e., there is less intersection between the found backup routes) and we named it *INTS*. For every span in the set INTS (i.e., S_{INTS}), we selected a backup route of the first-routed span that includes the span S_{INTS} and has the closest assigned capacity to the remainder of working traffic. Then, the selected backup route was replaced by a new backup route that does not include the shared span S_{INTS}. Algorithm 8 describes the high-level pseudocode of the proposed *HeuristicRouting* function. To increase the diversity and

randomness of the routing process, we employed a Random-Router heuristic algorithm that distributes the traffic over several routes.

Algorithm 8. Pseudocode of *HeuristicRouting* function

Input:

The ordered set of dual-failure scenarios in a chromosome

Output:

BR: the set of backup routes

RWT: the remainder of the working traffic

***HeuristicRouting* function:**

```
1. for k from 1 to length (ordered set of dual-failure scenarios) {
2.     RWT← MustUseSpare function(dual-failure scenario_k)
3.     if RWT == 0
4.         terminate
5.     else
6.         if the dual-failure scenario is in [0, 20%×length(ordered set of dual-failure
               scenarios) ]
7.             BR←DisjointRouter function (RWT)
8.         else
9.             R* ←Dijkstra (CostD based on RWT)
10.            Value←minimum spare capacity allocated on R*
11.            if Value ≥ RWT
12.                Temporary_Routes ← RandomRouter (RWT)
13.                append the route with a minimum cost between (R*,
                       Temporary_Routes) to BR
14.            else if 0 < Value < RWT
15.                Temporary_Route ←Dijkstra (CostD based on Value)
16.                append Temporary_Route to BR with Value allocated spare
                       capacities
17.                RWT = RWT – Value
18.                while RWT > 0
19.                    Temporary_Route1 ←Dijkstra (CostD based on RWT)
20.                    Value2←minimum spare capacity allocated on
                       Temporary_Route1
```

```
21.                 if Value2 ≥ RWT
22.                     Temporary_Routes* ← Random-Router (RWT)
23.                     append the route with a minimum cost between
                                (Temporary_Route1, Temporary_Routes*) to BR
                                with RWT allocated spare capacities
24.                 else if 0 < Value2 < RWT
25.                     Temporary_Route2 ←Dijkstra (CostD based on Value2)
26.                     append Temporary_Route2 to BR with Value2 allocated
                            spare capacities
27.                     RWT = RWT – Value2
28.                 else if Value2 = 0
29.                     Temporary_Routes3 ← RandomRouter (RWT)
30.                     append the route with a minimum cost between
                                (Temporary_Route1, Temporary_Routes3) to BR
32.             else if Value = 0
33.                 Temporary_Routes ← RandomRouter (RWT)
34.                 append the route with a minimum cost between (R*,
                        Temporary_Routes) to BR}
```

After applying the *MustUseSpare* function to a dual-failure scenario, there could be a number of working traffic that still has not been assigned any backup route. Thus, we proposed novel routing functions for optimal routing of the remainder of such working traffics. As we move forward through the chromosome and find backup routes for more dual-failure scenarios, the amount of the allocated spare capacity on the network will increase. Thus, for routing the remainder of the working traffics after applying the *MustUseSpare* function, we divide the dual-failure scenarios into two groups based on their location in the chromosome. The first group includes the first 20% of the dual-failure scenarios in the chromosome and the second group includes the remaining dual-failure scenarios. Then, we apply the routing function specifically designed for each

group to restore the remainder of the working traffic after applying the *MustUseSpare* function.

(1) Routing the first 20% of the DUAL-FAILURE SCENARIOs in a chromosome:

To increase the chance of using previously allocated spare capacity during routing of the upcoming dual-failure scenarios later in the chromosome, we proposed a function named *DisjointRouter* that encourages the *HeuristicRouting* function to distribute the restoration flow over more than one backup route during routing of the dual-failure scenarios. In other words, as we plan the restoration of more dual-failure scenarios one by one, there will be more and more number of backup routes in the network, i.e., the spans that carry the spare capacity for the restoration of the working traffics will be more diversely distributed. Thus, the future dual-failure scenarios would have a higher chance to use the already allocated spare capacity, instead of placing new spare capacity in the network. Algorithm 9 provides the pseudocode of the proposed *DisjointRouter* function.

Algorithm 9. Pseudocode of the *DisjointRouter* function

Input:

Div: the number of restoration backup routes for every span in the dual-failure scenario

WT: the amount of working traffic to be restored

CostD: the set of cost coefficients for Dijkstra's algorithm

F_{in}: the set of span installation costs

BR_t_i: the set of temporarily backup routes for span *i* that is under evaluation

Output:

Route_temp: temporary variable to store one found backup route at a time

BR_temp: the set of temporary backup routes

Sol_temp: the set of temporarily-assigned spare capacities on the spans

F_{out}: the updated set of span installation costs

***DisjointRouter* function:**

1. for *i* from 1 to *Div*{
2. if *i* < *Div*
3. *temp_WT* = integer(*WT*/*Div*)
4. else
5. *temp_WT*=*WT* – *i* × integer(*WT*/*Div*)
6. *Route_temp*←Dijkstra (*CostD*)
7. if cost of *Route_temp* == ∞
8. break
9. append (*Route_temp*) to BR_t_i with magnitude *temp_WT*}
10. *ODiv* = length(*BR_temp*)
11. *temp_WT* = integer(*WT*/*ODiv*)
12. for *i* from 1 to *ODiv* {
13. if *i* == *ODiv*
14. *temp_WT*=*WT* – *i* × integer(*WT*/*ODiv*)
15. Replace the magnitude of BR_t_i with *temp_WT*
16. F_{in}(spans in BR_t_i) = 0
17. *UpdateAvailableSpare*:

```
18.     for j from 1 to length(BR_t_i) {
19.         if Sol_temp(BR_t_ij) < temp_WT
20.             Sol_temp(BR_t_ij) = 0
21.         else
22.             Sol_temp(BR_t_ij) = Sol_temp(BR_t_ij) - temp_WT }
23.     F_out=F_in
```

(2) Routing the remaining DUAL-FAILURE SCENARIOs in a chromosome:

For the remaining dual-failure scenarios, after applying the *MustUseSpare* function, if there is a non-zero amount of working traffic that is unrestored (*RWT*), we first employ Dijkstra's algorithm to find a backup route for *RWT*. Then, we determined the minimum (non-zero) spare capacity of the spans along the found backup route and name it *Value*. Finally, one of the two following scenarios was investigated:

(1) *Value* ≥ *RWT* means that the found backup route can carry a portion of the RWT freely using the previously allocated spare capacities (note that the backup route contains spans with zero allocated spare capacity as well, thus, the total cost of the found backup route is not zero). In this scenario, we take a further step to distribute the *RWT* over several routes, one more time to see if it might result in a better solution with less cost. The number of routes was selected by employing the Random-Router heuristic algorithm.

(2) On the other hand, when *Value* < *RWT* and *Value* > 0, we use Dijkstra's algorithm to find a new backup route for restoring the working capacity of *Value* (and not *RWT*). Since the *Value* is less than *RWT*, the probability of using the allocated spare capacities to build a new backup route increases. Then, if *RWT* - *Value* is not equal to zero, we repeat the same process to find

a new backup route, calculate its associated new *Value*, compare the new *Value* to the new RWT and follow the same procedure until RWT is equal to zero.

(3) *Value* = 0 means there is no spare capacity allocated on any of the spans of the found backup route. In this case, we employ the *RandomRouter* heuristic algorithm to find backup routes for the restoration of the unrestored working traffic.

6.4.2 Genetic Operators

The general structure of GA consists of two sets of operators named mutation and crossover responsible for creating new chromosomes from the existing ones. Every genetic operator based on a specific procedure takes a specific number of the already generated chromosomes as input and creates new chromosomes as offspring. In this way, having a set of the initial population of chromosomes, GA can evolve the initial population toward the optimal solution. For GA to be able to find the optimal solution among all of the possible solutions in the search space, and have a fast converging rate through the evolving process, it requires: (1) starting from a rather good initial population (e.g., feasible and as optimal as possible), and (2) conducting a powerful search through the search space. A detailed description of the genetic operators and their requirements is presented in Chapter 2.

6.4.2.1 Crossover

To provide more diversity in the population, we benefit from a single-point crossover with a random point of combination. The crossover took two individuals using a uniform

random selection from the current population as parents and selects a uniform random number within the first 10% of the length of their chromosome as the index of the swapping point along the ordered set of dual-failure scenarios. Then, it swaps all of the dual-failure scenarios from the first one up to the selected random number between the two parents. Doing so, the offspring might have duplicates of some dual-failure scenarios and miss some others. Thus, by finding the repetitive dual-failure scenarios and replacing them with the missing ones from the parents, the offspring become a new feasible ordered set of dual-failure scenarios. The replacement of duplicate dual-failure scenarios with the missing ones is done by a one-by-one correspondence in the first part of offspring based on the appearing order of the duplicates in the second part. As an example, let us say we have two chromosomes that have been selected to be parent 1 and parent 2, each with 15 dual-failure scenarios. Figure 6.5, depicts the single-point crossover over the two selected parents where there are three duplicate dual-failure scenarios in each offspring. After replacing the duplicates with the missing members, we can see each duplicate in its first appearance in the offspring is replaced by a missing member, based on its index in the second part of the offspring (e.g., (3,4) in offspring 2 is the first duplicate in the second part, thus, in the first part of the offspring 2, it is replaced by the first missing member which is (1,5)).

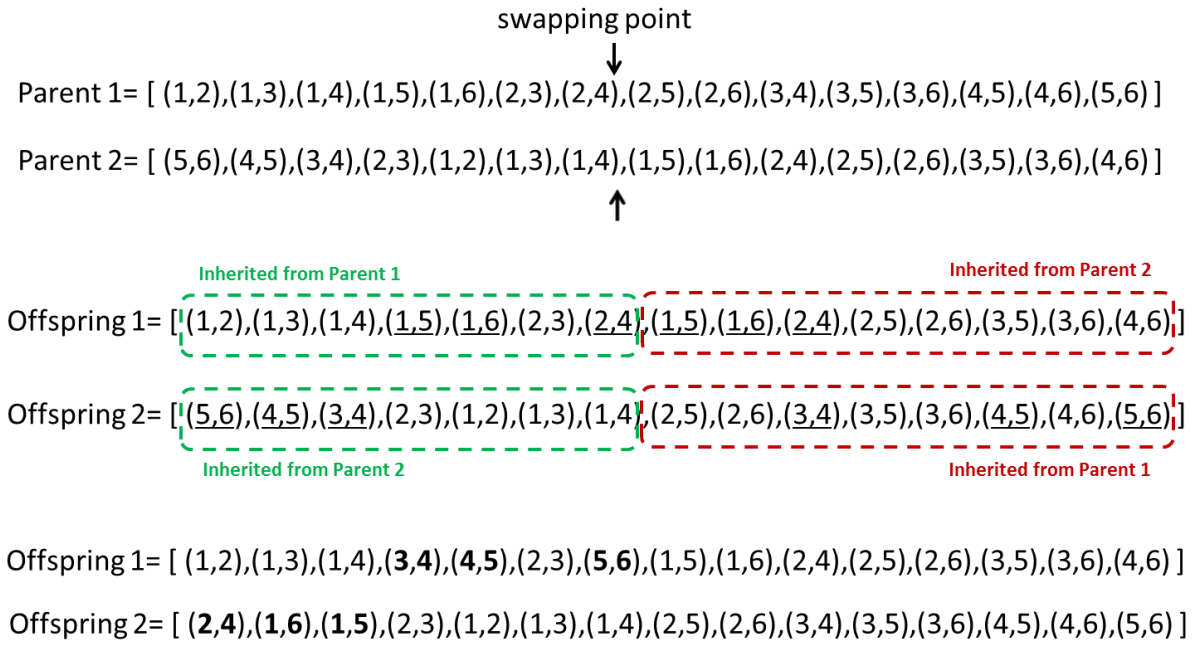


Figure 6.5. The application of the single-point crossover on two chromosomes with 15 dual-failure scenarios, where the random swapping point is after the seventh dual-failure scenario. After the swapping, the first pair of offspring has duplicate dual-failure scenarios that have been underlined. Finally, in the second pair of offspring, the duplicates were replaced by the missing dual-failure scenarios.

