HEC MONTRÉAL

Network Design Problem with Vulnerability and Budget Constraints

par

Jai Kumar Drave

Yossiri Adulyasak
HEC Montréal
Co-Directeur de recherche

Okan Arslan
HEC Montréal
Co-Directeur de recherche

Sciences de la gestion
(Spécialisation Global Supply Chain Management)

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences
(M. Sc.)

August 2024

Résumé

Étant donné un réseau et un ensemble de demandes de paires origine-destination (O-D) avec des besoins de communication, le problème de conception de réseau avec contraintes de vulnérabilité consiste à sélectionner un sous-ensemble des arêtes à un coût minimal de manière à ce que chaque demande soit connectée entre ses paires O-D par un chemin principal, et qu'elle soit également connectée par un chemin de secours pour garantir la durabilité en cas de défaillance simultanée d'un certain nombre d'arêtes dans le réseau. De plus, les chemins principaux et de secours doivent satisfaire une exigence de qualité de service (QDS). Le niveau de capacité de survie (NCS) du réseau est exprimé en nombre d'arêtes simultanément défaillantes dans le réseau, et la QDS est exprimée comme le nombre d'arêtes sur les chemins principaux et de secours utilisés pour connecter les paires O-D. Dans cette description classique du problème, le NCS du réseau et la QDS pour les chemins principaux et de secours de chaque demande sont des paramètres du problème. Dans ce mémoire, nous les considérons comme des variables et étudions le compromis entre la QDS et le NCS dans la conception du réseau avec une contrainte budgétaire. À cette fin, nous créons une fonction de compromis qui exprime les préférences du décideur entre la QDS et le NCS. Nous formulons le problème comme un programme linéaire mixte basé sur le flux qui maximise la fonction de compromis développée. Nous proposons ensuite une formulation alternative utilisant des « coupes à longueur bornée » et nous développons un algorithme de branch-and-cut pour la résoudre. Nous dérivons également différentes familles d'inégalités valides (IV). Nous réalisons une étude computationnelle approfondie et nous comparons l'efficacité des deux formulations pour résoudre le problème. Nous montrons également que les IV contribuent efficacement à réduire les temps d'exécution, jusqu'à 20 %. Nous examinons le compromis entre la QDS et le NCS et nous découvrons qu'une détérioration d'environ 8 % de la QDS peut améliorer le NCS d'environ 24 %. Cependant, pour améliorer le NCS au même niveau, la QDS se détériore d'environ 14 %. En d'autres termes, nos résultats montrent que, lorsqu'un budget fixe est utilisé, des niveaux élevés d'amélioration du NCS sont plus réalisables qu'une amélioration en pourcentage comparable de la QDS.

Mots-clés

Programmation linéaire entière, Coupe limitée en longueur, Branchement et coupe, Conception de réseau, Résilience, Contraintes budgétaires, Contraintes limitant les sauts, Capacité de survie, Vulnérabilité.

Méthodes de recherche

Modélisation mathématique, analyse exploratoire des données

Abstract

Given a network and a set of origin-destination (O-D) pair demands with communication needs, the Network Design Problem with Vulnerability Constraints selects a subset of the edges at a minimum cost such that every demand is connected between their O-D pairs by a primary path, and they are also connected by a backup path to ensure survivability when a certain number of edges simultaneously fails to operate in the network. Additionally, the primary and back up paths should satisfy a quality-of-service (QoS) requirement. The survivability level (SL) is expressed as the number of simultaneously failing edges in the network, and the QoS is expressed as the number of edges on the primal and backup paths used to connect the O-D pairs. In this classical problem description, the SL of the network and the QoS for the primary and backup paths of every demand are parameters of the problem. In this thesis, we consider them as variables, and study the tradeoff between the QoS and SL in network design subject to a budget constraint. To this end, we devise a tradeoff function that expresses the decision maker's preferences between the QoS and SL. We formulate the problem as a flow-based mixed-integer linear program that maximizes the tradeoff function developed. We then provide an alternative formulation using length-bounded cuts, and develop a branch-and-cut algorithm to solve it. We also derive different families of valid inequalities (VI). We carry out an extensive computational study and compare the effectiveness of the two formulations in solving the problem. We also show that the VIs effectively help in reducing the run times additionally by up to 22%. We investigate the tradeoff between the QoS and the SL and find that a deterioration of approximately 8% in the QoS can improve the SL by approximately 24%. However, to improve the SL by the same level, the QoS deteriorates by approximately 14%. In other words, our findings show that, when using a fixed budget, high levels of improvement in the SL is more achievable than a comparable percentage improvement in the QoS.

Keywords

Integer linear programming, Length-Bounded Cut, Branch-and-cut, Network design, Resilience, Budgeting constraints, Hop-limiting constraints, Survivability, Vulnerability.

Research Methods

Mathematical Modeling, Exploratory Data Analysis.

Table of Contents

Résumé i

Abstract iii

List of acronyms ix

Acknowledgements x

1 Introduction 1

1.1 Literature review 2

1.2 Scientific contributions and the organization of the thesis 7

2 Preliminaries 7

2.1 Metric definitions 7

2.2 Notation 8

2.3 Tradeoff functions between the QoS and SL 10

3 Problem Definition and Model Formulations 12

3.1 Model based on arc flows 12

3.2 Model based on length-bounded cuts 14

4 Model Enhancements 16

4.1 Lifting the length-bounded cut constraints 16

4.2 Variable elimination 16

4.3 Knapsack-type valid inequalities 18

4.4 Generalized knapsack-type valid inequalities 18

5 The Separation Problem 19

5.1 Length-bounded cuts for paths of at most three hops 20

5.2 Minimum cut heuristic 20

5.3 An algorithm for identifying the length-bounded cuts of any hop-limit 20

5.4 An exact IP model for detecting length-bounded cuts 22

5.5 Strengthening the detected length-bounded cuts 23

5.6 The extended branch-and-cut framework 24

6 Computational Study **26**

6.1 Data 26

6.2 Experimental design 26

7 Results and Discussion **29**

7.1 Key performance indicators 29

7.2 Computational performance comparison of the models 29

7.3 The impact of valid inequalities on the computational performance of M2 model 30

7.4 The evolution of various KPIs with respect to budget 36

7.5 Computational performance of the best performing algorithm on the complete dataset 39

7.6 The tradeoff mechanism 41

8 Conclusion **46**

Bibliography **i**

A Appendix for Nomenclature **v**

A.1 Sets v

A.2 Parameters vi

A.3 Variables vii

B Dataset **ix**

C Appendix for the alternative form of the separation algorithm **ix**

List of Tables

1 Minimum, average, and maximum number of edge sets of the networks of "RE-1" considered in the problem for different values of Δ^s and Δ^b. 27

2 General parameter settings for the models in the experimental runs. 28

3 First group experiment results comparing the solution performance of M1 and M2 models without any valid inequality. 30

4 Second group experiment results comparing the solution performance of M2 model with different settings. 31

5 Second group experiment results of best performing M2 model (3rd setting) for "RE-3" networks with different average budget levels and n settings. . . . 33

6 Second group experiment results of best performing M2 model (3rd setting) for different Δ^p, Δ^b, Δ^s and n settings on "RE-3" networks. 35

7 Second group experiment results of best performing M2 model (3rd setting) for every "RE-3'" network for different values of n. 36

8 Values of different Metrics averaged across all the "RE-3" networks for different n values in the second group of experiments. 37

9 Values of different KPIs averaged across all the "RE-3" networks for different budget levels in the B_{set} and n values in the second group of experiments. . . 39

10 Overview of the computational results of the best performing algorithm on all 50 networks in the dataset in the third group of experiments. 40

11 Average percentage of the instances with "Optimal", and "Feasible" optimization status when solved for different datasets of networks considered in this study. 40

12 Detailed results of the best performing algorithm on all 50 networks in the dataset in the third group of experiments. 41

13 Properties of Instance Sets. ix

List of Figures

1 The tradeoff function F_{rhk} in equation (1) for different values of n and k. . . . 11

2 The tradeoff between the backup QoS and SL of the network "D7x7-5-1_10_10_20-1" for different combinations of (n, Δ^p, Δ^b) and $\Delta^s = 3$. 45

List of acronyms

2CNBR: Two-Connected Network with Bounded Rings Problem

DND: Distribution Network Design

ERP: Enterprise Resource Planning

IP: Integer Programming

***k*-ESNDP:** *k*-Edge-disjoint Survivable Network Design Problem

***k*-HSNDP:** *k*-Hop-constrained Survivable Network Design Problem

KPI: Key Performance Indicators

lbcut: Length-bounded Cut

LP: Linear Programming

MIP: Mixed Integer Programming

NDPVC: Network Design Problem with Vulnerability Constraints

NDPVBC: Network Design Problem with Vulnerability and Budget Constraints

NDPVC-PER: Network Design Problem with Vulnerability Constraints and Probabilistic Edge Reliability

O-D: Origin-Destination

FMCG: Fast Moving Consumer Goods

PWCE: Protected Working Capacity Envelope

QoS: Quality-of-Service

RNDP: Ring Network Design Problem

SL: Survivability Level

VI: Valid Inequalities

Acknowledgments

I would like to take this opportunity to express my heartfelt gratitude to all those who have supported and guided me throughout this journey of completing my thesis. Their unwavering encouragement, assistance, and understanding have been instrumental in making this work possible. First and foremost, I extend my deepest appreciation to my esteemed thesis advisors, Professor Yossiri Adulyasak and Professor Okan Arslan, for their invaluable guidance, insightful feedback, and patient mentorship. Their expertise and dedication have significantly contributed to shaping this research and refining my academic abilities. I am truly indebted for the time and effort they have invested in me, especially for recommending me as a MITACS intern at Hydro-Québec. I want to thank Josée Deslongchamps and especially my project manager Amira Dems for providing a flexible working environment and acknowledging and even incorporating my ideas in the Hydro-Québec project. I would like to express my gratitude towards Abderrehman Bani, Mouad Morabit and Charlie Chang, who I have been fortunate to have as research collaborators during my time at Hydro-Québec. I also want to thank Amal Khabou for her extensive technical support to carry out the computations. Koray Yenal and Selami Khaled, thank you for providing the student company throughout our time at Hydro-Québec. In addition to this, I am grateful for all the resources and facilities provided by HEC Montréal, GERAD and the Digital Alliance Research of Canada that facilitated the smooth execution of this study. I also want to personally thank my great friend during my undergraduate studies; a PhD student then and now a Professor, Ram Krishna Shah as well as to Late Professor Sameer Khandekar, for their unwavering support for sharpening my calibre and building my foundation in the realm of academia during my undergraduate studies. Finally, a special note of thanks goes to my friends and family, especially my *Aai* (Mother) Madhavi Arvind Kumar Drave, who has been the pillar of strength and encouragement throughout my life. *Aai*, your unwavering belief in me, your encouragement, and your love have always had an unmeasurable impact, giving me the motivation to overcome challenges and persevere. Without your unconditional love and support, I would not have been here today. Thank you, for everything.