6.4.2.2 Mutation

Among the genetic operators, mutation provides the most diversity in the population by improving the search process. In the presented GA, we utilized three mutation functions, and the followings are their detailed procedure:

(1) Mutation 1: The idea is to select a random number of dual-failure scenarios and swap the spans in the selected dual-failure scenario tuples. To do so, first, the ordered set of dual-failure scenarios in the selected chromosome was shuffled and a random number of the dual-failure scenarios within the first 10% of the chromosome's length were selected for swapping, using uniform random selection. For example, if one of the selected dual-failure scenarios was (1, 6),

the mutated chromosome would have the dual-failure scenario (6, 1) in the same location in the chromosome.

(2) **Mutation 2:** The idea is similar to mutation 1 except for the swapping that is conditioned upon the magnitude of the working traffic on the spans in the selected dual-failure scenarios. If the second span of a selected dual-failure scenario had more working traffic than the first span, they were swapped. In every selected dual-failure scenario, if the second span had more working traffic (i.e., more traffic to be restored), it was swapped with the first span, so the span with bigger restoration requirements would be analyzed first. For example, if one of the selected dual-failure scenarios was (1, 6), and the working traffics on spans 1 and 6 were 100 and 200, respectively; the mutated chromosome would have the dual-failure scenario (6, 1) in the same location. While, if the working traffics were 200 and 100, respectively; the dual-failure scenario would not be changed.

(3) **Mutation 3:** This mutation function provides more diversity in the chromosomes by changing the order of the dual-failure scenarios without changing their spans' orders. First, two random numbers within the first 10% of the length of the ordered set of dual-failure scenarios in the chromosome were selected. The two randomly selected numbers represent the indices of the dual-failure scenarios that we want to mutate. All of the dual-failure scenarios between the two random indices were inversed in terms of their order in the chromosome. Figure 6.6 depicts a schematic of mutation 3 on a chromosome with 105 dual-failure scenarios. In the repetitive process of GA evolution, whenever a chromosome

was selected for the mutation, the algorithm randomly chooses one of the three mutation functions to be applied to the selected chromosome.

The choice of rates and selections are made based on our pre-assessment analysis and the explanation earlier in this Chapter, that the genetic operators are mostly focused on the early parts of the chromosomes and generate new chromosomes by making changes in the early dual-failure scenarios in a chromosome. We present a detailed explanation and analysis of the determination process of the control parameters in Section 6.5.2.

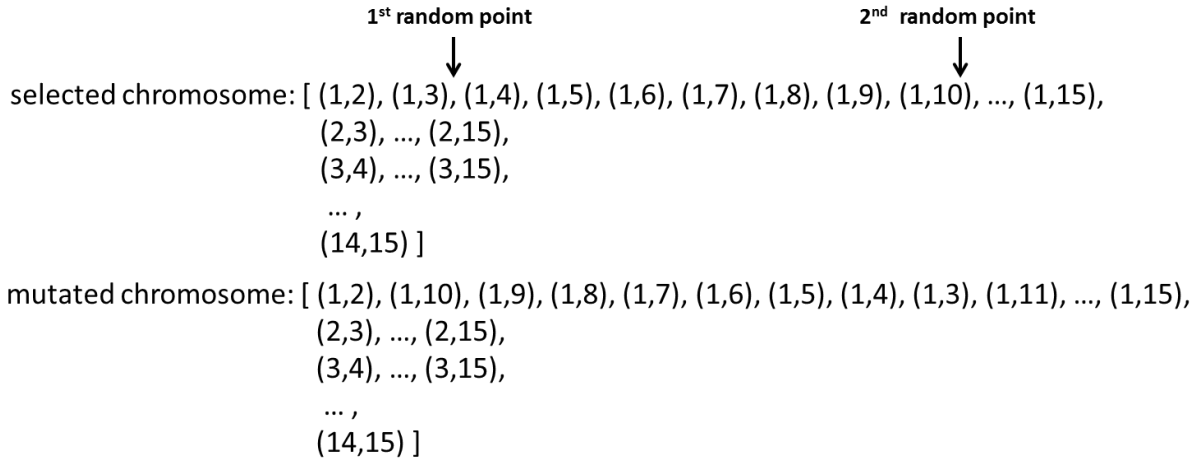


Figure 6.6. The application of mutation3 on a chromosome with 105 dual-failure scenarios, where random indices were 2 and 9.

In this research, we focused on large-scale networks where the length of the ordered set of dual-failure scenarios would be in the order of thousands. As mentioned earlier, by evaluating dual-failure scenarios one by one for restoration purposes, we allocate some spare capacities to the network. Therefore, as we proceed, there will be more and more spare capacities allocated all over the network. The more diversity the early dual-failure scenarios in the ordered set of dual-failure scenarios had, the more spare capacities would be allocated to the more areas of the network. In other words, if the early dual-failure scenarios included more variety of working spans, there would be

more spare capacity allocation in the network, earlier. Thus, as we proceeded further down the ordered set of dual-failure scenarios, there was less need to evaluate the final dual-failure scenarios. Because there have been enough spare capacities already allocated on the network for the restoration of all of the dual-failure scenarios. For example, in a network with ten dual-failure scenarios, if one ordered set of dual-failure scenarios starts with five dual-failure scenarios as: (1,2), (1,3), (1,4), (2,3), (2,4), by the time the fifth dual-failure scenario is evaluated and its spans have been assigned backup routes and spare capacity, only four spans (ie., spans 1, 2, 3, and 4) of ten spans had been evaluated. While, if the ordered set of dual-failure scenarios starts with five dual-failure scenarios: (1,2), (3,4), (5,6), (7,8), (9,10), by evaluating the first five dual-failure scenarios, we evaluate eight spans (i.e., spans 1 through 8) which is more spans all over the network compared to the previous 4 spans. In the latter, the allocated spare capacities might be distributed over greater areas of the network, thus, the upcoming dual-failure scenarios could benefit from those spare capacities through the *MustUseSpare* function. We benefit from this feature in two ways: (1) In our genetic operators, we consider the changes in the chromosomes (e.g., switching and swapping points) early in the ordered set of dual-failure scenarios to be able to generate more diversity in the allocated spare capacity pattern and generate more diverse solutions. For instance, the swapping point in our crossover is randomly selected between the first dual-failure scenario and the dual-failure scenario on the one-tenth of the length of the ordered set of dual-failure scenarios, to make the most difference and variation in the early section of the ordered dual-failure scenario. (2) We conduct a set of experiments in which up until the last iteration of the GA, only a portion of the dual-failure scenarios

are evaluated. To be more precise, let us say the length of an ordered set of dual-failure scenarios was L. We considered a *dual-failure scenario rate* of c_1 where c_1×L of the dual-failure scenarios are to be evaluated in every iteration except for the last iteration. We considered various values for the coefficient c_1 over several runs and conducted comprehensive tuning tests to determine the most appropriate value for it as 0.3.

6.4.3 Objective Function

Herein, the objective function is defined as a cost function that consists of two types of costs: (1) the cost of spare capacity allocation on every span in the network ($\sum_{ij \in S_{all}}(C_{ij} \times S_{ij})$), and (2) the cost of new span installation in the network ($\sum_{ij \in S_{absent}}(F_{ij} \times \hat{\delta}_{ij})$), where the two or three-connectivity connection had not been established and there was a need for new span installation. Once we had the set of backup routes and the number of spare capacities on them, we were able to calculate the total amount of spare capacity required on every span for the full dual failure restorability. To do so, we proceeded as follows: (1) for every dual-failure scenario, the number of spare capacities allocated to any span through the backup routes for both working spans in that dual-failure scenario, had been accumulated. (2) The final number of spare capacities on every span *i* in the network would be the maximum among all of the allocated spare capacities on span *i* over all of the dual-failure scenarios.

6.5 Simulations/Experiments

Our simulations consisted of two categories: (1) the genetic algorithm for the problem of full dual failure restorability of large-scale networks that was generated using Python [73] and (2) the ILP model of dual-failure RN-FCS was programmed in AMPL [59] and

solved using Gurobi [60] 8.1.0 as the benchmark. All of the computations were conducted on a Windows Server 2012 R2- 128 GB RAM- Intel® Xeon® CPU E5- 2650 v3 @ 2.30GHz (2 processors).

6.5.1 Simulation/Experimental Setup

We designed 6 test cases of various sizes and topologies. The detailed specification of the studied networks has been presented in Table 6.1. We considered the spare capacity allocation cost and the new span installation cost to be proportional to the physical length of the spans as the Euclidean distance between the end nodes of a span. Also, as the coordinates of the nodes and the amount of working traffic on every WS were selected randomly, we included various large-scale network instances to check the repeatability of the algorithm's performance.

Table 6.1. The topological specifications of the test cases

Test Case	$\|N\|$	$\|S_{existing}\|$
1	20	27
2	40	49
3	80	85
4	120	133
5	140	152
6	200	227

6.5.2 Sensitivity Analysis (Tuning Control Parameters)

In the presented genetic algorithm, the *HeuristicRouter* function searches for backup routes through the search space. We can tune the control parameters and coefficients

to control the output of the *HeuristicRouter* function. The followings are details on the tuning process of such parameters.

There are control parameters that directly affect the performance of the proposed GA, such as crossover rate, mutation rate, population size, number of generations (iterations), and the previously defined dual-failure scenario rate (c_1). To choose the best values for the control parameters, we executed the proposed GA 15 times for every control parameter setting on test case 1. Notably, for every control parameter, the value of the parameter was changed within a valid range associated with that parameter, while the rest of the parameters were kept constant. The results of the control parameter tunings have been illustrated in Figures 6.7 through 6.12 where the data points represent the calculated average values over 15 runs.

Figure 6.7 demonstrates the effect of various crossover rates on the total dual failure restorable network design cost. We considered the crossover rate of 0.3 as after this rate, the improvements in the cost were not significantly better but the processing times were increased significantly (e.g., compared to the crossover rate of 0.3, the processing time doubled for a crossover rate of 0.5).

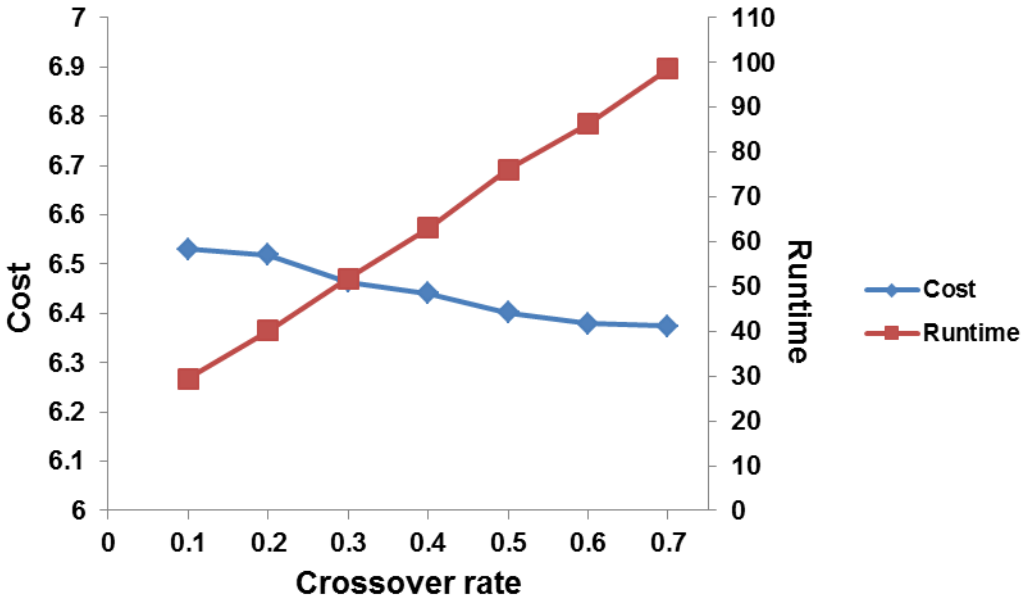


Figure 6.7. The restorable network design cost improvement and runtime increase with respect to various crossover rates.

Figure 6.8 demonstrates the effect of various mutation rates on the total dual failure restorable network design cost. We considered the mutation rate of 0.5 as after this rate, the improvements in the cost were not significantly better but the processing times were increased.

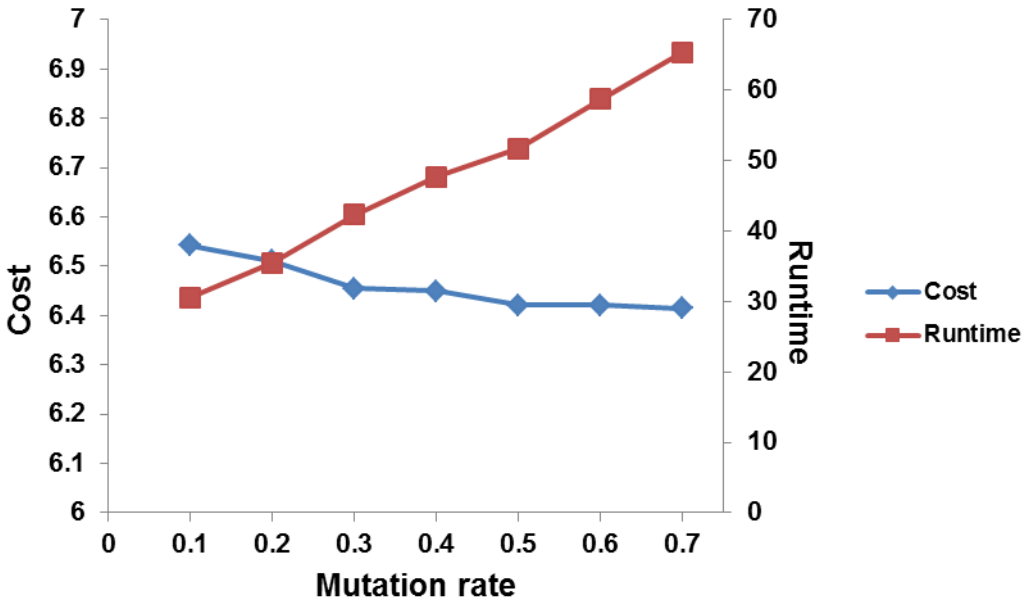


Figure 6.8. The restorable network design cost improvement and runtime increase with respect to various mutation rates.

Figure 6.9 demonstrates the effect of increasing the GA population size on the total dual failure restorable network design cost. We considered the population size of 10 as it had a significant difference from the previous size of 5. As we increased the population size, the improvements in the cost were not significantly better but the processing times almost doubled with every increase in the population size.

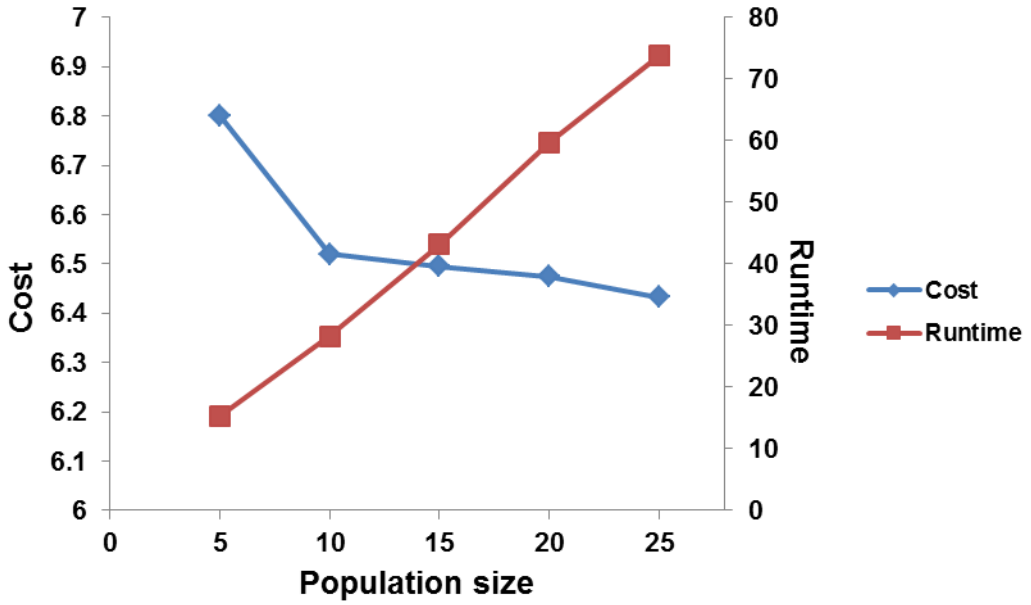


Figure 6.9. The restorable network design cost improvement and runtime increase with respect to various GA population sizes.

Figures 6.10 and 6.11 demonstrate the effect of increasing the number of GA iterations (populations) on the total dual-failure restorable network design cost, considering population sizes of 5 and 10, respectively. We decided 10 generations was the best number as it was significantly better than 1 and no setting was significantly better than 10. As we increased the number of generations over 10, the improvements in the cost were not significantly better but the processing times significantly increased with every increase in the number of generations.

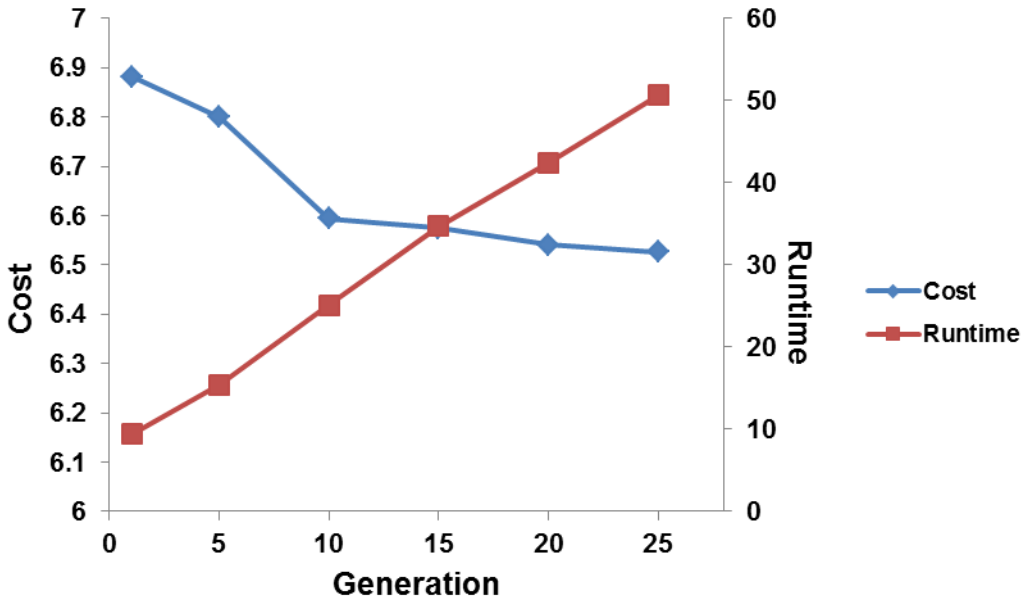


Figure 6.10. The restorable network design cost improvement and runtime increase with respect to the number of GA generations. The results represent average values over 15 runs for every setting with a population size of 5.

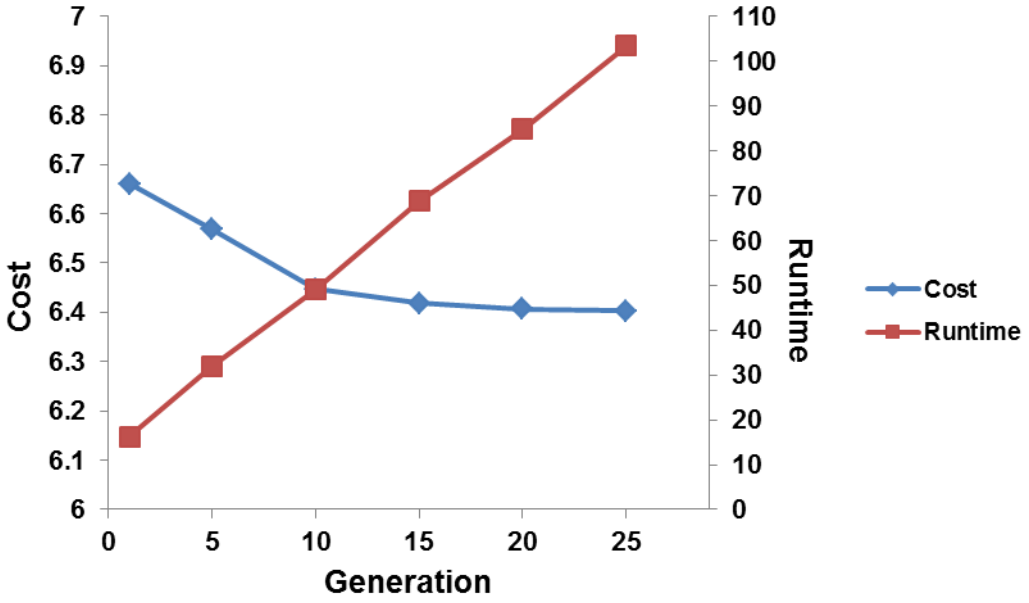


Figure 6.11. The restorable network design cost improvement and runtime increase with respect to the number of GA generations. The results represent average values over 15 runs for every setting with a population size of 10.

Figure 6.12 demonstrates the effect of various dual-failure scenario rates (c_1) on the total restorable network design cost. As we increased the dual-failure scenario rate, the

cost improvements after the dual-failure scenario rate of 0.1 were not remarkable (almost 0.5%) and compared to the dual-failure scenario rate of 0.5 no setting was significantly better, while the processing times were increased tremendously. Moreover, to obtain the exact objective function value associated with analyzing all of the dual-failure scenarios, we had to consider the dual-failure scenario rate of 1 at least in the last iteration. Thus, for all of the iterations except for the last one, we considered the dual-failure scenario rate of 0.3 and at the last iteration, we analyzed all of the dual-failure scenarios in every chromosome of the population.

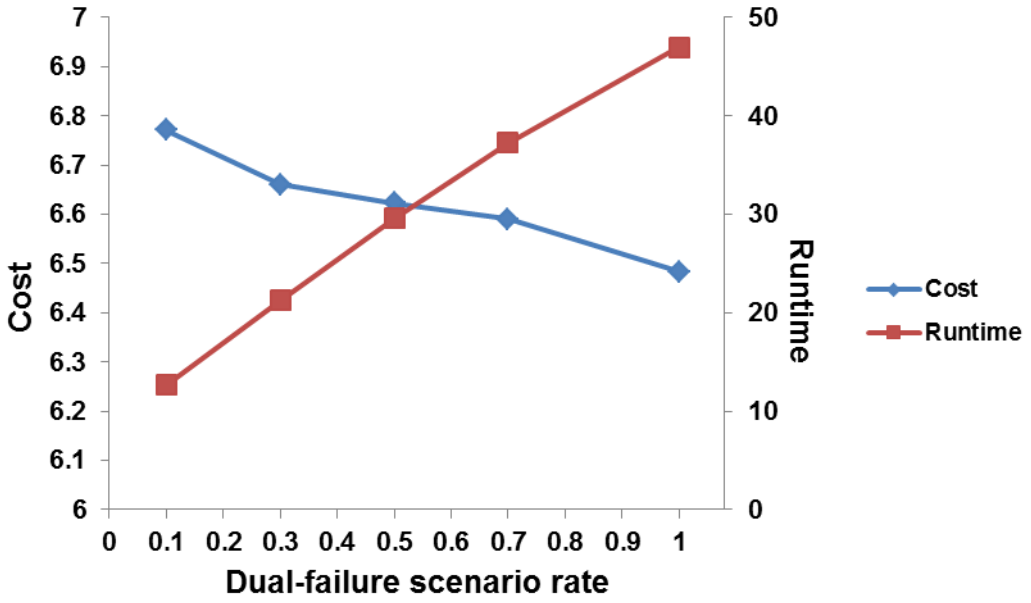


Figure 6.12. The restorable network design cost improvement and runtime increase with respect to the dual-failure scenario rate.