Snow, Mota, Brownie, Kallu, Kalli, your brothers and sisters, and Mani, I dedicate this thesis to you.

1 Introduction

General network design problems aim to identify potential points from a given set of nodes and establish links in the networks to facilitate the flow of entities among the same to optimize a certain objective (Wong 1976; Contreras and Fernández 2012). In the context of supply chain management, this idea of network design extends to the distribution network design (DND) problem, which is a prominent and well-studied problem in the Operations Research community (see the comprehensive reviews of Vidal and Goetschalckx 1997; Beamon 1998; Bilgen and Ozkarahan 2004; Meixell and Gargeya 2005). In DND, the potential points can comprise the locations of suppliers, production plants, warehouses and retail stores, among which the flow of several entities such as single or multi-products, information and cash, is maintained, transferred and (temporarily) stored in the most efficient manner so that several individual objectives can be met, that may include minimization of network design cost, enhancement of the quality-of-service (QoS) (e.g., product delivery), profit maximization or even a multiobjective function (Mangiaracina et al. 2015). DND offers a balanced convolution of a multitude of decisions (e.g., the location, capacity and number of facilities to open, transportation routing, and inventory management), which can yield a significant reduction of 60% in the organization costs (Harrison 2005). This is essentially why DND can be crucial to the overall profitability of a firm. With the rapid advancement in information technologies (IT), companies are getting internationalized to expand their economic operations by exploiting the comparative advantage (e.g., cheap labour) of developing nations. Consequently, the distribution networks have essentially become complex, making them vulnerable to failures. The recent COVID-19 pandemic is a prime example of how global supply chains ranging from medical, fast fashion to even the fast-moving consumer goods (FMCG) sector, got disrupted due to a shortage of essential supplies. Over the years, network design research has expanded to incorporating survivability requirements that add a layer of resiliency to the designed network so that the flow of the concerned entities remains undisrupted (see the discussion on survivable network design problems in Kerivin and Mahjoub 2005).

The resilient or survivable network design problem has found numerous applications in multiple domains, particularly the telecommunication and IT sector being the notable "customers". It is well known that ERP systems have a crucial role in integrating supply chains both internally and externally by facilitating seamless information transfer to ensure end-

to-end supply chain visibility, which ultimately gives companies the necessary competitive advantage. This is due to the crucial role of high-speed internet, which has in fact become an integral part of our society, all due to telecommunications networks, which are designed to provide seamless data transferability among a predefined set of origin-destination (O-D) pairs R (e.g., routers, switches) through transferable links (e.g., fibre optical cables). As networks have a finite processing capacity, they are bound to suffer various levels of technical delays, which eventually impact their connectivity performance. Designing networks with a certain level of QoS, is, therefore, a necessary priority for the network managers for which they often focus on reducing the *jitter*, which represents the time difference between the maximum and the minimum delay among the flow of all packets of a data within a network, to tolerable levels (see Chapter 17 in Resende and Pardalos 2008). Each data packet or signal while being routed, has to be queued, and processed before being forwarded to the next node (router) or link, which aggravates the total delay in its propagation within the network. Since node and queuing delays are the dominant components of delay, bounding the number of nodes or equivalently the hops (links) that each data packet has to traverse through its routing path, through hop-limiting constraints is a usual yet effective way to control the jitter. Moreover, it helps in simplifying the network operations and significantly reduces the routing costs (Balakrishnan and Altinkemer 1992; Wierzbicki and Burakowski 2011). Despite deploying the best equipment, networks are susceptible to physical breakdowns (e.g., cable cuts); therefore, a network's capability to "survive" in order to secure connectivity among its nodes after any failure is another critical criterion to be considered during its design.

1.1 Literature review

Two main survivability mechanisms that have been considered in the literature are the *local rerouting* and the *end-to-end rerouting* (Kerivin and Mahjoub 2005; Gouveia et al. 2008). In the local rerouting mechanism, the demand is rerouted on a new path, built across the extremities of the failed link. On the other hand, end-to-end rerouting builds a completely new path between the origin and destination, to reroute the demand. The literature mainly focuses on the case of end-to-end rerouting. In this context, the k Edge-disjoint Survivable Network Design Problem (k-ESNDP), identifes k edge (or vertex)-disjoint paths among all O-D pairs by selecting a subset of edges (links) from a given network topology at the minimum

network cost possible. The polyhedral properties of k-ESNDP were studied by Stoer (1992) and later investigated for any k values by Grötschel et al. (1995). Note that k-ESNDP may result in longer connectivity paths due to the absence of hop-limiting constraints, which have been studied in the context of closely related (sub)problems of minimum spanning and Steiner trees by Gouveia (1996) and Gouveia (1998) and the shortest path problem by Dahl and Gouveia (2004). Recently, two hop-indexed formulations, characterised by nodes and arcs variables were presented and their linear programming (LP) relaxations were compared by Fortz et al. (2022) for the Steiner problem with hop-limiting constraints. Fortz et al. (2000) proposed and investigated the 2-connected network with bounded meshes problem which involves hop-constrained cycles that guarantee undisrupted connectivity of every O-D pair through two disjoint paths such that each edge must belong to at least one of the cycles in the designed network configuration. A description of the polyhedron associated with the two-connected networks with bounded rings can be found in articles of Fortz and Labbé (2002) and Fortz et al. (2006).

In conjunction with hop-limiting constraints, the k-ESNDP extends to k-Hop constrained Survivable Network Design Problem (k-HSNDP), which ensures that data signals can be efficiently propagated in between an O-D pair through k possible disjoint paths of at most h hops; characterizing the network's QoS. We ask interested readers to refer to the articles by Kerivin and Mahjoub (2005) and Bendali et al. (2010a) to get a comprehensive overview of the k-ESNDP and the cutting plane algorithms for the same. The polytope of k-HSNDP has been investigated in detail for a single O-D pair by Bendali et al. (2010b); as a result of which a polynomial-solvable cutting plane algorithm was proposed for any $k \geq 2$ when $h = 2, 3$. Huygens and Mahjoub (2007) developed integer programming (IP) formulations for the edge as well as vertex (involving vertex disjoint paths) variant of k-HSNDP when $h = 4$ and $k \geq 2$. k-HSNDP for multiple O-D pairs was investigated by Dahl and Johannessen (2004) for which they introduced valid inequalities and developed a branch-and-cut algorithm. An exact algorithm based on Benders decomposition was developed by Botton et al. (2013) by capitalizing on the layered network flow formulation of Gouveia (1998); efficiently solving k-HSNDP for $k \in \{1, 2, 3\}$ with hop-limits $h \leq 5$ in graphs of up to 21 vertices. Very recently, Diarrassouba and Mahjoub (2023) presented and studied the properties of several new classes that are applicable to previously developed models by Diarrassouba et al. (2016) through their

polyhedral investigation of k-HSNDP.

Observe that k-HSNDP relies on the assumption that all the links on the $k-1$ primal paths connecting an O-D pair will simultaneously fail and so the k^{th} backup path will keep the communication intact in the same O-D pair, which is pretty conservative provided the current generation of telecommunication equipments are highly reliable. Moreover, it can lead to costly or even infeasible network designs even when $k \geq 2$. In fact, a primal path may have only partial failures while a certain section of it may remain functional. Leveraging this idea, Gouveia and Leitner (2017) introduced the Network Design Problem with Vulnerability Constraints (NDPVC) that guarantees the existence of hop-constrained primal paths and backup paths after the failure of $k-1$ edges in the network topology for all the O-D pairs. It is important to highlight that the paths in NDPVC need not be disjoint as they can share links and so the backup path has to reroute any signal only across the failed section of the primal path via new nodes/links while it may utilise the working part of the same. Consequently, NDPVC is less restrictive than k-HSNDP with the integer expression $k-1$, representing the SL of a network, along with the requirement of hop-limits serving as the QoS of the primal and backup paths. The NDPVC can be naturally extended in several dimensions. Very recently, Arslan and Laporte (2024) investigated the scenario of multiple links failing simultaneously according to their respective failure rates, for which the network configuration has to be designed by considering a minimum reliability level. They proposed and described the new variant as the Network Design Problem with Vulnerability Constraints and Probabilistic Edge Reliability (NDPVC-PER) and showed that strict reliability requirements can have a significant impact on the cost of the network to be designed.

While there is an incentive to opt for shorter hop-constrained routing paths to design a multi-edge failure-resistant network, network managers often have a limited monetary budget to fulfil QoS and SL requirements simultaneously. Due to the practical importance of budget constraints in real-life situations, they certainly do have a rich literature. For instance, Costa et al. (2009), studied the Steiner problem with revenues by combining both hop-limiting and budgeting constraints. The authors proposed various formulations based on Miller-Tucker-Zemlin, Dantzig-Fulkerson-Johnson subtour elimination constraints and hop-indexed or arc-based variables, and developed their branch-and-cut algorithms, whose effectiveness depended on the hop-limit. For the same problem, Sinnl and Ljubić (2016) strength-

ened the quality of root relaxation bounds through their proposed novel node-based layered graph model, which compromises only node variables on the layered graph rather than the arc variables. Their branch-and-cut algorithm developed for their compact model solved the majority of the considered instances to optimality within seconds. The budget constraints have found applications from the perspective of capacity or financial restriction in the design of dynamic survivable networks such as the Protected Working Capacity Envelope (PWCE), which accomplishes survivability through reserved sufficient slack on the directed p-cycles consisting of at least three arcs. Similarly, in the Ring Network Design Problem (RNDP), the size of the rings providing undisrupted connectivity through spare capacity can be controlled via budgeting constraints. A comprehensive review of the role of budget constraints in PWCE and RNDP can be found in Chapter 16 (and the references therein) of Resende and Pardalos (2008).

In the majority of network design studies, the O-D pairs often have the same hop-limits and survivability requirements and these are often known in advance or stated by the network managers through their experience. Although, through a computational study involving real-life network instances, Orlowski and Wessäly (2006) concluded that demand-dependent or local-level hop-limits should be prioritised over a fixed global hop-limit for all O-D pairs as it neglects information on network size/density and therefore, cost feasible designs of larger networks may require larger hop-limits which will eventually deteriorate the QoS they offer. Also, a larger survivability requirement may often result in very costly networks. In fact, the extensive computational results of Arslan et al. (2020) show that only 16.67% of the tested network configurations were feasible to be designed when $k \geq 6$. Moreover, it is obvious and can inferred from Arslan et al. (2020) that the cost of a NDPVC network is proportional to k and it requires at least half of the budget of a single edge resistant network with every unit increase in k. Note that the O-D pairs usually carry a distinct load of signal demands, which implies the signal distribution is not uniform across the O-D pairs. Thus not all O-D pairs contribute to signal propagation on equal terms and thereby, the resilience of O-D pairs with the smallest shortest path connectivity should be prioritized. Also, since a major portion of the budget is required for network construction (see Chapter 11 in Resende and Pardalos 2008), it will be therefore, more attractive to design a network with individual O-D pair SLs rather than a fixed global SL, so that maintenance resources (and hence, maintenance budget)

can be meticulously invested to less resilient O-D pairs, which can't sustain multiple edge failures. However, coming up with an optimal configuration of (h,k) for every O-D pair in a large network can itself be combinatorially challenging. It is therefore a requirement to treat both h and k as variables in the network design models. To the best of our knowledge, only the article of Almathkour et al. (2024), which deals with the edge-constrained survivable network design problem (ESNDP) with non-uniform low connectivity requirements, involves a varying number (k here) of connecting disjoint paths from O-D pair to another. However, the study was restricted to a polyhedral point of view. Moreover, hop-limiting constraints were not incorporated in the study. Overall, network design problems have never been investigated under the simultaneous application of hop-limiting, survivability and budgeting constraints. In addition to this, different network managers may have their distinct objectives of either improving the QoS of the network or making the network more robust against multi-edge failures. There is a void in the literature which lacks a discussion about any mechanism that designs a network by considering the local level trade-off between QoS and SL of all O-D pairs; which we will fill through our study in this thesis.

Being a closely related problem to the k-HSNDP, NDPVC is also a NP-hard problem. Flow-based formulations have been proposed for the NDPVC in the literature (Gouveia and Leitner 2017), which rely on constructing variables for each combination of edges, O-D pair and hop-limit. Consequently, the LP relaxation bounds for such models were poor, which were though successively improved through the development of branch-and-cut and Benders decomposition algorithms by Gouveia et al. (2018); accelerating solution and computational performance of the same. Unfortunately, as these models have a huge bottleneck of being reliant on explicit enumeration of edge sets to reconstruct paths, they can quickly become combinatorially complex in terms of variables and constraints. Arslan et al. (2020) observed that it is sufficient that the existence of paths is ensured, while their explicit construction is not necessary. Thus, they developed models based on the idea of length-bounded cuts (lbcuts), which rather than enumerating the edge set, implicitly ensure that connecting paths exist in the network. Moreover, these models do not suffer from intractability issues even for higher values of $k(\geq 3)$, unlike the former models. The lbcut based models proved to be very efficient on a wide range of problem instances than the former models and therefore, we will extend the same solution idea for the NDPVBC.

1.2 Scientific contributions and the organization of the thesis

The contibutions of the current study are manifold. First, we propose a new objective function that captures the tradeoff between the edge survivability and quality-of-service per O-D pair in a network. To this end, we generalize the NDPVC by considering these two measures as variables and present the Network Design Problem with Vulnerability and Budget Constraints (NDPVBC). We then develop enhancement mechanisms for our model formulations, which involves utilizing the structural properties of lbcuts to expand their applicability for a wide range of hop-limits and survivability requirements, which in turn strengthens and extends the separation problem of the branch-and-cut framework of Arslan et al. (2020). Moreover, we identify and derive three new families of valid inequalities (VIs). Through extensive computational experimentation, we demonstrate that incorporation of these VIs can accelerate the optimization process by approximately 22%. Additionally, we provide managerial insights and discuss in detail the mechanism offered by our new objective function, which quantifies the network managers' changing preferences between survivability and quality-of-service.

The thesis is structured as follows. Section 2 starts with the preliminaries, QoS and SL metric definitions, tradeoff function between the two metrics, and the formal problem definition. In Section 3, we present two mixed-integer linear formulations involving a model based on arc flows and a model based on lbcuts. Section 4 is dedicated to several model enhancement techniques. In Section 5, we discussed the separation problem and its solution methods. Section 6 presents the data and the experimental design, followed by the results and discussion in Section 7. The study concludes in Section 8.

2 Preliminaries

We now discuss the QoS and SL metrics in Section 2.1, present the notation in Section 2.2, and the tradeoff function between the two metrics in Section 2.3.

2.1 Metric definitions

There are two key metrics in the network design, related to the survivability of the network, and the quality-of-service provided to the demand.

- The **Survivability Level (SL)** measures the ability to maintain connectivity of an O-D demand in the case of communication disruptions due to edge failures. It is expressed as the number of edges simultaneously failing in the network. In the context of NDPVC, every O-D pair is required to be connected in case of simultaneous failure of any $k-1$ edges, where k is a global parameter that states the survivability requirements of a network. A higher value of k implies greater survivability. In the NDPVBC considered in this study, we investigate the SL at a local level, i.e., per O-D demand. The SL per O-D demand is a variable of the problem rather than a parameter.

- The **Quality-of-Service (QoS)** of a demand is measured by the number of edges on the shortest path connecting the O-D pair (or alternatively referred to as "hops"). Having a small number of hops implies a better QoS. In the context of survivable network design problems, a QoS is associated with the primal path of a demand. After the disruption of primal connectivity, the backup path restores the connection; however, the hop-limit of the backup path is generally more relaxed (i.e., may have a higher hop-limit) than that of the primal path.

In the NDPVC, the network infrastructure has to ensure for every demand that a primal path connecting its origin to its destination exists. Additionally, it should ensure that a backup path exists when every edge in any subset of the edge set with cardinality equal to $k-1$ fails. These primal and backup paths must respect the primal and backup hop-limits dictated by the QoS requirements. The parameter k as well as the hop-limit requirements of the demand are assumed to be given by the decision makers apriori in the NDPVC. In our study, we relax this assumption and determine the SL and QoS levels per demand for a given budget.

2.2 Notation

Let $\overline{G} = (V,E)$ be an undirected graph, where V is the set of vertices and E is the set of edges $e = [i,j]$ with $i,j \in V$ and $i < j$. Let $c_e = c_{ij}$ be the non-negative cost of an edge $e = [i,j] \in E$. Consider the corresponding directed graph, $G = (V,A)$, where the arc set A contains two opposite arcs (i,j) and (j,i) for each edge $e = [i,j] \in E$. Let d_{ij} be the hop-distance, defined as the minimum number of arcs connecting vertices i and $j \in V$. We define demand r by a tuple $\langle o_r, d_r, H_r^{min} \rangle$, where o_r and d_r are origin and desination, respectively, and H_r^{min} is the

minimum hop-distance between them, that is, $H_r^{min} = d_{o_r d_r}$. Let R be the set of all demands. We define $H_R^{min} = \max_{r \in R}\{H_r^{min}\}$ as the global hop-distance in the network, representing the maximum of the minimum hop-distances of all demand $r \in R$. Finally, parameter B represents the total budget for the network design.

Let $K = \{\kappa, \kappa+1, \ldots, \kappa+\Delta^s\}$ be the set of survivability levels, where $\kappa - 1$ and $\kappa + \Delta^s - 1$ represents the minimum and maximum survivability levels, respectively and $\kappa \geq 2$. Let SL_r be the survivability level of demand $r \in R$. Having $SL_r = k-1$ for $k \in K$ implies that the network should ensure the existence of a backup path when any $k-1$ edges in the network fails. Solving the NDPVBC will determine SL_r for $r \in R$ (note that survivability may not be provided for some demands).

We also define $H_r^p = \{H_r^{min}, H_r^{min}+1, \ldots, H_R^{min}+\Delta^p\}$ to be the set of QoS levels of the primal paths, where H_r^{min} and $H_R^{min}+\Delta^p$ represent the minimum and maximum hop-limits, respectively, and solving the NDPVBC will determine a value in this set as the QoS of the primal path of every demand. Similarly, the set $H_r^b = \{H_r^{min}, H_r^{min}+1, \ldots, H_R^{min}+\Delta^b\}$ is the set of QoS levels of the backup paths, where H_r^{min} and $H_R^{min}+\Delta^b$ represent the minimum and maximum hop-limits, respectively (note that backup connectivity may not be provided for some demands).

Observe that, for demand $r \in R$ and hop-limit $h \in \mathbb{N}$, an arc $(i,j) \in A$ can be used to route the demand from its origin o_r to its destination d_r if and only if $d_{o_r i} + d_{j d_r} + 1 \leq h$. We therefore build a subgraph per demand to remove the superfluous arcs. For demand $r \in R$, let $A_{rh}^p = \{(i,j) \in A \mid d_{o_r i} + d_{j d_r} + 1 \leq h\}$ for $h \in H_r^p$ be the primal and $A_{rh}^b = \{(i,j) \in A \mid d_{o_r i} + d_{j d_r} + 1 \leq h\}$ for $h \in H_r^b$ be the backup arc sets. Let V_{rh}^p and E_{rh}^p be the vertex and edge sets respectively induced by the primal arc set A_{rh}^p. Similarly, let V_{rh}^b and E_{rh}^b be the vertex and edge sets respectively induced by the backup arc set A_{rh}^b. We refer to the graphs $G_{rh}^p = (V_{rh}^p, A_{rh}^p)$ and $G_{rh}^b = (V_{rh}^b, A_{rh}^b)$ as the primary and backup graphs, respectively, induced by hop-limit h, for each demand $r \in R$.

For $r \in R, h \in H_r^b$ and $k \in K$, consider an edge set $C_k \subset E_{rh}^b$ with cardinality $k-1$. Let $G_{rh}^b(C_k)$ be the graph induced by the arcs $A_{rh}^b(C_k) = \{(i,j) \in A_{rh}^b : [i,j] \in C_k\}$. When C_k has only one edge, we write $A_{rh}^b([i,j])$ rather than $A_{rh}^b[\{[i,j\}]$. In other words, $A_{rh}^b([i,j])$ contains both arcs (i,j) and (i,j) if exists in A_{rh}^b. Similarly, the set $A_{rh}^p[\{[i,j]\}]$ is defined for the primal path.