6.5.3 Simulation/Experimental Results and Discussions

Employing the proposed GA, the obtained simulation/experimental results have been tabulated in Table 6.2. The GA's results are the average over five consecutive tests and they were compared to the ILP model as the exact benchmark for the dual failure span restorable network design problem. The normalized costs were calculated using the

ratio of costs obtained from the proposed GA to the benchmark. The processing time improvement percentage was calculated using the difference percentage between the GA and ILP processing times for every test case. The optimality gap was calculated based on the absolute objective function value differences between the GA and employed benchmark models.

Table 6.2. The proposed GA results compared to the dual-failure RN-FCS ILP model. The normalized cost represents the ratio of obtained cost from GA to the best-obtained results from the benchmark

Test case	Normalized cost	GA Processing time (hours)	ILP Processing time (hours)	Processing time improvement %	Optimality gap %
1	1.47	0.01	0.26	94.76	31.98
2	1.43	0.05	5.05	98.97	30.22
3	1.38	0.63	45.92	98.63	27.48
4	1.37	3.11	3 days	95.68	26.86
5	1.39	5.83	>7days	96.69	27.92
6	N/A[1]	13.07	N/A	N/A	N/A

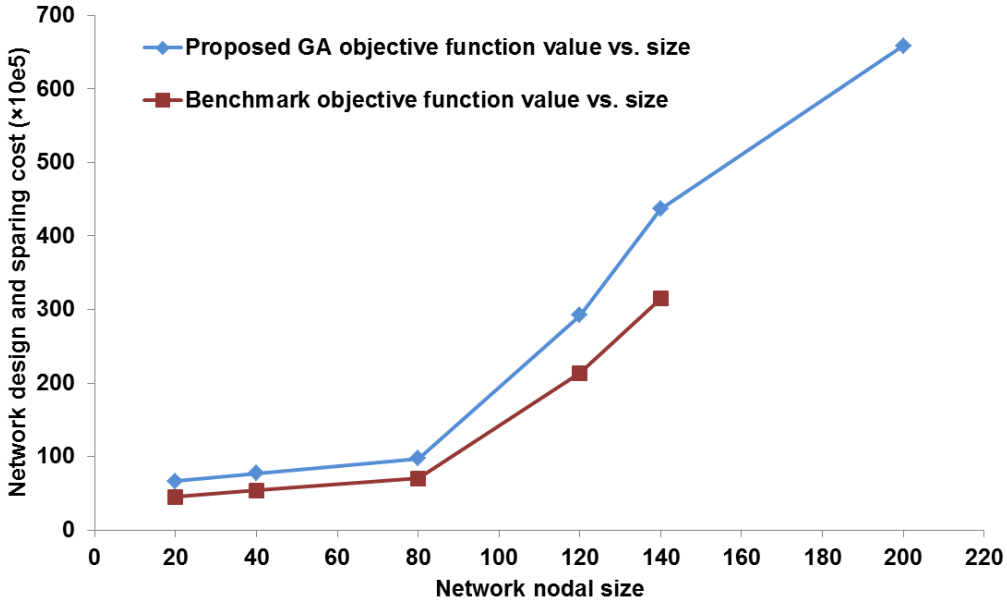
[1] Using the available resources, we were not able to solve test case number 6 as the AMPL was not able to generate the model file for the Gurobi solver.

The obtained results show the optimality gap ranged between 26.86% and 31.98% where their average is 28.89% and the standard deviation is 2.1 and we did not observe any drastic increase in the optimality gap with an increase in the test case sizes. Also, the runtime improvement ranged between 94.76% and 98.97% with an average of 96.95% and a standard deviation of 1.8. Moreover, in addition to the processing time, the RAM usage was significantly lower in GA (e.g., less than 50% for large-scale test cases) compared to the ILP model where 100% of the memory was used.

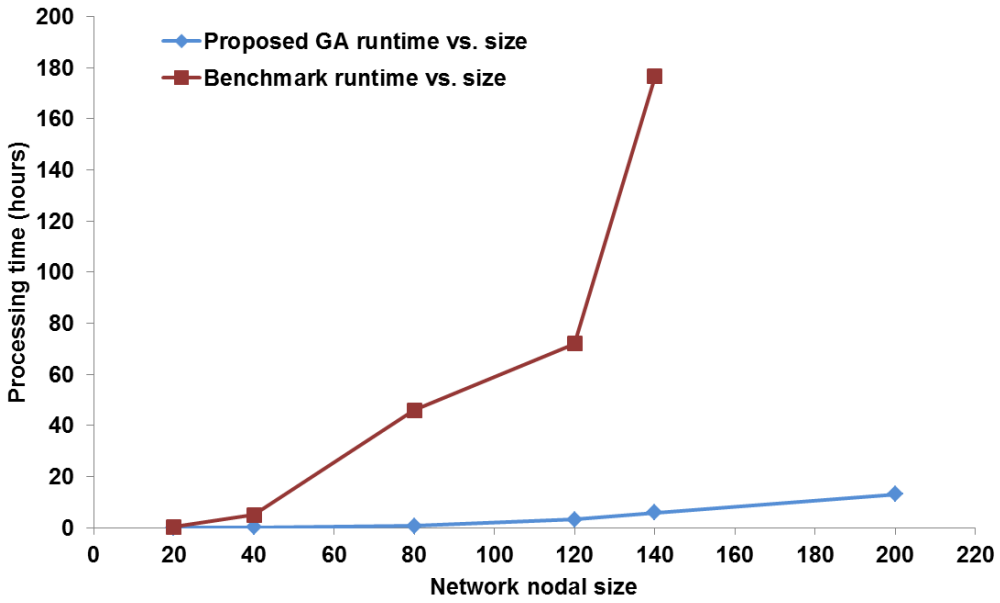
The performance of the solution approach depends on two factors: (1) the available computational infrastructure, and (2) the computational complexity of the

mathematical model. Although the exact solution approaches such as ILP models are guaranteed to find the optimal solution among all of the possible solutions, they have computational complexities that make them inefficient even for medium-size problems when using a powerful machine. Employing ILP-based benchmarks, we observed such computational complexities in the forms of prolonged (prohibitive) processing times and exhaustive CPU usage compared to the proposed GA. Figures 6.13 and 6.14 represent a comparison between the performances of the ILP-based benchmark and the proposed GA with the increase in the network size. According to Figure 6.13, the slope of the cost-size line in GA decreases with the increase in the network size while the benchmark's slope keeps increasing as the network size increases. Moreover, the ILP-based benchmark (dual-failure RN-FCS ILP model) was not able to find any solution for the 200-node network.

According to Figure 6.14, with the increase in the network size, the slope of processing time-size lines increases. However, this increasing trend has a lower rate in GA compared to the benchmark's slope. In other words, the slope of the processing time-size line is lower for the proposed GA compared to the benchmark. Meaning the proposed GA encounters less increase in processing time compared to the benchmark, with an increase in the network size. Moreover, the difference between their performances increases with an increase in network size. For instance, for an 80-node network, this difference is almost 20 units whereas, for a 140-node network, the difference becomes almost 65 units.



Figures 6.13. Performance comparison between the presented GA and benchmark regarding the network cost-size trend. Using the available resources, the dual-failure RN-FCS ILP model was not able to find any solution for the 200-node network.



Figures 6.14. Performance comparison between the presented GA and benchmark regarding the processing time-size trend. Using the available resources, the dual-failure RN-FCS ILP model was not able to find any solution for the 200-node network.

The employed benchmarks can find the optimal solution for any network size in infinite time considering there are an infinite CPU and RAM capacity. On the other hand, heuristics such as GA, provide the opportunity to search for a sub-optimal solution within a considerably lower computational cost (i.e., using smaller RAM and CPU powers within much shorter processing times). Considering various demand matrices and any additional constraints, the network design problem might be needed to be solved several times based on various boundary conditions. Having access to an approach that provides a sub-optimal solution, requires not specifically a supercomputer to execute, is user-friendly(easy) to apply any kind of constraints and objective function structures, and performs in a timely fashion, is beneficial to the network design industry.

6.6 Conclusion

In this chapter, we introduced a novel genetic algorithm for the minimum-cost design of full dual failure span restorable large-scale networks. The restorability against any dual span failure requires the network to be three-connected with regard to every working span. As the initial network under study might only be connected (not necessarily restorable) or two-connected (single failure restorable), we allowed for new span installations upon the requirement to make the network three-connected where necessary. Therefore, the cost function consisted of two main terms: the spare capacity allocation cost and the new span installation cost.

We considered all of the possible dual failure scenarios in the form of an ordered set of dual-failure scenarios as the main section in the designed chromosome. Using the presented routing functions, a set of backup routes for every span in the dual-failure scenarios were routed and they have been assigned a number of spare capacities to facilitate the restoration process of the spans upon failures. The GA chromosomes containing the ordered set of dual-failure scenarios were applied to a variety of genetic operators to perform the evolution through the breeding process. The contributing factors to the performance of the GA (i.e., the control parameters) were determined over a detailed set of tuning tests.

The achieved results were validated using previously developed ILP models as benchmarks. Considering large-scale test cases, the benchmarks were not able to find an optimal solution within the logical time using the available infrastructure, where the introduced GA was able to find a sub-optimal solution within 96.95% $\pm$ 1.8 faster processing times and 50% less memory usage. We were able to achieve an optimality

gap within 28.89% $\pm$ 2.1 compared to the available benchmark. The speed of the introduced GA in a trade-off with its optimality gap can be useful for designing large-scale networks, especially where the network designer wants to try several coefficient matrices and boundary conditions during several sets of runs.

Chapter 7 : Closing Discussion and Summary

The importance of the telecommunication networks' role in our lives is undeniable. We employ the telecommunication networks as the main infrastructure to perform routine activities such as establishing a video conference between two persons in two different parts of the world, completing financial transactions, conducting emergency operations, and generally transferring a huge amount of data per second worldwide. Especially, with recent technological advancements and the huge increase in the number of remote communications all around the world [158], the need for larger networks that are more reliable and cost-effective has been increased.

In this thesis, we focused on the large-scale network design problem and investigated various aspects that are affecting the performance of the network design process that has not been addressed efficiently in recent research works. We have presented the mathematical models for designing large-scale network topologies and routing the pre-defined traffic over the network while taking into consideration the survivability of the network against complex failure scenarios in a cost-effective context.

The employed mathematical models in this thesis include both the deterministic and non-deterministic approaches. When dealing with large-scale networks, the deterministic approaches may encounter computational difficulties and inefficiencies when trying to establish the survivability mechanisms. In some cases, it is not even possible to obtain a feasible solution that allows us to find ways to restore the network functionality. Therefore, we presented non-deterministic approaches for such cases.

In Chapter 3, we investigated the network topology design and routing problem. Given a set of fixed nodes and a set of pre-defined traffic demands to be routed between specified nodes, the goal was to design the network topology by establishing the required spans between the nodes and assigning working capacities to those spans to facilitate the transfer of the traffics with the minimum required cost. The total cost to be minimized in this problem consists of two factors: (1) the fixed cost of any span establishment and (2) the cost of transferring working traffic over every established span. This problem is called Fixed Charge plus Routing and previously, an Integer Linear Programming model was presented for solving it [32], [33]. The presented Integer Linear Programming model provides the minimum cost required for the network topology design and routing problem. However, for large-scale networks, the Integer Linear Programming model was not able to find the optimal solution within logical processing time. Thus, we presented a non-deterministic solution approach based on a Genetic Algorithm that can find sub-optimal solutions with smaller computational expenses. We tested the presented approach on a variety of networks of different sizes with various topological features.

Later in Chapter 4, we investigated the effects of the traffic distributions on the performance of the presented genetic algorithm in Chapter 3. We considered two traffic distributions as scattered and hubbed patterns and investigated the performance of the presented genetic algorithm for either of the traffic patterns on a variety of large-scale networks. The performance of the presented algorithm was evaluated based on the total network topology design and routing cost across the traffics that have been sorted

based on their magnitude as ascending and descending in both scattered and hubbed distributions.