2.3 Tradeoff functions between the QoS and SL

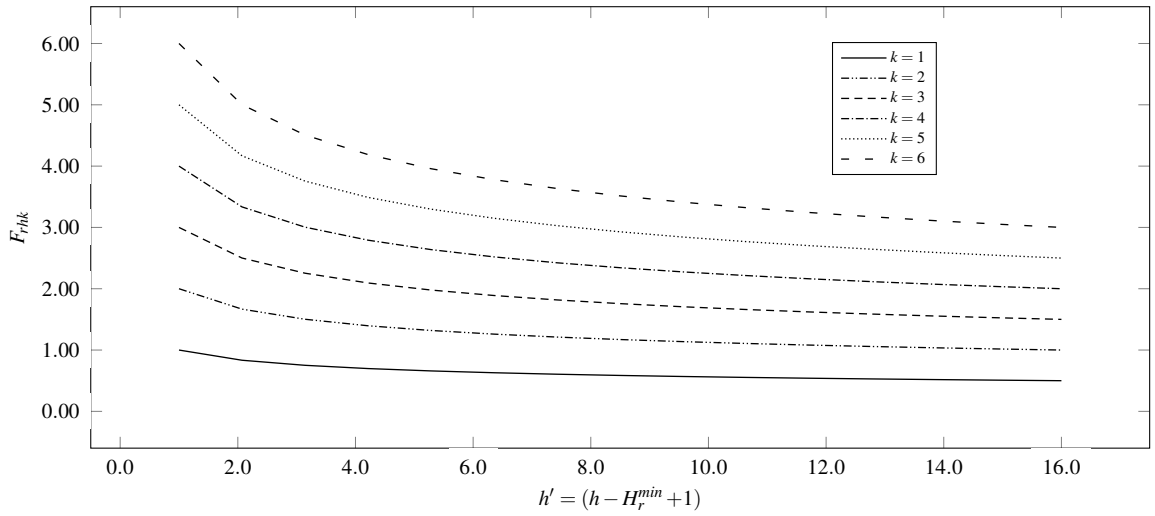
When designing a network, a high level of QoS as well as a high SL for every O-D pair is desired. To achieve this, we introduce the following function to capture the tradeoff between the SL and the QoS:

$$F_{rhk} = \frac{k}{(h - H_r^{min} + 1)^n} \tag{1}$$

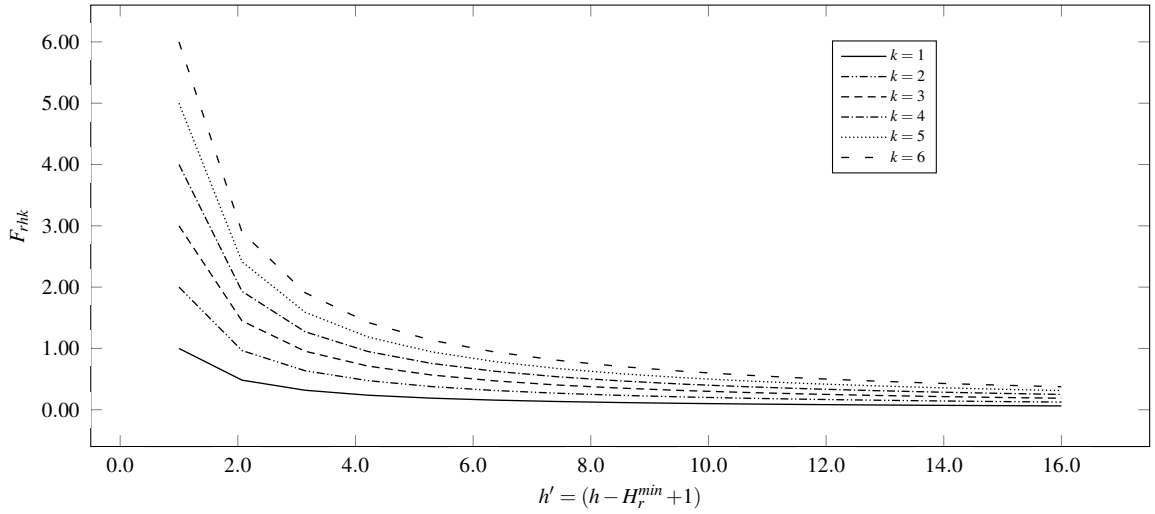
where, for $r \in R$, F_{rhk} is the benefit collected from having a hop-limit of h for the backup paths when the SL is $k-1$ (i.e., the communication can be maintained after the failure of any $k-1$ edges). When $k = 1$, the F_{rh1} represents the benefit collected from having a hop-limit of h for the primal path. The parameter n is a mechanism to prioritise the design of a network towards either a better SL or a better QoS. From the QoS perspective, the higher the value of the hop-limit h is, the smaller F_{rh1} and F_{rhk} values are. From the SL perspective, the higher the value of the k is, the higher the F_{rhk} value is. The denominator has a value of 1 to avoid division by zero.

Consider a demand $r \in R$. For the sake of a concise presentation, we define $h' = h - H_r^{min} + 1$. Figure 1 plots the function F_{rhk} on the vertical axis with respect to h' on the horizontal axis. Figures 1a to 1c show the function with respect to h' for different values of k as n varies from 0.25 to 4.0. Observe that, in all three subfigures, the function value decreases as h' increases and as k decreases. The decrease is more pronounced particularly when h' is small and n is large. Observe that, having $0 < n < 1$ prioritizes SL over QoS while $n > 1$ yields the opposite effect, reflecting the preferences of the decision makers.

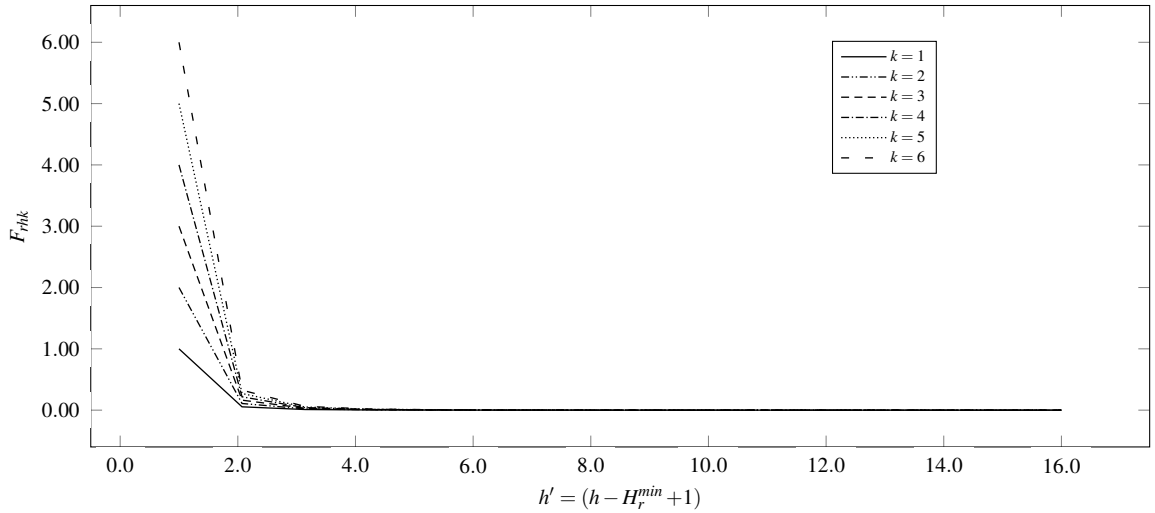
Note that one can treat SL as a requirement (by incorporating it as a constraint in a mathematical model), while maximizing QoS in the objective function under a given budget for edge selection. Alternatively, QoS can be treated as a requirement, with SL as the objective. These special cases correspond to fixing one of the two measures in our function (i.e., either the horizontal or vertical axis in Figure 1). Since both QoS and SL are impacted by edge selection decisions, enforcing one of these two measures induces a certain level in the other measure. Therefore, to allow greater flexibility in edge selection and to better capture the tradeoff between these two metrics, we treat both QoS and SL as variables in our study. The methods developed in this thesis are general and can be adapted to the aforementioned special cases, where one measure is treated as a constraint and the other as the objective.



(a) $n = 0.25$



(b) $n = 1.0$



(c) $n = 4.0$

Figure 1: The tradeoff function F_{rhk} in equation (1) for different values of n and k.

3 Problem Definition and Model Formulations

We now present the NDPVBC and two mathematical models for it. The first model is based on the notion of arc flows, which involves explicitly building paths for every edge failure, while the second one relies on the idea of length-bounded cuts, implicitly ensuring that paths exist without explicitly building them.

Definition 1. *Given an undirected graph $G = (V, E)$, a demand set R, a budget B and the primal Δ^p and backup hop-limits Δ^b respectively and survivability limit Δ^s, the network design problem with vulnerability and budget constraints (NDPVBC) is defined as finding a subset of edges that ensures the connectivity of all demands from their origins to their destinations respecting their primary hop-limits, and determining the SL and the QoS per demand by maximizing the tradeoff function.*

We next introduce the variables to be used in the models. For $e \in E$, let variable x_e equal 1 if edge e is selected and 0 otherwise. For every demand $r \in R$, let y_{rhk} equal 1 if a backup path with a minimum of h hops exists after the failure of any $k-1$ edges, and 0 otherwise. Similarly, variable z_{rh} equals 1 if a primal path of a minimum of h hops exists, and 0 otherwise. Finally, the variable u_{ij}^{rh} equals 1 if the arc (i, j) is used in the primal path of h hops from the vertex o_r to d_r, and 0 otherwise. Similarly, the variable $v_{ij}^{rhC_k}$ equals 1 if and only if the arc (i, j) is used in the backup path of h hops from the vertex o_r to d_r, after the failure of any $k-1$ edges in the set $C_k \subset E_{rh}^b$.

3.1 Model based on arc flows

We now present the arc-based mathematical model that explicitly builds the paths for every edge failure case in the network by enumerating apriori all the subsets of the edge set C_k for $k \in K$. We refer to this model as M1, which is formulated as follows.

$$\text{maximize} \quad \sum_{r \in R} \sum_{h \in H_r^b} \sum_{k \in K} F_{rhk}\, y_{rhk} + \sum_{r \in R} \sum_{h \in H_r^p} G_{rh}\, z_{rh} \tag{2}$$

subject to

$$\sum_{j:(i,j)\in A^p_{rh}} u^{rh}_{ij} - \sum_{j:(j,i)\in A^p_{rh}} u^{rh}_{ji} = \begin{cases} z_{rh} & \text{if } i = o_r \\ -z_{rh} & \text{if } i = d_r \\ 0 & \text{otherwise} \end{cases} \qquad i \in V^p_{rh}, h \in H^p_r, r \in R, \quad (3)$$

$$\sum_{j:(i,j)\in A^b_{rh}} v^{rhC_k}_{ij} - \sum_{j:(j,i)\in A^b_{rh}} v^{rhC_k}_{ji} = \begin{cases} y_{rhk} & \text{if } i = o_r \\ -y_{rhk} & \text{if } i = d_r \\ 0 & \text{otherwise} \end{cases} \qquad i \in V^b_{rh}, k \in K, h \in H^b_r, r \in R, \quad (4)$$

$$\sum_{(i,j)\in \overline{A}^b_{rh}(C_k)} v^{rhC_k}_{ij} \leq \begin{cases} h\, y_{rhk} & \text{if } C_k \neq \emptyset \\ -y_{rhk} & \text{if } C_k = \emptyset \end{cases} \qquad C_k \subset E^b_{rh}, k \in K, h \in H^b_r, r \in R, \quad (5)$$

$$\sum_{(i,j)\in A^p_{rh}} u^{rh}_{ij} \leq h\, z_{rh}, \qquad h \in H^p_r\, r \in R, \quad (6)$$

$$\sum_{(p,q)\in A^p_{rh}([i,j])} u^{rh}_{pq} \leq x_e \qquad e = [i,j] \in E^p_{rh}, h \in H^p_r, r \in R, \quad (7)$$

$$\sum_{(p,q)\in A^b_{rh}([i,j])} v^{rhC_k}_{pq} \leq x_e \qquad C_k \subset E^b_{rh}, e = [i,j] \in E^b_{rh}, h \in H^b_r, r \in R, \quad (8)$$

$$\sum_{h\in H^p_r} z_{rh} = 1 \qquad r \in R, \quad (9)$$

$$\sum_{h\in H^b_r} \sum_{k\in K} y_{rhk} \leq 1 \qquad r \in R, \quad (10)$$

$$\sum_{e\in E} x_e c_e \leq B \quad (11)$$

$$\sum_{h=H^{min}_r}^{H^{min}_R+\Delta^p} \left(\sum_{k\in K} (H^{min}_R + \Delta^p - h)\, y_{rhk} + (h-1)\, z_{rh} \right) \leq H^{min}_R + \Delta^p - 1 \qquad r \in R, \quad (12)$$

$$x_e \in \{0,1\} \qquad r \in R, \quad (13)$$

$$y_{rhk} \in \{0,1\} \qquad k \in K, h \in H^b_r, r \in R, \quad (14)$$

$$z_{rh} \in \{0,1\} \qquad h \in H^p_r, r \in R, \quad (15)$$

$$u^{rh}_{ij} \in \{0,1\} \qquad (i,j) \in A^p_{rh}, h \in H^p_r, r \in R, \quad (16)$$

$$v^{rhC_k}_{ij} \in \{0,1\} \qquad (i,j) \in A^b_{rh} \setminus A^b_{rh}(C_k), C_k \subset E^b_{rh}, h \in H^b_r, r \in R. \quad (17)$$

The objective function (2) maximizes the benefits collected from providing service to the demand at a certain QoS and SL. Note that the objective function coefficients F_{rhk} are determined by the parameter n as in equation (1). Constraints (3) are balance equations,

which along with constraints (9) yield a primary path for every demand $r \in R$. Similarly, if y_{rhk} equals 1, the constraints (4) and (10) build a backup path for demand $r \in R$ using at most h edges hops after the simultaneous failure of edges in set $C_k \subset E^b_{rh}$. Constraints (5) and (6) ensure that the primal and backup paths respect their respective hop-limits. To avoid having $y_{rhk} = 1$ when $C_k = \emptyset$, we have $-y_{rhk}$ in RHS of the constraints (5). Constraints (7) and (8) ensure that an arc can only be used on the primal and backup paths if the corresponding edge is selected. Constraint (11) ensures that the budget is respected for the network design. Observe that, if a better hop-limit QoS can be provided for a backup path, than this path should be used as the primal path. In other words, the path with the best QoS should be allocated to the primal path. This is guaranteed through the constraints (12) that the number of edges in the primal path is at most as many as that of any backup path for every demand $r \in R$. Finally, the constraints (13)$-$(17) define the domains of the variables.

3.2 Model based on length-bounded cuts

Observe that, for $k \in K$, $h \in H^b_r$ and $r \in R$, the number of $C_k \subset E^b_{rh}$ is $\binom{E^b_{rh}}{k-1}$, as it consists of all the subsets with cardinality $k-1$. The number of $v^{rhC_k}_{ij}$ variables in M1 is $|A^b_{rh} \setminus A^b_{rh}(C_k)| \times |C_k| \times |H^b_r| \times |R|$; exponential in size. Therefore, we now develop a new model based on the notion of length-bounded cuts (lbcuts) without the need to enumerate the edge sets C_k, for $k \in K$. We first formally define the length-bounded cut.

Definition 2. *Given a directed graph $G = (V,A)$, a demand $r \in R$ with its O-D pair (o_r, d_r), and a positive integer h, a set of arcs $\overline{S} \subseteq A$ is called a length-bounded cut (lbcut), if all the paths of at most h arcs connecting o_r and d_r in G can be destroyed by the deletion of arcs in $\overline{S}$ from G.*

Observe that every graph cut disconnecting o_r and d_r (including minimum cuts) in G is essentially a lbcut. Additional lbcuts may be present in graph G. Consider the set of edges $S \subseteq E$ induced by the corresponding arcs in $\overline{S}$, which defines the lbcut that disconnects the O-D pair in the corresponding undirected graph G. Let Γ^p_{rh} be the set of all such edge sets $S \subseteq E^p_{rh}$ corresponding to the lbcuts $\overline{S} \subseteq A^p_{rh}$ of length bound h in G^p_{rh}. Similarly, Γ^b_{rh} is defined as the set of all edge sets $S \subseteq E^b_{rh}$ corresponding to the lbcuts $\overline{S} \subseteq A^b_{rh}$ of length bound h in G^b_{rh}. The lbcuts help ensure the existence of hop-constrained primal and backup paths in case

of simultaneous failure of any k -1 edges. We now repeat the Proposition 1 and Theorem 1 in Arslan et al. (2020) for the sake of completeness and refer the reader to the same reference for the proofs.

Proposition 1. *For a given graph G, a demand $r \in R$ with its O-D pair (o_r, d_r) and a hop bound h, there exists a path of length at most h from o_r to d_r if and only if every lbcut contains at least one edge of the path.*

Theorem 1. *For a given $r \in R$, an integer $k \geq 2$, and a binary vector $x \in \{0,1\}^{|E|}$, there exists a hop-constrained backup path of length at most h after the failure of any k-1 edges in the graph if and only if $\sum_{e \in E} x_e \geq k$ for all lbcuts $S \in \Gamma^b_{rh}$.*

The alternative formulation we present next, implicitly ensures the existence of paths via lbcuts, while avoiding the need for exponentially many variables. We refer to this model as $\overline{\text{M2}}$.

$$(\overline{\text{M2}}) \text{ maximize} \quad \sum_{r \in R} \sum_{h \in H^p_r} F_{rh1}\, z_{rh} + \sum_{r \in R} \sum_{h \in H^b_r} \sum_{k \in K} F_{rhk}\, y_{rhk} \tag{2}$$

subject to

(9)−(15)

$$\sum_{e \in S} x_e \geq z_{rh} \qquad S \in \Gamma^p_{rh}, h \in H^p_r, r \in R, \tag{18}$$

$$\sum_{e \in S} x_e \geq \sum_{k \in K} k\ y_{rhk} \qquad S \in \Gamma^b_{rh}, h \in H^b_r, r \in R. \tag{19}$$

The objective function of the model is the same as M1. The existence of the paths are implicitly ensured through (18) and (19). Constraints (9) and (18) together guarantee that only a single hop-bounded primal path of size $h \in H^p_r$ is constructed for each O-D pair $r \in R$. Constraints (10) and (19) together ensure that, demand $r \in R$ can restore connectivity using backup paths of length at most $h \in H^b_r$, after the failure of any $k-1$ edges in the network. The rest of the constraints are as in model M1.

4 Model Enhancements

We will now discuss mechanisms that enhance the computational performance of our formulations. All the improvements reported in this section are novel and apply to NDPVBC.

4.1 Lifting the length-bounded cut constraints

For a hop-limit h, a lbcut destroys all paths of at most h hops. Observe that a lbcut of a hop-limit h is also a lbcut for any hop-limit $g \in \{1,2,...,h-2,h-1\}$. We then have $\Gamma^p_{rg} \subseteq \Gamma^p_{rh}$ for $0 \leq g \leq h$. Similarly, we have $\Gamma^b_{rg} \subseteq \Gamma^b_{rh}$ for $0 \leq g \leq h$. Therefore, constraints (18) and (19) can be strengthened as follows:

$$\sum_{e \in S} x_e \geq \sum_{g=H^{min}_r}^{h} z_{rg} \qquad S \in \Gamma^p_{rh}, h \in H^p_r, r \in R. \tag{20}$$

$$\sum_{e \in S} x_e \geq \sum_{g=H^{min}_r}^{h} \sum_{k \in K} k\, y_{rgk} \qquad S \in \Gamma^b_{rh}, h \in H^b_r, r \in R. \tag{21}$$

Note that variables z_{rg} for $g = H^{min}_r, \ldots, h-1$ are added to the right-hand-size of constraints (18) in order to obtain constraints (20). Similarly, $k\, y_{rgk}$ terms are added to constraints (19) to obtain constraints (21). With the lifted constraints, we introduce model M2 as follows:

$$\text{(M2) maximize} \quad \sum_{r \in R} \sum_{h \in H^p_r} F_{rh1}\, z_{rh} + \sum_{r \in R} \sum_{h \in H^b_r} \sum_{k \in K} F_{rhk}\, y_{rhk} \tag{2}$$

subject to

(9)−(15), (20)−(21).

4.2 Variable elimination

The number of y_{rhk} variables in M2 model can grow very large. However, some of these variables can be eliminated by exploiting the network topology and the available budget. To this end, we need the following definitions. Let P^b_{rh} to be the set of edges on the minimum cost path connecting the O-D pair of demand r and $\beta^b_{rh} = \sum_{e \in E^b_{rh}} c_e$ be total the cost of selecting all the edges in graph G^b_{rh}. We also define $E^b_{rh}(i)$ as the set of edges $e \in E^b_{rh}$ that are incident to vertex $i \in V^b_{rh}$.