Chapters 5 and 6 are focused on the survivable network design and spare capacity allocation problem. We investigated the dual failure restorability of large-scale networks by employing the span restoration mechanism. We employed an Integer Linear Programming model named Reserved Network Fixed Charge plus Spare capacity for the restoration of Dual failures (dual-failure RN-FCS) based on the presented model in [33] for full dual failure restorability of networks. The developed ILP model showed computational inefficiencies in solving large-scale networks. Because of this, as our contribution to this area of knowledge, we developed two heuristic approaches to solve this problem in logical processing time, and these are discussed in detail in Chapters 5 and 6.

In Chapter 5, we studied the dual failure restorability of networks using an ILP-based heuristic approach. Given a network with a set of existing spans with assigned working capacities, the presented approach seeks to augment the network topology to make it restorable against any dual failure scenario (i.e., make the network three-connected) with minimum design cost. The design cost included two factors: (1) the new span establishment cost and (2) the spare capacity assignment cost. The presented ILP-based heuristic employs various sets of pre-enumerated backup routes to choose among for restoration purposes. Using the dual-failure RN-FCS model as a benchmark, we evaluated the performance of the presented ILP-based heuristic model over a set of networks of various sizes considering different sets of backup routes. Using the available resources, the investigated benchmarks were not able to solve the problem for

large-scale networks after days of running while the presented approach was able to solve such large-scale networks to a sub-optimal solution.

In Chapter 6, we presented a heuristic approach based on a Genetic Algorithm that solves the dual-failure RN-FCS problem for large-scale networks. We designed a set of genetic operators and a *HeuristicRouting* function for routing the spare capacities for the restoration of the affected working traffics under the span restoration mechanism. The performance of the presented Genetic Algorithm was evaluated by the dual-failure RN-FCS model as the benchmark, over a set of test cases of various sizes. Employing the available resources, the mentioned benchmark was not able to solve the 200-node test case while the presented approach was able to solve it in a matter of hours.

7.1 Limitation and Future Work

There are limitations associated with the presented work in this thesis. In Chapter 3, we wanted a set of sub-optimal solutions in the fastest way possible to consider as the initial population for the proposed Master-GA. Therefore, solving the TSP problem to an exact optimal solution was not our priority. However, a limitation regarding the work presented in Chapter 3 is to try to solve the TSP problem using other existing algorithms mentioned in Chapter 2 instead of the IPG-GA and investigate its performance.

Although the proposed IGA has already been validated using a deterministic benchmark in Chapter 3, and as the purpose of Chapter 4 was to only investigate and analyze the effects of various traffic patterns on the performance of the presented IGA,

the presented results in Chapter 4 can be further evaluated using the obtained results from the deterministic benchmarks mentioned in Chapter 3.

Moreover, based on the processing times and the resource usage of the benchmarks of the proposed approaches in Chapters 5 and 6, there were limitations on the number of the test cases that were investigated. Although the performance of the proposed algorithms did not have significant variation throughout the investigated test cases of various sizes, where there are more resources available, a library of test cases could be investigated.

One direction for future work could be employing the proposed network design approaches for mathematically modelling, designing, and analyzing other categories of networks such as airline network design [159], energy and heating network design [160], and transit network design [161], to name a few. While there could be differences with regard to the way that the cost coefficients, decision variables, and constraints will be defined, the main infrastructure of the proposed GA-based approach could be applied and compared against the existing approaches.

With the focus on telecommunication networks, the work presented in this thesis including the mathematical models and algorithms can be extended in several network design areas. The presented algorithms studied the survivable network design problem by employing the span restoration mechanism. The presented models can be extended to other restoration mechanisms mentioned in Chapter 2 where the backup route determination and spare capacity sharing schemes are different. Therefore, the general structure of the presented Genetic Algorithm remains the same while the information

embedded in the genes of the chromosomes will have to change to be tailored to another restoration mechanism. Similarly, the *HeuristicRouting* function will need adjustments to be tailored for other restoration mechanisms. Upon changing the restoration mechanism in any of the presented heuristic algorithms, the control parameters of the algorithm such as the rate of using the operators should be tuned again as their effects on the performance of the algorithm may change.

Another area that the proposed approaches can be extended to is to incorporate traffic uncertainty into the network design problem, similar to the work presented in [162] and evaluate the performance of the proposed approach using existing benchmarks. Moreover, the presented heuristic algorithms based on Genetic Algorithms can be combined with other evolutionary algorithms, similar to the works presented in [163] and [164] to improve the efficiency of the solution process and algorithm's performance. Thus, the performance of the proposed algorithm can be further improved for an increase in the programming and computational cost.

Acknowledging Support

This research work was financially supported by:

1. Alberta Innovates Graduate Student Scholarship-2020.
2. The Natural Science and Engineering Research Council of Canada (NSERC) under Discovery Grant 2017-06198.
3. Alberta Graduate Excellence Scholarship 2019/20
4. University of Alberta Graduate Fellowship 2019/20
5. University of Alberta Doctoral Recruitment Scholarship 2017/18

References

[1] Y. Liu, D. Tipper, and P. Siripongwutikorn, “Approximating Optimal Spare Capacity Allocation by Successive Survivable Routing,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 198–211, 2005.

[2] J. Doucette and 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] D. Tipper, “Resilient network design: Challenges and future directions,” *Telecommunication Systems*, vol. 56, no. 1, pp. 5–16, 2014.

[4] J. E. Doucette, “Advances on Design and Analysis of Mesh-Restorable Networks,” University of Alberta, 2005.

[5] J. Walrand and P. Varaiya, *High-Performance Communication Networks*, Second. Morgan Kaufmann, 2000, 1999.

[6] S. V. Kartalopoulos, *Introduction to DWDM Technology: Data in a Rainbow*. New York: Wiley-IEEE Press, 2000.

[7] A. K. Pradhan, K. Das, and T. De, “Multicast traffic grooming with survivability in WDM mesh networks,” in *2nd International Conference on Signal Processing and Integrated Networks, SPIN 2015*, 2015, pp. 1020–1025.

[8] F. Harary, *Graph Theory (on Demand Printing Of 02787)*. CRC Press, 1994.

[9] F. Harary, “Recent Results in Topological Graph Theory,” *Acta Mathematica*

Academiae Scientiarum Hungaricae, vol. 15, pp. 405–412, 1964.

[10] W. D. Grover, *Mesh-Based Survivable Networks, Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Bernard Goodwin, Prentice Hall PTR, 2004.

[11] W. Wang, “Network Design and Availability Analysis for Large-Scale Mesh Networks,” University of Alberta, 2018.

[12] T. Gomes *et al.*, “A survey of strategies for communication networks to protect against large-scale natural disasters,” *Proceedings of 2016 8th International Workshop on Resilient Networks Design and Modeling, RNDM 2016*, pp. 11–22, 2016.

[13] P. Cholda, A. Mykkeltveit, B. E. Helvik, O. J. Wittner, and A. Jajszczyk, “A Survey of Resilience Differentiation Frameworks in Communication Networks,” *IEEE Communications Surveys & Tutorials*, vol. 9, no. 4, pp. 32–55, 2007.

[14] K. Clay, “https://www.forbes.com/sites/kellyclay/2013/08/19/amazon-com-goes-down-loses-66240-per-minute/#5366fb92495c,” *Retrieved on 4 September 2020*, 2013. .

[15] W. Strong and A. Brockman, “https://www.cbc.ca/news/canada/north/yellowknife-internet-outage-costs-yellowknife-grocer-1.4159669,” *Retrieved on 4 September 2020*, 2017. .

[16] M. Tutton, “https://globalnews.ca/news/3654901/concern-network-outage-atlantic-

canada/," *The Canadian Press, Retrieved on 4 September 2020*, 2017. .

[17] M. Lohatepanont, "Airline Fleet Assignment and Schedule Design: Integrated Models and Algorithms," Massachusetts Institute of Technology, 2002.

[18] V. Miletic, D. Maniadakis, B. Mikac, and D. Varoutas, "On the Influence of the Underlying Network Topology on Optical Telecommunication Network Availability under Shared Risk Link Group Failures," in *10th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2014, pp. 1–8.

[19] B. Todd, "The Use of Demand-wise Shared Protection in Creating Topology Optimized High Availability Networks," University of Alberta, 2009.

[20] M. Clouqueur and W. D. Grover, "Availability Analysis of Span-Restorable Mesh Networks," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 20, no. 4, pp. 810–821, 2002.

[21] S. Kwong, D. W. F. Lam, K. S. Tang, and K. F. Man, "Optimization of Spare Capacity in Self-Healing Multicast ATM Network Using Genetic Algorithm," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 6, pp. 1334–1343, 2000.

[22] I. Diarrassouba, M. K. Labidi, and A. R. Mahjoub, "A parallel hybrid optimization algorithm for some network design problems," *Soft Computing*, vol. 23, pp. 1947–1964, 2019.

[23] W. D. Grover, R. R. Iraschko, and Y. Zheng, "Comparative Methods and Issues In Design of Mesh-Restorable STM and ATM Networks," in *Sansò B., Soriano P.*

(eds) Telecommunications Network Planning, Springer, Boston, MA: Centre for Research on Transportation, 1999.

[24] B. J. Todd, “Socio-Economic Implications on the Design of Telecommunication Networks,” University of Alberta, 2018.

[25] J. Doucette, “Network Optimization and Reliability, Lecture Notes of ENG M 680-B1.” Department of Mechanical Engineering, University of Alberta, Edmonton, Alberta, Canada.

[26] I. N. Bronshtein and K. A. Semendyayev, *Handbook of Mathematics*. Springer, 1985.

[27] T. Weise, *Global Optimization Algorithms – Theory and Application* –, Third. Thomas Weise, tweise@gmx.de, 2011.

[28] W. L. Winston, *Operations Research Applications and Algorithms*. Duxbury Press, 2003.

[29] S. Zionts, *Linear and integer programming*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1974.

[30] B. Ramamurthy and B. Mukherjee, “Wavelength Conversion in WDM Networking,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1061–1073, 1998.

[31] T.-H. Wu, *Fiber network service survivability*. Artech House, 1992.

[32] B. Gendron, T. G. Crainic, and A. Frangioni, *Multicommodity Capacitated Network*

Design. Boston, MA: Sansò B., Soriano P. (eds) Telecommunications Network Planning. Centre for Research on Transportation. Springer, 1999.

[33] W. D. Grover and J. Doucette, “Topological Design of Survivable Mesh-Based Transport Networks,” *Annals of Operations Research*, vol. 106, pp. 79–125, 2001.

[34] L. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, V. Goel, and Y. Shao, “A parallel local search framework for the fixed-charge multicommodity network flow problem,” *Computers & Operations Research*, vol. 77, pp. 44–57, 2017.

[35] B. Thiongane, J. Cordeau, and B. Gendron, “Formulations for the nonbifurcated hop-constrained multicommodity capacitated fixed-charge network design problem,” *Computers & Operations Research*, vol. 53, pp. 1–8, 2015.

[36] J.-F. Labourdette, E. Bouillet, R. Ramamurthy, and A. A. Akyamaç, “Fast Approximate Dimensioning and Performance Analysis of Mesh Optical Networks,” *IEEE/ACM TRANSACTIONS ON NETWORKING*, vol. 13, no. 4, pp. 906–917, 2005.

[37] M. M. Flood, “The traveling-salesman problem,” *Operations Research*, vol. 4, no. 1, pp. 1–137, 1956.

[38] S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

[39] D. S. Johnson and L. A. Mcgeoch, *The Traveling Salesman Problem : A Case Study in Local Optimization. In: Aarts, E.H.L. and Lenstra, J.K., Eds., Local*

Search in Combinatorial Optimization. New York: John Wiley & Sons, 1997.

[40] K. Michalak, “Feasibility-Preserving Genetic Operators for Hybrid Algorithms using TSP solvers for the Inventory Routing Problem,” *Procedia Computer Science- 25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, vol. 192, pp. 1451–1460, 2021.

[41] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[42] O. Kachaiev, “https://gist.github.com/kachayev/5990802,” *Retrieved on 25 August 2020*, 2015. .