Lemma 1. *Consider an optimal solution* $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ *of M2. For demand* $r \in R$*, there exists some* $k_r^* \in K$ *for which* $\sum_{e \in S} x_e^* \geq k_r^*$ *for all* $S \in \Gamma_{rh}^b$ *with at least one lbcut* $\hat{S} \in \Gamma_{rh}^b$ *with* $\sum_{e \in \hat{S}} x_e^* = k_r^*$.

Proof. The result follows directly from the objective function (3) of model M2 and the constraints (21). □

Observe that, in Lemma 1, the value of k_r^* is induced by the values of y^* variables in the solution. Therefore, we have $SL_r = k_r^* - 1$. In the sequel, we build bounds on this k^* value.

Remark 1. *Due to Menger's theorem (Menger 1927), the* k_r^* *in Lemma 1 represents the maximum number of edge-disjoint paths that can exist between the O-D pair of demand r. Note that these paths are not constrained by any hop limit.* □

Lemma 2. *For* $r \in R$*, we have* $k_r^* \leq \left\lfloor \frac{\beta_{rh}^b}{\sum_{e \in P_{rh}^b} c_e} \right\rfloor$.

Proof. Since P_{rh}^b represents the set of edges on the minimum cost path connecting the O-D pair of demand r and β_{rh}^b represents the total the cost of selecting all the edges in graph G_{rh}^b, the result follows. □

This upper bound can be improved as follows:

Lemma 3. *For* $r \in R$*, we have* $k_r^* \leq min\left\{ |E_{rh}^b(o_r)|, |E_{rh}^b(d_r)|, \left\lfloor \frac{min\{B, \beta_{rh}^b\}}{\sum_{e \in P_{rh}^b} c_e} \right\rfloor \right\}$.

Proof. The edge sets $E_{rh}^b(i)(o_r)$ and $E_{rh}^b(i)(d_r)$ are lbcuts in Γ_{rh}^b since their removal destroys all connectivity between the O-D pair for demand $r \in R$. Therefore, $k_r^* \leq min\{|E_{rh}^b(o_r)|, |E_{rh}^b(d_r)|\}$. Additionally, observe that the maximum cost of edges selected in graph G_{rh}^b is $min\{B, \beta_{rh}^b\}$. The result then follows with Lemma 2. □

Lemma 3 leads to the following proposition.

Proposition 2. *We have*

$$y_{rhk} = 0 \text{ for all } r \in R, h \in H_r^b, k \in K : k > min\left\{ |E_{rh}^b(o_r)|, |E_{rh}^b(o_r)|, \left\lfloor \frac{min\{B, \beta_{rh}^b\}}{\sum_{e \in P_{rh}^b} c_e} \right\rfloor \right\}. \tag{22}$$

Proof. The result directly follows from Lemma 3. □

To facilitate the presentation, we refer to (22) as VI-1.

4.3 Knapsack-type valid inequalities

Observe that when eliminating variables using Proposition 2, we explicitly consider the edges on the minimum cost paths in a graph G^b_{rh} of demand $r \in R$. In the next valid inequality, we also account for the edges that are not considered in Proposition 2.

Proposition 3. *The following is a valid inequality for M2 formulation.*

$$\sum_{e \in E \setminus E^b_{rh}} c_e x_e + \sum_{k \in K} \pi_{rhk} y_{rhk} \leq B \qquad h \in H^b_r, r \in R. \tag{23}$$

where

$$\pi_{rhk} = \begin{cases} k \sum\limits_{e \in P^b_{rh}} c_e, & k \leq min \left\{ |E^b_{rh}(o_r)|, |E^b_{rh}(d_r)|, \left\lfloor \dfrac{min\{B, \beta^b_{rh}\}}{\sum_{e \in P^b_{rh}} c_e} \right\rfloor \right\} \\ M, & otherwise \end{cases}$$

and M is a big number.

Proof. Due to Proposition 2 and $P^b_{rh} \subset E^b_{rh}$, we have $\sum_{k \in K} \pi_{rhk}\ y_{rhk} \leq \sum_{e \in E^b_{rh}} c_e x_e$ for $r \in R$ and $h \in H^b_r$. Since $\sum_{e \in E} c_e x_e = \sum_{e \in E^b_{rh}} c_e x_e + \sum_{e \in E \setminus E^b_{rh}} c_e x_e$, we have $\sum_{e \in E} c_e x_e \geq \sum_{k \in K} \pi_{rhk}\ y_{rhk} + \sum_{e \in E \setminus E^b_{rh}} c_e x_e$. As $\sum_{e \in E} c_e x_e \leq B$, the result follows. □

We refer to (23) as VI-2.

4.4 Generalized knapsack-type valid inequalities

The constraints (23) can be generalized as in the following proposition.

Proposition 4. *The following is valid for M2 formulation:*

$$\sum_{e \in E \setminus (\bigcup_{h \in H^b_r} E^b_{rh})} c_e x_e + \sum_{h \in H^b_r} \sum_{k \in K} \pi_{rhk} y_{rhk} \leq B \qquad r \in R. \tag{24}$$

Proof. Recall that the constraints (10) ensure that the backup paths for demand r are built for a single hop value $h \in H^b_r$. Therefore, the second term in the constraint will utilise a part of the budget B and give the same value as in the second term of the constraint (23). As the second term involves edges from $\cup_{h \in H^b_r} P^b_{rh}$ and $P^b_{rh} \subset E^b_{rh}$ for all $h \in H^b_r$, the remaining budget can be invested on the edges that are disjoint to the minimum cost paths of the second term. This generalizes the constraints (23) and hence, the validity of (24) holds.

□

We refer to (24) as VI-3.

5 The Separation Problem

Consider a given solution vector $\hat{\mathbf{x}} \in \mathbb{R}^{|E|}, \hat{\mathbf{y}} \in \mathbb{R}^{|R|\times|H_r^b|\times|K|}$ and $\hat{\mathbf{z}} \in \mathbb{R}^{|R|\times|H_r^p|}$ of NDPVBC. Then the separation problem for the constraints (20) and (21) either identifies a lbcut of a given weight in the subgraph induced by the solution vector $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$, or concludes that none exist. We will carry out the separation in the following ways:

- For $r \in R$ and $h \in H_r^p \setminus H_r^b$, the constraint (20) is separated by a lbcut $S \in \Gamma_{rh}^p$ with $\sum_{e\in S} \hat{x}_e < \sum_{g=H_r^{min}}^{h} \hat{z}_{rg}$.
- For $r \in R$, $h \in H_r^b \setminus H_r^p$ and $k \in K$, the constraint (21) is separated by a lbcut $S \in \Gamma_{rh}^b$ with $\sum_{e\in S} \hat{x}_e < \sum_{g=H_r^{min}}^{h} \sum_{k\in K} k\, \hat{y}_{rgk}$.
- For $r \in R$, $h \in H_r^p \cap H_r^b$ and $k \in K$, the constraints (20) and (21) are simultaneously separated by a lbcut $S \in \Gamma_{rh}^p \cup \Gamma_{rh}^b$ with $\sum_{e\in S} \hat{x}_e < \max\{\sum_{g=H_r^{min}}^{h} \hat{z}_{rg}, \sum_{g=H_r^{min}}^{h} \sum_{k\in K} k\, \hat{y}_{rgk}\}$. In this case, the lbcut has to be detected only once for both primal or backup graphs. It is obvious $\sum_{g=H_r^{min}}^{h} \hat{z}_{rg} < \sum_{g=H_r^{min}}^{h} \sum_{k\in K} k\, \hat{y}_{rgk}$, if the constraints (10) are binding at optimality. Thus, adding a violated constraint (21) implies that the constraint (20) is also respected in the model M2. Hence, we consider the maximum of $\left(\sum_{g=H_r^{min}}^{h} \hat{z}_{rg}\right)$ and $\left(\sum_{g=H_r^{min}}^{h} \sum_{k\in K} k\, \hat{y}_{rgk}\right)$. Additionally, since primal lbcuts are limited in number, so during separation, it is possible that the size of the lbcut may exceed the minimum of RHS of constraints (20) and (21), but it may be less than the maximum of RHS of constraints (20) and (21). This justifies the use of the maximum operator in the separation. We call this separation as "common" separation.

In the subsequent sections, we discuss an integer programming (IP) model and various algorithms to find the minimum-weight lbcut, followed by the details on the implementation of the separation algorithm in our branch-and-cut framework.

Note that the separation problem involves detecting a lbcut with the smallest weight possible. Since finding lbcuts is NP-hard when hop-limit $h \geq 4$, the separation problem is itself NP-hard. Thus, the separation is carried out in a heuristic fashion, by prematurely terminating a lbcut detection technique. This yields a lbcut of a suboptimal weight. On the other hand, exact separation will involve running a model or an algorithm without any time-limit. This exact separation is essential as it will guarantee the optimality of the separation problem when heuristic algorithms fails to detect any cuts.

5.1 Length-bounded cuts for paths of at most three hops

Paths of at most three hops have a unique structure such that all the vertices are always incident to either the origin or destination vertices of a connected O-D pair. Exploiting this trait, Mahjoub and McCormick (2010) built a linear-time network transformation procedure in which an application of the maximum flow minimum cut theorem on the transformed graph condenses to finding a minimum weight lbcut in the original graph. Note that as the identified cut is a lbcut, this method provides an exact separation for the constraints (20) and (21) when the hop-limit is at most 3 hops, otherwise it is a heuristic in the separation problem, which has been briefly described in the next section. We refer to this technique as lbcut3($\hat{G}$), when solved on an input directed graph $\hat{G}$.

5.2 Minimum cut heuristic

In a graph, all the minimum-weight cuts are essentially the lbcuts. Therefore, a lbcut can be identified by running any classical minimum graph cut finding algorithm as a heuristic to detect the violations in constraints in both fractional and integer separation.