[43] J. Doucette, M. Clouqueur, and W. D. Grover, “On the availability and capacity requirements of shared backup path-protected mesh networks,” *Optical Networks Magazine*, no. December, pp. 29–44, 2003.

[44] W. Wang and J. Doucette, “Dual-Failure Availability Analysis of Span-Restorable Mesh Networks,” *Journal of Network and Systems Management*, vol. 24, pp. 534–556, 2016.

[45] G. Ellinas and T. E. Stern, “Automatic protection switching for link failures in optical networks with bi-directional links,” in *IEEE Global Telecommunications Conference (GLOBECOM’96)*, 1996, pp. 152–156.

[46] J. Doucette and W. D. Grover, “Comparison of mesh protection and restoration schemes and the dependency on graph connectivity,” in *International Workshop*

on Design of Reliable Communication Networks (DRCN), 2001, pp. 121–128.

[47] D. Stamatelakis and W. D. Grover, “Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles (‘p-cycles’),” *IEEE Transactions on Communications*, vol. 48, no. 8, pp. 1262–1265, 2000.

[48] W. D. Grover and D. Stamatelakis, “Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration,” in *ICC “98, IEEE International Conference on Communications, Conference Record, Affiliated with SUPERCOMM”98 (Cat. No.98CH36220)*, 1998, pp. 537–543.

[49] D. A. Schupke, C. G. Gruber, and A. Autenrieth, “Optimal configuration of p-cycles in WDM networks,” in *IEEE International Conference on Communications, ICC 2002 (Cat. No.02CH37333)*, 2002, vol. 5, pp. 2761–2765.

[50] M. Herzberg, S. J. Bye, and 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, 1995.

[51] R. R. Iraschko, M. H. MacGregor, and 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, 1998.

[52] Y. Xiong and 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.

[53] S. Venkatesan, M. Patel, and N. Mittal, “A Distributed Algorithm for Path Restoration in Circuit Switched Communication Networks,” in *24th IEEE Symposium on Reliable Distributed Systems (SRDS’05)*, 2005, vol. 11, no. 4, pp. 226–235.

[54] D. Lucerna, M. Tornatore, B. Mukherjee, and A. Pattavina, “Availability target redefinition for dynamic connections in WDM networks with shared path protection,” in *7th International Workshop on Design of Reliable Communication Networks*, 2009, pp. 235–242.

[55] T. Stidsen, B. Petersen, S. Spoorendonk, M. Zachariasen, and K. B. Rasmussen, “Optimal routing with failure-independent path protection,” *Networks*, vol. 55, no. 2, pp. 125–137, 2010.

[56] S. Kini, M. Kodialam, S. Sengupta, T. Lakshman, and C. Villamizar, “Shared Backup Label Switched Path Restoration.” Internet Engineering Task Force (IETF), Internet Draft, pp. 1–10, 2001.

[57] M. Modarres, M. Kaminskiy, and V. Krivtsov, *Reliability engineering and risk analysis, a practical guide*. New York: Marcel Dekker, 1999.

[58] M. Ben-Daya, S. O. Duffuaa, J. Knezevic, D. Ait-Kadi, and A. Raouf, *Handbook of Maintenance Management and Engineering*. Springer, 2009.

[59] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Second. https://ampl.com/resources/the-ampl-book/, 2003.

[60] Gurobi Optimization LLC, “Gurobi optimizer reference manual,” *www.Gurobi.Com*. 2021.

[61] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Boston, MA, United States: Addison-Wesley Longman Publishing Co., Inc., 1984.

[62] F. Glover, “Tabu Search – Part 1,” *ORSA Journal on Computing*, vol. 1, no. 2, pp. 190–206, 1989.

[63] F. Glover, “Tabu Search: A Tutorial,” *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[64] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[65] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Inc., 1996.

[66] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” in *IEEE International Conference on Neural Networks, IV*, 1995, pp. 1942–1948.

[67] Y. Shi and R. C. Eberhart, “A modified particle swarm optimizer,” in *IEEE International Conference on Evolutionary Computation*, 1998, pp. 69–73.

[68] M. Dorigo and T. Stützle, *Ant Colony Optimization*. London, England: MIT Press, 2004.

[69] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[70] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, Second edi. Hoboken, New Jersey: John Wiley {&} Sons, Inc, 2004.

[71] V. Grant, *The evolutionary process*. New York: Columbia University Press, 1985.

[72] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, Second. John Wiley {&} Sons, Ltd, 2007.

[73] G. Van Rossum and F. L. Drake, “Python 3 reference manual.” CreateSpace, Scotts Valley CA, 2009.

[74] D. A. Schupke and R. G. Prinz, “Capacity efficiency and restorability of path protection and rerouting in WDM networks subject to dual failures,” *Photonic Network Communications*, vol. 8, no. 2, pp. 191–207, 2004.

[75] D. A. Schupke, W. D. Grover, and M. Clouqueur, “Strategies for enhanced dual failure restorability with static or reconfigurable p -cycle networks,” in *International Conference on Communications*, 2004, pp. 1628–1633.

[76] A. Jaekel, A. Bari, Q. Rahman, Y. Chen, S. Bandyopadhyay, and Y. Aneja, “Resource efficient network design and traffic grooming strategy with guaranteed survivability,” *Optical Switching and Networking*, vol. 9, no. 4, pp. 271–285, 2012.

[77] A. Fumagalli, I. Cerutti, and M. Tacca, “Optimal design of survivable mesh networks based on line switched WDM self-healing rings,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 501–512, 2003.

[78] X. Dong, T. E. H. El-gorashi, and J. M. H. Elmirghani, “On the Energy Efficiency of

Physical Topology Design for IP OverWDM Networks," *Journal of Lightwave Technology*, vol. 30, no. 12, pp. 1931–1942, 2012.

[79] V. Y. Liu and D. Tipper, "Spare capacity allocation using shared backup path protection for dual link failures," *Computer Communications journal*, vol. 36, pp. 666–677, 2013.

[80] N. Bahria El Asghar, A. Ben Fradj, M. Frikha, and S. Tabbane, "A heuristic approach for solving the virtual topology design problem in next generation networks," in *the 14th International Symposium on Wireless Personal Multimedia Communications: Communications, Networking and Applications for the Internet of Things, WPMC'11*, 2011, pp. 1–5.

[81] W. D. Grover and J. Doucette, "A novel heuristic for topology planning and evolution of optical mesh networks," in *IEEE Global Telecommunications Conference (GLOBECOM'01)*, 2001, pp. 2169–2173.

[82] L. W. Clarke and G. Anandalingam, "A Bootstrap Heuristic for Designing Minimum Cost Survivable Networks," *Computers and Operations Research*, vol. 22, no. 9, pp. 921–934, 1995.

[83] B. Jaumard and H. A. Hoang, "Design and dimensioning of logical survivable topologies against multiple failures," *Optical Communications and Networking*, vol. 5, no. 1, pp. 23–36, 2013.

[84] B. Gendron, S. Hanafi, and R. Todosijević, "An efficient matheuristic for the multicommodity fixed-charge network design problem," *IFAC-PapersOnLine*, vol.

49, no. 12, pp. 117–120, 2016.

[85] H. Üster and S. K. S. Kumar, “Algorithms for the design of network topologies with balanced disjoint rings,” *Heuristics*, vol. 16, pp. 37–63, 2010.

[86] D. Wang and J. Y. Mcnair, “Topological optimization for spare-sharing-based wavelength-routed all-optical networks,” *Optical Switching and Networking*, vol. 9, pp. 297–313, 2012.

[87] T. Mikossfl, T. Takenaka, R. Sugiyama, A. Masuda, and K. Shiomoto, “Multi-Layer Network Topology Design for Large-Scale Network,” in *23rd International Teletraffic Congress (ITC 2011)*, 2011, pp. 306–307.

[88] D. Pereira Junior and M. Camillo Penna, “A new algorithm for dimensioning resilient optical networks for shared-mesh protection against multiple link failures,” *Optical Switching and Networking*, vol. 13, pp. 158–172, 2014.

[89] J. Sun, Q. Zhang, and J. Li, “Two-level evolutionary approach to the survivable mesh-based transport network topological design,” *Heuristics*, vol. 16, pp. 723–744, 2010.

[90] K. Inoue, S. Arakawa, and M. Murata, “A biological approach to physical topology design for plasticity in optical networks,” *Optical Switching and Networking*, vol. 25, pp. 124–132, 2017.

[91] Y. Xin, M. Shayman, R. J. La, and S. I. Marcus, “Reconfiguration of survivable IP over WDM networks,” *Optical Switching and Networking*, vol. 21, pp. 93–100,

2016.

[92] D.-R. Din and Y.-S. Chiu, “A genetic algorithm for solving virtual topology reconfiguration problem in survivable WDM networks with reconfiguration constraint,” *Computer Communications*, vol. 31, no. 10, pp. 2520–2533, 2008.

[93] Y. Aneja, S. Bandyopadhyay, and A. Jaekel, “On routing in large WDM networks,” *Optical Switching and Networking*, vol. 3, pp. 219–232, 2006.

[94] T. G. Crainic, Y. Li, and M. Toulouse, “A first multilevel cooperative algorithm for capacitated multicommodity network design,” *Computers and Operations Research*, vol. 33, pp. 2602–2622, 2006.

[95] A. Olivera, F. R. Amoza, and C. E. Testuri, “A GRASP algorithm for a capacitated, fixed charge, multicommodity network flow problem with uncertain demand and survivability constraints,” *International Transactions in Operational Research*, vol. 17, pp. 765–776, 2010.

[96] B. Gendron, S. Hanafi, and R. Todosijevic, “Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design,” *European Journal of Operational Research*, vol. 268, pp. 70–81, 2018.

[97] C. Chu, G. Premkumar, and H. Chou, “Digital data networks design using genetic algorithms,” *European Journal of Operational Research*, vol. 127, pp. 140–158, 2000.

[98] S. Kwong, T. M. Chan, K. F. Man, and H. W. Chong, “The use of multiple objective genetic algorithm in self-healing network,” *Applied Soft Computing*, vol. 2, no. 2, pp. 104–128, 2002.

[99] D. P. Onguetou and W. D. Grover, “Solution of a 200-node p-cycle network design problem with GA-based pre-selection of candidate structures,” in *IEEE International Conference on Communications*, 2009, pp. 1–5.

[100] Y. Xin, G. N. Rouskas, and H. G. Perros, “On the physical and logical topology design of large-scale optical networks,” *Journal of Lightwave Technology*, vol. 21, no. 4, pp. 904–915, 2003.

[101] H. T. T. Binh and S. H. Ngo, “Survivable flows routing in large scale network design using genetic algorithm,” in *Jeong H., S. Obaidat M., Yen N., Park J. (eds) Advances in Computer Science and its Applications. Lecture Notes in Electrical Engineering*, vol. 279, Berlin, Heidelberg: Springer, 2014, pp. 345–351.

[102] D. Schupke and R. Prinz, “Performance of path protection and rerouting for WDM networks subject to dual failures,” in *OFC 2003 Optical Fiber Communications Conference*, 2003, vol. 1, pp. 209–210.

[103] J. Ahmed, C. Cavdar, P. Monti, and L. Wosinska, “Hybrid survivability schemes achieving high connection availability with a reduced amount of backup resources,” *Journal of Optical Communications and Networking*, vol. 5, no. 10, pp. A152–A161, 2013.

[104] M. Clouqueur and W. D. Grover, “Mesh-restorable networks with enhanced dual-

failure restorability properties," *Photonic Network Communications*, vol. 9, no. 1, pp. 7–18, 2005.

[105] M. Clouqueur and W. D. Grover, "Mesh-restorable networks with complete dual failure restorability and with selectively enhanced dual-failure restorability properties," in *Optical Networking and Communications*, 2002, vol. 4874, pp. 1–12.

[106] B. Todd and J. Doucette, "Demand-wise shared protection network design and topology allocation with dual-failure restorability," *International Conference on the Design of Reliable Communication Networks, DRCN 2015*, pp. 73–80, 2015.

[107] B. Todd and J. Doucette, "Survivable Network Capacity Allocation and Topology Design Using Multi-period Network Augmentation," *Journal of Network and Systems Management*, vol. 25, no. 3, pp. 481–507, 2017.

[108] N. Gonzalez-Montoro and J. M. Finochietto, "Survivability analysis on non-triconnected optical networks under dual-link failures," in *2017 XLIII Latin American Computer Conference (CLEI)*, 2017, pp. 1–9.

[109] J. M. Gutierrez, M. Jensen, T. Riaz, and J. M. Pedersen, "Restorability on 3-connected WDM networks under single and dual physical link failures," in *2013 International Conference on Computing, Networking and Communications, ICNC 2013*, 2013, pp. 128–132.

[110] P. Sasithong, L. Q. Quynh, P. Saengudomlert, P. Vanichchanunt, N. H. Hai, and L. Wuttisittikulkij, "Maximizing double-link failure recovery of over-dimensioned

optical mesh networks," *Optical Switching and Networking*, vol. 36, no. February 2020, 2020.

[111] M. Lackovic, "An Approach to Optical Network Design using General Heuristic Optimization Framework," *Journal of Information and Organizational Sciences*, vol. 34, no. 2, pp. 175–194, 2010.

[112] M. Bentall, B. C. H. Turton, and C. W. L. Hobbs, "Benchmarking the Restoration of Heavily Loaded Networks using a Two Dimentional Order-Based Genetic Algorithm," in *Second International Conference On Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1997, no. 446, pp. 151–156.

[113] E. Ghashghai and R. L. Rardin, "Using a hybrid of exact and genetic algorithms to design survivable networks," *Computers & Operations Research*, vol. 29, pp. 53–66, 2002.

[114] Y. Wang and C. H. Chen, "Improved Genetic Algorithm to Solve Preplanned Backup Path on WDM Networks," in *19th International Conference on Advanced Information Networking and Applications (AINA'05)*, 2005, pp. 1–8.

[115] X. Cheng, X. Shao, and Y. Wang, "Multiple link failure recovery in survivable optical networks," *Photonic Network Communications*, vol. 14, no. 2, pp. 159–164, 2007.

[116] J. Villalba Paez, A. Talia, E. Davalos Gimenez, and D. Pinto Roa, "Optimal selection of p-cycles on WDM optical networks with shared risk link group independent restorability using genetic algorithm," *IEEE Latin America*

Transactions, vol. 10, no. 1, pp. 1385–1390, 2012.

[117] B. Zhou and H. T. Mouftah, “Spare capacity planning using survivable alternate routing for long-haul WDM networks,” in *IEEE Symposium on Computers and Communications*, 2002, pp. 732–738.

[118] H. T. T. Binh and H. D. Ly, “Genetic Algorithm for Solving Multilayer Survivable Optical Network Design Problem,” *International Journal of Machine Learning and Computing*, no. February 2016, pp. 812–816, 2012.

[119] R. M. Morais, C. Pavan, A. N. Pinto, and C. Requejo, “Genetic algorithm for the topological design of survivable optical transport networks,” *Journal of Optical Communications and Networking*, vol. 3, no. 1, pp. 17–26, 2011.

[120] M. Clouqueur and W. D. Grover, “Availability Analysis and Enhanced Availability Design in p -Cycle-Based Networks,” *Photonic Network Communications*, vol. 10, no. 1, pp. 55–71, 2005.

[121] W. Wang and J. Doucette, “Optimized design and availability analysis of large-scale shared backup path protected networks,” *Telecommunication Systems*, vol. 68, no. 2, pp. 351–372, 2018.

[122] W. Wang and J. Doucette, “Availability Optimization in Shared-Backup Path Protected Networks,” *Journal of Optical Communications and Networking*, vol. 10, no. 5, pp. 451–460, 2018.

[123] W. Wang and J. Doucette, “Dual-Failure Availability Analysis for Multi-Flow

Shared Backup Path Protected Mesh Networks," *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 127–133, 2016.

[124] M. Held and L. Zhou, "Redundancy , Restorability and Path Availability in Optical Mesh Networks," *2006 International Conference on Transparent Optical Networks*, vol. 3, pp. 120–125, 2006.

[125] W. Wang and J. Doucette, "On the Availability and Spare Capacity Requirements of Path-Restorable Transport Networks," in *10th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2018, pp. 1–7.

[126] L. Zhou and M. Held, "Optimization of path availability of span-restorable optical networks," in *SPIE 6193, Reliability of Optical Fiber Components, Devices, Systems, and Networks III*, 2006.

[127] A. Castillo, T. Nakashima-Paniagua, and J. Doucette, "Dual-Failure Restorability of Meta-Mesh Networks," in *10th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2018, pp. 1–8.

[128] A. Castillo, T. Nakashima-paniagua, and J. Doucette, "Dual-failure restorability analysis of span-restorable meta-mesh networks," *Networks*, vol. 75, pp. 405–419, 2020.

[129] S. Doostie, T. Nakashima-Paniagua, and J. Doucette, "A Novel Genetic Algorithm-Based Methodology for Large-Scale Fixed Charge plus Routing Network Design Problem with Efficient Operators," *IEEE Access*, vol. 9, pp. 114836–114853, 2021.

[130] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, “Hamilton paths in grid graphs,” *SIAM Journal on Computing*, vol. 11, no. 4, pp. 676–686, 1982.

[131] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., Wiley Series in Discrete Mathematics & Optimization, 1998.

[132] W. D. Grover and 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.

[133] Y. Wei, G. Shen, and S. K. Bose, “Span-restorable elastic optical networks under different spectrum conversion capabilities,” *IEEE Transactions on Reliability*, vol. 63, no. 2, pp. 401–411, 2014.

[134] J. Doucette and Z. Shen, “Meta-mesh span restoration and increased lightpath routing in sparse network topologies,” in *IEEE International Conference on Communications*, 2008, no. February, pp. 5274–5280.

[135] J. Doucette and 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.

[136] J. Doucette and W. D. Grover, “Capacity design studies of span- restorable mesh transport networks with shared-risk link group (SRLG) effects,” in *Optical Networking and Communications*, 2002, vol. 4874, pp. 25–38.

[137] R. G. Prinz, A. Autenrieth, and D. A. Schupke, “Dual failure protection in multilayer networks based on overlay or augmented model,” in *International Workshop on Design of Reliable Communication Networks DRCN- “Reliable Networks for Reliable Services,”* 2005, pp. 179–186.

[138] R. Diestel, *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2005.

[139] V. Y. Liu and D. Tipper, “Spare capacity allocation using partially disjoint paths for dual link failure protection,” *2013 9th International Conference on the Design of Reliable Communication Networks, DRCN 2013*, pp. 171–178, 2013.

[140] L. Zhou, “Restoration mechanism and maximum dual failure restorability of span-restorable optical networks,” in *Asia Optical Fiber Communication and Optoelectronic Exposition and Conference, AOE*, 2006, pp. 1–10.

[141] Nohd0, “Dijkstra Algorithm- Dijkstras.” p. https://github.com/nohd0/Dijkstras/blob/master/Dij, 2018.

[142] J. W. Suurballe and R. E. Tarjan, “A quick method for finding shortest pairs of disjoint paths,” *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[143] MATLAB, “The Math Works, Inc. MATLAB. Version R2020a.” The MathWorks Inc., 2020.

[144] D. Hutchison and J. P. G. Sterbenz, “Architecture and design for resilient networked systems,” *Computer Communications*, vol. 131, pp. 13–21, 2018.

[145] J. Rak *et al.*, “Future research directions in design of reliable communication

systems," *Telecommunication Systems*, vol. 60, no. 4, pp. 423–450, 2015.

[146] W. D. Grover, "Self-Organizing Broad-Band Transport Networks," *Proceedings of the IEEE*, vol. 85, no. 10, pp. 1582–1611, 1997.

[147] H. Sakauchi, Y. Nishimura, and S. Hasegawa, "A self-healing network with an economical spare-channel assignment," in *GLOBECOM '90: IEEE Global Telecommunications Conference and Exhibition*, 1990, pp. 438–443.

[148] Y. Wei, G. Shen, and S. K. Bose, "Span-restorable elastic optical networks under different spectrum conversion capabilities," *IEEE Transactions on Reliability*, vol. 63, no. 2, pp. 401–411, 2014.

[149] D. Schupke and R. Prinz, "Performance of path protection and rerouting for WDM networks subject to dual failures," in *OFC 2003 Optical Fiber Communications Conference*, 2003, vol. 1, no. 3, pp. 209–210.

[150] D. S. Yadav, S. Rana, and S. Prakash, "A mixed connection recovery strategy for surviving dual link failure in WDM networks," *Optical Fiber Technology*, vol. 19, no. 2, pp. 154–161, 2013.

[151] N. Gonzalez-Montoro and J. M. Finochietto, "Incremental spare capacity allocation for optical networks restoration," in *2015 XVI Workshop on Information Processing and Control (RPIC)*, 2015, pp. 1–6.

[152] Y. Wang and J. Doucette, "ILP-based Heuristic Method for the Multi-period Survivable Network Augmentation Problem," in *8th International Workshop on*

Resilient Networks Design and Modeling (RNDM), 2016, pp. 58–65.

[153] M. Clouqueur and W. D. Grover, “Mesh-restorable networks with enhanced dual-failure restorability properties,” *Photonic Network Communications*, vol. 9, no. 1, pp. 7–18, 2005.

[154] X. Li, S. Lin, S. Chen, Y. P. Aneja, P. Tian, and Y. Cui, “An iterated metaheuristic for the directed network design problem with relays,” *Computers and Industrial Engineering*, vol. 113, pp. 35–45, 2017.

[155] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[156] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[157] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[158] B. Gangopadhyay, J. Pedro, and S. Spaelter, “5G-Ready Multi-Failure Resilient and Cost-Effective Transport Networks,” *JOURNAL OF LIGHTWAVE TECHNOLOGY*, vol. 37, no. 16, pp. 4062–4072, 2019.

[159] S. Birolini, A. Jacquillat, M. Cattaneo, and A. Pais Antunes, “Airline Network Planning: Mixed-integer non-convex optimization with demand – supply interactions,” *Transportation Research Part B: Methodological*, vol. 154, pp. 100–

124, 2021.

[160] D. Hering, A. Xhonneux, and D. Müller, “Design optimization of a heating network with multiple heat pumps using mixed integer quadratically constrained programming,” *Energy*, vol. 226, no. 120384, 2021.

[161] A. Shan, N. Hong Hoang, K. An, and H. L. Vu, “A framework for railway transit network design with first-mile shared autonomous vehicles,” *Transportation Research Part C: Emerging Technologies*, vol. 130, no. May, p. 103223, 2021.

[162] S. E. Terblanche, R. Wessäly, and J. M. Hattingh, “Survivable network design with demand uncertainty,” *European Journal of Operational Research*, vol. 210, no. 1, pp. 10–26, 2011.

[163] L. Shi, J. Gong, and C. Zhai, “Application of a hybrid PSO-GA optimization algorithm in determining pyrolysis kinetics of biomass,” *Fuel*, vol. 323, no. April, p. 124344, 2022.

[164] P. He, Q. Fang, H. Jin, Y. Ji, Z. Gong, and J. Dong, “Coordinated design of PSS and STATCOM-POD based on the GA-PSO algorithm to improve the stability of wind-PV-thermal-bundled power system,” *International Journal of Electrical Power and Energy Systems*, vol. 141, no. April, p. 108208, 2022.

Appendix A. Input file example for the network topology design and routing problem

Following is the template of the input file to the proposed Genetic Algorithm for a 20-node and 190-span network. Presenting all of the test cases will take the space of more than a couple of hundreds of pages. Therefore, we only provided an example of such input files. The complete set of test cases can be provided upon request.