5.3 An algorithm for identifying the length-bounded cuts of any hop-limit

In a given directed graph, the algorithm of Golovach and Thilikos (2011) can detect a lbcut of at most a given size or conjecture that no such cut exists. In this section, we will present and discuss the weighted version of their algorithm with certain modifications to suit our requirements. Consider a graph $\hat{G} = (\hat{V}, \hat{A})$, an O-D pair r, an integer hop-limit h, a set of arcs X and the vectors $\hat{x}^{|\hat{A}|}$, $\hat{z}^{|H_r^p|}$ and $\hat{y}^{|H_r^b|\times|K|}$, representing the edge, primal graph and backup graph weights respectively. Then a lbcut $S \subset \hat{G}$ can be identified that can destroy all the paths of at most h arcs connecting r such that the total weight of edges in S, $w(X)$ is strictly less than R_{cut}, where R_{cut} represents the RHS of the constraints (20) or (21) and it varies depending on the value of hop-limit h. The R_{cut} has to be computed as follows:

$$R_{cut} = \begin{cases} \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^p} \hat{z}_{rg}, & h \in H_r^p \setminus H_r^b \\ \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^b} \sum_{k \in K} k\, \hat{y}_{rgk}, & h \in H_r^b \setminus H_r^p \\ \max \left\{ \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^b} \sum_{k \in K} k\, \hat{y}_{rgk}, \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^p} \hat{z}_{rg} \right\}, & h \in H_r^p \cap H_r^b \end{cases}$$

The details of the algorithm have been presented as a pseudo-code in Algorithm 1. The algorithm initiates by computing $w(X)$, followed by the value comparison between $w(X)$ and R_{cut} and terminates if $w(X)$ is at least R_{cut}. Otherwise, the shortest path connecting r is computed in the subgraph $\hat{G}\backslash X$, which is devoid of any edges in X. In case the length of the path is at most h hops, it implies that X is not a feasible lbcut since by Proposition 1, a lbcut must contain at least one edge of the path sp. Therefore, the arcs of sp are iteratively appended to the set X until the value R_{cut} exceeds the set's weight $w(X)$ and thereby, making set X a feasible lbcut. As the set is finite, the algorithm converges. Observe that the separation problem for the O-D pair r is solved via a single call of lbcutA($\hat{G}, r, h, \phi, \hat{x}, t$). Note that we have made two modifications to the original algorithm. First, similar to Arslan et al. (2020), the weight $w(X)$ of set X is compared with R_{cut} rather than its cardinality (line 2). Secondly, in addition to the set X, our algorithm returns the set's weight $w(X)$ (lines 2, 8 and 10). Particularly, when $h \in H_r^p \cap H_r^b$ and if a lbcut is identified, then $w(X)$ has to be compared with the summation of both primal ($\hat{z}$) weights across $h \in H_r^p \setminus H_r^b$ and the summation of backup weights ($\hat{y}$) across $h \in H_r^p \setminus H_r^b$ and $k \in K$; separately to identify potential violations of both the constraints (20) and (21).

Additionally, in the cases of networks having multiple zero-weight arcs represented by the set X_0, calling lbcutA($\hat{G}, r, h, X_0, (\hat{x}, \hat{z}, \hat{y}), t$), is more computationally effective in identifying lbcuts due to no contribution of those zero-weight arcs to the total cut weight $w(X)$. Despite being a fully polynomial time algorithm, the algorithm may stuck in case of higher hop-limits h (Arslan et al. 2020) and therefore, we will run it during the heuristic separation only within a certain time limit t (line 8). For simplicity, we will call our algorithm as lbcutA($\hat{G}, t$).

Algorithm 1 lbcutA$(\hat{G}, r, h, X, \hat{x}, R_{cut}, t)$ Algorithm

Input: r, h, $\hat{x}$, R_{cut}, t

Output: An lbcut $S \supseteq X$ of weight, $w(X) < R_{cut}$, destroying paths of length h or false if no such lbcut exists.

```
w(X) ← Σ_{(i,j)∈X} x̂_ij
if w(X) ≥ R_cut then return (NULL, NULL)
Let sp be the shortest path for r in Ĝ\X
if |sp| ≥ h+1 then return (X, w(X))
else
    for (i, j) ∈ sp do
        Let T_run be the algorithm's current running time
        if T_run > t then return (NULL, NULL)
        Y, w(Y) = lbcutA(Ĝ, r, h, X ∪ (i, j), x̂, R_cut, t)
        if Y ≠ NULL then return (Y, w(Y))
return (NULL, NULL)
```

5.4 An exact IP model for detecting length-bounded cuts

We now present a mathematical model to compute a minimum-weight lbcut. For a given directed graph $\hat{G} = (\hat{V}, \hat{A})$ with its vertex and arc sets of $\hat{V}$ and $\hat{A}$ respectively, a positive integer length bound h and an O-D pair (o_r, d_r), consider each arc $(i, j) \in \hat{A}$ has a unit length $\bar{x}_{ij} \in [0, 1]$ and an associated binary variable f_{ij}. $\hat{G}$ equals G^p_{rh} and G^b_{rh} when finding a lbcut in a primal and backup graph induced by the hop bound $h \in H^p_r \setminus H^b_r$ and H^b_r, respectively, for every $r \in R$. For every arc $(i, j) \in \hat{A}$, $f_{ij} = 1$ if and only if the arc (i, j) is an element of the minimum lbcut and arc weights $\bar{x}_{ij}$ takes values from the corresponding edge variables in either the fractional or integer solution of $\hat{x}_{ij} \in \hat{\mathbf{x}}$ in the separation problem. In addition, consider the integer variable ω_i as the shortest path from the vertex $i \in \hat{V}$ to the destination vertex d_r if and only if it will not constitute any arc, which is an element of the lbcut. Otherwise, ω_i is a large number M.

$$\text{(lbCutM) minimize} \quad \sum_{(i,j)\in A} \bar{x}_{ij} f_{ij} \tag{25}$$

subject to

$$\omega_{d_r} = 0 \tag{26}$$

$$\omega_i \leq \omega_j + 1 + M f_{ij} \quad (i,j) \in \hat{A}, \tag{27}$$

$$\omega_{o_r} \geq h + 1 \tag{28}$$

$$\omega_i \geq 0 \qquad i \in \hat{V}, \tag{29}$$

$$f_{ij} \in \{0,1\} \qquad (i,j) \in \hat{A}. \tag{30}$$

The objective function (25) minimizes the total weight of the lbcut. Constraints (26) and (27) ensure that shortest paths from every vertex $i \in \hat{V}$ to d_r are correctly computed by the ω_i variables. Constraint (28) restricts the O-D pair distance to at least $h+1$ hops. Constraints (29) and (30) define the domain space for the variables. We refer to this mathematical model as lbcutM($\hat{G}, t$), where t is the time-limit, and it will be used during both heuristic and exact separation to detect lbcuts in the graph $\hat{G}$. In comparison to the other discussed algorithms in the Section 5.1−Section 5.3, the lbcutM($\hat{G}$, t) contributes the highest number of cuts per instance in the separation problem as showcased by Arslan et al. (2020).

5.5 Strengthening the detected length-bounded cuts

Note that the lbcuts identified through the exact model or the algorithms are potentially vulnerable to including many zero-weight arcs. Koch and Martin (1998) has demonstrated that the cardinality of the minimum weight cuts can be effectively minimized, by assigning a very small ε value to zero-weight arcs at the cost of very little optimality spoilage and additional computational time. We will thus refer the techniques: lbcut3 and minCut as ε-lbCut3 method and ε-MinCut heuristic respectively. Although for the lbcuts derived from the lbcutM model and lbcutA algorithm, a modified version of the lbcutM model was developed in which the objective function coefficients of the arc variables available in the generated lbcut are equated to one, while the rest of the coefficients are assigned a large number. Solving this model, for a small time-limit of 0.1 seconds has been proven to be very effective in strengthening the lbcuts (Arslan et al. 2020). We call this technique as CutEnhancer($Cutset$), where $Cutset$ contains the identified lbcuts in the separation problem and t is the time-limit.

5.6 The extended branch-and-cut framework

Our framework is an extended version of the branch-and-cut framework, developed by Arslan et al. (2020), which was applicable for $|H_r^p| = |H_r^b| = |K| = 1$ only. The pseudo-code for the separation algorithm to detect any violated solutions in a node in the branch-and-bound tree is presented in Algorithm 2.

In every callback, the constraints (20) or (21) are dynamically separated by identifying cuts (LHS of constraints (20) or (21)) and detecting their violations for any potential integer solution for every hop-limit $h \in H_r^p \cup H_r^b$. When $h \leq 3$, the computationally effective lbcut3 provides an exact separation (lines 13−15), while for $h > 3$, heuristic separation (lines 16−22) is achieved via the implementation of the ε-MinCut along with the hybrid application of the model lbcutM and algorithm lbcutA with their respective time-limits of 0.1 s and 0.016 s (Arslan et al. 2020). Due to the existence of potentially many fractional solutions, the separation of the fractional solutions was carried out only at the root node until the improvement in LP relaxation is at least the tailing-off threshold of 0.01% (Sherali and Driscoll 2000) in the last three callbacks. Otherwise, we resort to branching when any algorithm gets stuck. Moreover, the violated constraints (20) or (21) added at the root node will be applicable throughout the branch-and-bound tree. As $H_r^p \subseteq H_r^b$, so when $h \in H_r^p \cap H_r^b$, every algorithm will be executed only once as indicated in line 9, in which c stands for common separation; rather than separately for primal ('p') and backup ('b') graphs and thus, every algorithm or model can yield at most two cuts in a single callback. Finally, as the exact separation (lines 24−29) when $h > 3$ is NP-hard, we exit the loop (lines 25−29) as quickly as a violation is detected since the identification of cuts for the remaining O-D pairs for the hop-limit h can be carried out within the heuristic separation (lines 16−22) in the next callback. Note that the separation algorithm can be modified to identify primal and backup cuts separately as summarized in the Algorithm 3 in the Appendix C.