#The set of the nodes

N = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19}

#The set of all of the possible spans

S_{all} = {

0: (0, 1),
1: (0, 2),
2: (0, 3),
3: (0, 4),
4: (0, 5),
5: (0, 6),
6: (0, 7),
7: (0, 8),
8: (0, 9),
9: (0, 10),
10: (0, 11),
11: (0, 12),
12: (0, 13),
13: (0, 14),
14: (0, 15),
15: (0, 16),
16: (0, 17),
17: (0, 18),
18: (0, 19),
19: (1, 2),
20: (1, 3),
21: (1, 4),
22: (1, 5),
23: (1, 6),
24: (1, 7),
25: (1, 8),
26: (1, 9),
27: (1, 10),
28: (1, 11),
29: (1, 12),
30: (1, 13),
31: (1, 14),
32: (1, 15),
33: (1, 16),
34: (1, 17),
35: (1, 18),
36: (1, 19),
37: (2, 3),
38: (2, 4),
39: (2, 5),
40: (2, 6),
41: (2, 7),
42: (2, 8),
43: (2, 9),
44: (2, 10),
45: (2, 11),
46: (2, 12),
47: (2, 13),
48: (2, 14),
49: (2, 15),
50: (2, 16),
51: (2, 17),
52: (2, 18),
53: (2, 19),
54: (3, 4),
55: (3, 5),
56: (3, 6),
57: (3, 7),
58: (3, 8),
59: (3, 9),
60: (3, 10),
61: (3, 11),
62: (3, 12),
63: (3, 13),
64: (3, 14),
65: (3, 15),
66: (3, 16),
67: (3, 17),
68: (3, 18),
69: (3, 19),
70: (4, 5),
71: (4, 6),
72: (4, 7),
73: (4, 8),
74: (4, 9),
75: (4, 10),
76: (4, 11),
77: (4, 12),
78: (4, 13),
79: (4, 14),
80: (4, 15),
81: (4, 16),
82: (4, 17),
83: (4, 18),
84: (4, 19),
85: (5, 6),
86: (5, 7),
87: (5, 8),
88: (5, 9),
89: (5, 10),
90: (5, 11),
91: (5, 12),
92: (5, 13),
93: (5, 14),
94: (5, 15),
95: (5, 16),
96: (5, 17),

97: (5, 18),
98: (5, 19),
99: (6, 7),
100: (6, 8),
101: (6, 9),
102: (6, 10),
103: (6, 11),
104: (6, 12),
105: (6, 13),
106: (6, 14),
107: (6, 15),
108: (6, 16),
109: (6, 17),
110: (6, 18),
111: (6, 19),
112: (7, 8),
113: (7, 9),
114: (7, 10),
115: (7, 11),
116: (7, 12),
117: (7, 13),
118: (7, 14),
119: (7, 15),
120: (7, 16),
121: (7, 17),
122: (7, 18),
123: (7, 19),
124: (8, 9),
125: (8, 10),
126: (8, 11),
127: (8, 12),
128: (8, 13),
129: (8, 14),
130: (8, 15),
131: (8, 16),
132: (8, 17),
133: (8, 18),
134: (8, 19),
135: (9, 10),
136: (9, 11),
137: (9, 12),
138: (9, 13),
139: (9, 14),
140: (9, 15),
141: (9, 16),
142: (9, 17),
143: (9, 18),
144: (9, 19),
145: (10, 11),
146: (10, 12),
147: (10, 13),
148: (10, 14),
149: (10, 15),
150: (10, 16),
151: (10, 17),
152: (10, 18),
153: (10, 19),
154: (11, 12),
155: (11, 13),
156: (11, 14),
157: (11, 15),
158: (11, 16),
159: (11, 17),
160: (11, 18),
161: (11, 19),
162: (12, 13),
163: (12, 14),
164: (12, 15),
165: (12, 16),
166: (12, 17),
167: (12, 18),
168: (12, 19),
169: (13, 14),
170: (13, 15),
171: (13, 16),
172: (13, 17),
173: (13, 18),
174: (13, 19),
175: (14, 15),
176: (14, 16),
177: (14, 17),
178: (14, 18),
179: (14, 19),
180: (15, 16),
181: (15, 17),
182: (15, 18),
183: (15, 19),
184: (16, 17),
185: (16, 18),
186: (16, 19),
187: (17, 18),
188: (17, 19),
189: (18, 19),
})
C = {12.53,
152.78,
38.63,
111.83,
54.45,
109.84,
35.44,
104.39,
100.3,
101.07,
50.45,
123.17,
70.6,
157.32,
66.37,
26.25,
125.57,
161.12,
79.08,
155.76,
42.11,
111.02,
44.92,
109.57,
28.02,
109.17,
102.84,
102.18,
49.66,
124.92,
58.46,
164.35,
55.01,
24.7,
128.65,
166.19,
67.36,
114.28,
58.52,
139.29,
56.04,
136.44,
51.31,
52.95,
56.01,
107.79,
33.24,
166.6,
56.09,
154.43,
131.4,
27.2,
29.53,
162.39,
75.01,
49.41,

72.47, 20.22, 104.09, 23.19, 143.18,
32.8, 102.88, 44.69, 111.2, 59.82,
67.07, 89.05, 153.44, 87.32, 29.15,
61.68, 83.38, 36.24, 99.04, 191.43,
62.77, 38.83, 15.65, 77.49, 51.88,
18.25, 106.12, 110.04, 31.32, 130.27,
84.58, 29.53, 150.75, 75.03, 173.49,
77.08, 163.08, 49.65, 107.86, 13.42,
123.36, 18.44, 20.12, 75.93, 104.39,
67.48, 35.44, 31.78, 68.26, 143,
19.92, 114.27, 65.73, 125.25, 65.3,
87.09, 156.95, 33.53, 57.2, 43.86,
124.1, 31.24, 132.23, 25.06, 139.18,
80.6, 86.02, 60.37, 81.15, 182.78}
84.72, 44.38, 120.42, 121.7, F = {1253,
5.1, 25.08, 86.05, 69.43, 15278,
86.73, 14.14, 26.25, 133.36, 3863,
49.48, 59.94, 57.04, 73.6, 11183,
30.15, 23.71, 131.06, 121.2, 5445,
19.03, 111.29, 11.7, 100.28, 10984,
61.59, 95.52, 55.08, 13.89, 3544,
27.2, 99.18, 23.6, 55.76, 10439,
110.04, 85, 117.72, 129.29, 10030,
99.9, 36.24, 77.42, 192.51, 10107,
98.01, 79.4, 105.65, 12.17, 5045,
86.61, 106.53, 78.45, 60, 12317,
40.31, 94.54, 26.08, 142.41, 7059,
82.93, 84.01, 67.91, 185.59, 15732,
104.81, 80.89, 115.32, 11.66, 6637,
85.21, 29.07, 52.89, 180.75, 2625,

12557, 3324, 6159, 9552, 5508,
16112, 16660, 2720, 9918, 2360,
7908, 5609, 11004, 8500, 11772,
15576, 15443, 9990, 3624, 7742,
4211, 13140, 9801, 7940, 10565,
11102, 2720, 8661, 10653, 7845,
4492, 2953, 4031, 9454, 2608,
10957, 16238, 8293, 8401, 6791,
2802, 7501, 10481, 8089, 11532,
10917, 4941, 8521, 2907, 5289,
10284, 7247, 2022, 10409, 2319,
10218, 3279, 10288, 4469, 11120,
4966, 6706, 8905, 15344, 8732,
12492, 6168, 8338, 3624, 9904,
5846, 6277, 3883, 1565, 7748,
16435, 1825, 10612, 11004, 3132,
5501, 8458, 2953, 15075, 7503,
2470, 7708, 16308, 4965, 10786,
12865, 12336, 1844, 2012, 7593,
16619, 6748, 3544, 3178, 6826,
6736, 1992, 11427, 6573, 12525,
11428, 8709, 15694, 3353, 5720,
5852, 12410, 3124, 13222, 2506,
13929, 8059, 8602, 6037, 8115,
5604, 8472, 4438, 12042, 12170,
13644, 509, 2508, 8605, 6943,
5131, 8673, 1414, 2625, 13336,
5295, 4948, 5994, 5704, 7359,
5601, 3015, 2371, 13106, 12120,
10779, 1903, 11129, 1170, 10028,

1389, [0, 7, 91], [1, 19, 49], [3, 16, 89], [5, 17, 75],
5576, [0, 8, 44], [2, 3, 31], [3, 17, 27], [5, 18, 90],
12929, [0, 9, 98], [2, 4, 20], [3, 18, 96], [5, 19, 44],
19251, [0, 10, 20], [2, 5, 56], [3, 19, 49], [6, 7, 66],
1217, [0, 11, 16], [2, 6, 87], [4, 5, 93], [6, 8, 33],
6000, [0, 12, 72], [2, 7, 18], [4, 6, 82], [6, 9, 72],
14241, [0, 13, 31], [2, 8, 96], [4, 7, 85], [6, 10, 83],
18559, [0, 14, 11], [2, 9, 86], [4, 8, 63], [6, 11, 50],
1166, [0, 15, 34], [2, 10, 100], [4, 9, 40], [6, 12, 98],
18075, [0, 16, 38], [2, 11, 70], [4, 10, 81], [6, 13, 100],
14318, [0, 17, 24], [2, 12, 14], [4, 11, 49], [6, 14, 74],
5982, [0, 18, 75], [2, 13, 35], [4, 12, 30], [6, 15, 73],
2915, [0, 19, 42], [2, 14, 17], [4, 13, 11], [6, 16, 67],
19143, [1, 2, 25], [2, 15, 35], [4, 14, 64], [6, 17, 19],
5188, [1, 3, 91], [2, 16, 43], [4, 15, 15], [6, 18, 70],
13027, [1, 4, 71], [2, 17, 17], [4, 16, 90], [6, 19, 78],
17349, [1, 5, 35], [2, 18, 22], [4, 17, 72], [7, 8, 96],
1342, [1, 6, 82], [2, 19, 66], [4, 18, 51], [7, 9, 55],
10439, [1, 7, 49], [3, 4, 77], [4, 19, 97], [7, 10, 32],
14300, [1, 8, 17], [3, 5, 16], [5, 6, 78], [7, 11, 11],
6530, [1, 9, 85], [3, 6, 62], [5, 7, 71], [7, 12, 81],
4386, [1, 10, 31], [3, 7, 74], [5, 8, 17], [7, 13, 35],
13918, [1, 11, 25], [3, 8, 27], [5, 9, 40], [7, 14, 34],
18278} [1, 12, 15], [3, 9, 73], [5, 10, 31], [7, 15, 22],
D = {[0, 1, 66], [1, 13, 36], [3, 10, 30], [5, 11, 72], [7, 16, 36],
[0, 2, 19], [1, 14, 78], [3, 11, 63], [5, 12, 72], [7, 17, 95],
[0, 3, 64], [1, 15, 81], [3, 12, 48], [5, 13, 18], [7, 18, 13],
[0, 4, 17], [1, 16, 39], [3, 13, 44], [5, 14, 40], [7, 19, 64],
[0, 5, 62], [1, 17, 98], [3, 14, 67], [5, 15, 97], [8, 9, 85],
[0, 6, 59], [1, 18, 18], [3, 15, 97], [5, 16, 62], [8, 10, 48],

[8, 11, 79], [9, 14, 14], [10, 18, 59], [12, 16, 14], [14, 18, 43],
[8, 12, 80], [9, 15, 30], [10, 19, 85], [12, 17, 35], [14, 19, 93],
[8, 13, 80], [9, 16, 87], [11, 12, 63], [12, 18, 60], [15, 16, 57],
[8, 14, 31], [9, 17, 50], [11, 13, 38], [12, 19, 41], [15, 17, 73],
[8, 15, 43], [9, 18, 42], [11, 14, 50], [13, 14, 85], [15, 18, 18],
[8, 16, 63], [9, 19, 95], [11, 15, 75], [13, 15, 29], [15, 19, 17],
[8, 17, 41], [10, 11, 98], [11, 16, 65], [13, 16, 68], [16, 17, 77],
[8, 18, 25], [10, 12, 83], [11, 17, 63], [13, 17, 21], [16, 18, 54],
[8, 19, 11], [10, 13, 95], [11, 18, 80], [13, 18, 19], [16, 19, 64],
[9, 10, 42], [10, 14, 13], [11, 19, 51], [13, 19, 85], [17, 18, 72],
[9, 11, 63], [10, 15, 39], [12, 13, 41], [14, 15, 15], [17, 19, 10],
[9, 12, 86], [10, 16, 46], [12, 14, 95], [14, 16, 38], [18, 19, 99]};
[9, 13, 11], [10, 17, 56], [12, 15, 68], [14, 17, 62],