Algorithm 2 Separation Algorithm

Input: Demand Set R, the limits (Δ^p, Δ^b, Δ^k), κ, primal $\hat{z}_{rh}$ and backup $\hat{y}_{rhk}$ weights during a callback

Output: A set of violated constraints

$H \leftarrow \{H_R^{min} + \Delta^b, H_R^{min} + \Delta^b - 1, \ldots, \min_{r \in R}\{H_r^{min}\}\}$
for $h \in H$ **do**
  for $r \in R$ **do**
    if $h \geq H_r^{min}$ **and** $h > H_r^{min} + \Delta^p$ **then**
      $i_{rh} \leftarrow b$ // Identify Cut for backup ('b') path(s) only
      $G_{rh} \leftarrow G_{rh}^b$ // Utilise backup graphs for backup ('b') cuts only
      $R_{cut}^{i_{rh}} \leftarrow \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^b} \sum_{k=\kappa}^{\kappa+\Delta^k} k\, \hat{y}_{rhk}$ // Get the RHS value of the Cut
    else if $h \geq H_r^{min}$ **and** $h \in H_r^b \cap H_r^p$ **then**
      $i_{rh} \leftarrow c$ // Identify Cuts (if possible) for both 'p' and 'b' paths
      $G_{rh} \leftarrow G_{rh}^b$ // Here, $G_{rh}^b = G_{rh}^p$, so 'b' graphs used for both 'p' and 'b' cuts
      $R_{cut}^{i_{rh}} \leftarrow \max\left\{ \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^p} \hat{z}_{rh}, \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^b} \sum_{k=\kappa}^{\kappa+\Delta^k} k\, \hat{y}_{rhk} \right\}$
    if $R_{cut}^{i_{rh}} > 0$ **then** // Cuts with $R_{cut}^{i_{rh}} = 0$, are satisfied by default
      if $h \leq 3$ **then**
        /* Exact Separation */
        $CutSet^{i_{rh}}$ += ε-lbcut3(G_{rh}) // Section 5.1
      else
        /* Heuristic Separation */
        $CutSet^{i_{rh}}$ += ε-minCut(G_{rh}) // Section 5.2
        $tempCutSet^{i_{rh}}$ += lbcutA$(G_{rh}, 0.016)$ // Section 5.3
        $tempCutSet^{i_{rh}}$ += lbcutM$(G_{rh}, 0.1)$ // Section 5.4
        $tempCutSet^{i_{rh}} \leftarrow$ CutEnhancer$(tempCutSet^{i_{rh}}, 0.1)$ // Section 5.5
        $CutSet^{i_{rh}} \leftarrow CutSet^{i_{rh}} \cup tempCutSet^{i_{rh}}$
  /* Exact Separation */
  if $\bigcup_{i_{rh} \in \{c,b\}} CutSet^{i_{rh}} = NULL$ **then**
    for $r \in R$ **do**
      if $h \geq H_r^{min}$ **and** $h > 3$ **then**
        if $R_{cut}^{i_{rh}} > 0$ **then** // Cuts with $R_{cut}^{i_{rh}} = 0$, never violates
          $CutSet^{i_{rh}}$ += lbcutM$(G_{rh}, 10800)$ // Section 5.4
          $CutSet^{i_{rh}} \leftarrow$ CutEnhancer$(CutSet^{i_{rh}})$ // Section 5.5
return $CutSet^c \cup CutSet^b$

6 Computational Study

We now present the data in Section 6.1, the experimental design in Section 6.2, and the results in Sections 7. All the models and algorithms have been implemented using Python 3.11 and Gurobi 11.0.0 (Python API) in a Linux environment. All the experiments are conducted on the Béluga cluster of the Digital Research Alliance of Canada, using a single thread and 20 GB of RAM. A three-hour time limit is set for every experiment conducted.

6.1 Data

We use the NDPVC instances created by Gouveia and Leitner (2017). There are a total of 50 instances and the details are presented in Appendix B. Each dataset belongs to one of the 4 groups C, D, E and RE. The C and D groups have a grid structure while the remaining two are randomly generated. Each group has a distinct way of selecting O-D pairs and assigning costs to the edges. We refer the reader to Gouveia and Leitner (2017) for the generation scheme of these group instances.

6.2 Experimental design

Our experimental design consists of three groups of experiments, totaling 94,932 instances. The first group of experiments consists of the five networks in the "RE-1" group to compare the computational performances of models M1 and M2. We have the primal hop-limit range as $H_r^p = \{H_r^{min}, H_r^{min}+1, \ldots, H_R^{min}+\Delta^p\}$, the backup hop-limit range as $H_r^b = \{H_r^{min}, H_r^{min}+1, \ldots, H_R^{min}+\Delta^b\}$ and the survivability set as $K = \{\kappa, \kappa+1, \ldots, \kappa+\Delta^s\}$, for every problem instance.

For every network topology, we determine a minimum and a maximum budget level and for a given parameter settings. Let B_0 be the network design cost of servicing only the primal paths for every demand $r \in R$. The minimum budget level of B_0 implies that the corresponding NDPVBC instance is feasible. In order to determine a maximum budget level, we first consider that the budget is infinite (i.e., all the edges in the network can be selected). Solving the corresponding model for a given n value yields a unique network configuration that has the maximum level of service that can be provided per demand for a given n parameter. This maximum level of service can be either the best QoS or SL or their balanced mix depending

on the value of n. Observe that not all the edges are required when providing this service. So, let B_n^{max} be this minimized cost of providing this maximum service level for a given n value (Note that the optimal solution will not change beyond this budget level of B_n^{max}). We then define $B^{max} = \max_{n\in\{0.25,1.0,4.0\}}\{B_n^{max}\}$, as the universal maximum network design cost of providing the maximum level of service. Let Δ^B be the required number of budget levels between B_0 and B^{max} which is used to generate the budget set: $B_{set} = \{B_0, B_1, B_2, \ldots, B_{(\Delta^B+1)}\}$ of equally spaced budget levels. Note that $B_{(\Delta^B+1)} = B^{max}$ and $|B_{set}| = \Delta^B + 2$. Each budget level in B_{set} represents a unique NDPVBC instance, which is solved for any parameter settings chosen for a given network topology.

Due to the enumerative nature of model M1, the number of variables can grow very large. Table 1 presents the minimum, average and maximum number of edge sets associated with the "RE-1" networks for different values of Δ^b and Δ^s. As variables $v_{ij}^{rhC_k}$ in model M1 grow proportionally to the size of these edge sets, so, for testing purposes, the survivability set K was defined for $\Delta^s \leq 1$ and we set the highest primal and backup limits as $\Delta^p \leq \Delta^b \leq 1$ respectively, to define the hop-limit ranges H_r^p and H_r^b. The networks were designed for all values of n across a total of $\Delta^b + 2$ budget levels. This resulted in a total of 6 parameter settings per NDPVBC instance with $\Delta^B = 10$, $\Delta^p \in \{0,1\}$, $\Delta^b \in \{0,1\}$, $\Delta^s \in \{0,1\}$ and $n \in \{0.25, 1.0, 4.0\}$. Consequently, 1,146 experiments per model, were carried out over the "RE-1" network topologies to do the performance testing of both M1 and M2.

Table 1: Minimum, average, and maximum number of edge sets of the networks of "RE-1" considered in the problem for different values of Δ^s and Δ^b.

	Δ^b		
Δ^s	**0**	**1**	**2**
0	(101, 632.8, 1450)	(371, 1,288.4, 2,442)	(914, 2,206.6, 3,574)
1	(530, 15,357.8, 47188)	(4,815, 41,619.6, 97,859)	(20,472, 87,210.2, 162,733)
2	(2049, 317,377.2, 350,425)	(49,233, 1,074,770.20, 262,2544)	(35,0425, 2,622,544, 5,339,733)
3	(6,203, 5,696,759.2, 23,595,962)	(414,344, 22,961,378.8, 67,966,639)	(4,811,708, 63,216,803.4, 136,718,994)
4	(14,960, 90,096,126.4, 402,115,777)	(2,920,892, 416,012,472.6, 1,322,169,428)	(54,708,766, 1,263,966,742.2, 2,860,503,126)

The second group of experiments were conducted on the "RE-3" networks to determine the best valid-inequality (VI) setting for the model M2 using the parameters in Table 2. With $\Delta^p \leq \Delta^b$, both the primal Δ^p and backup limits Δ^B were tested from 0 to 2, while Δ^s varied up to 4. Parameters Δ^B and n had the same values as in the first group of experiments. This

resulted in a total of 90 parameter settings per NDPVBC instance, which along with 8 VI settings yielded a total of 42,624 runs for the second group of experiments.

Table 2: General parameter settings for the models in the experimental runs.

Parameter	**Possible values**
Δ^B	10
Δ^p	{0, 1, 2}
Δ^b	{0, 1, 2}
Δ^s	{0, 1, 2, 3, 4}
n	{0.25, 1.0, 4.0}

Finally, using the best valid-inequality setting and the 90 parameter settings of Table 2, a total of 53,208 experimental runs were conducted over the remaining 40 networks in the third and final group of experiments.

7 Results and Discussion

We first present the mathematical definitions of the key performance indicators (KPIs) characterising the metrics of network design. We then present the computational results and discuss the role of budget over and the tradeoff between the QoS and the SL.

7.1 Key performance indicators

Let $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ be the optimal solution vector of a NDPVBC instance. Then for an $r \in R$ the following local-level KPIs can be defined as an output of the optimal solution of our model:

$$SL_r = \sum_{r \in R} \sum_{h=H_r^b} \sum_{k \in K} (k-1)\, y^*_{rhk} \tag{31}$$

$$QoS_r^b = \sum_{r \in R} \sum_{h \in H_r^b} \sum_{k \in K} h\, y^*_{rhk} \tag{32}$$

Here, SL_r represents the SL of an O-D pair r. Note that SL_r indicates that r can survive the failure of at most $(\kappa + \Delta^s - 1)$ edges if the backup paths are constructed for a given budget. Otherwise, it will lose connectivity as soon as an edge fails. Thus, SL_r can range from 0 to $(\kappa + \Delta^s - 1)$ and a higher value of SL_r implies a better survivability level of an O-D pair r.

Similarly, the size of the backup paths for demand r is defined as QoS_r^b. Since it is not necessary to have backup paths, particularly when the budget is very tight, it can also vary between 0 and $H_R^{min} + \Delta^b$. However, a smaller value implies a better QoS.

With the local-level definitions, the metrics can be defined for the given network at a global level:

$$SL_R = \frac{\sum_{r \in R} SL_r}{|R|} \tag{33}$$

$$QoS_R^b = \frac{\sum_{r \in R} QoS_r^b}{|R|}. \tag{34}$$

The global level metrics are the averages of their local counterparts.

7.2 Computational performance comparison of the models

Table 3 presents the experimental results of the first group comparing the solution performance of the M1 and M2 models. The first column represents the setting number, and the second column reports the model solved. Setting number 0 corresponds to solving the M1

with default Gurobi settings while setting number 1 corresponds to the M2 without any VIs. The third column reports the status of the solution. The status "Optimal" implies that the model is solved to optimality, the status"Memory/Feasible" signifies that the model found a feasible solution but it was terminated prematurely due to a memory problem. "Memory/Unknown" is similar to "Memory/Feasible" except that no feasible solution was found. Note that there is no "Infeasible" status since within the budget B_{set}, the NDPVBC is always feasible. The fourth and fifth columns report the average solution time in seconds and the average optimality gap (%) as reported by Gurobi. The last two columns provide information on the number and percentage of instances solved under each optimization status for each setting. Among the 1,146 instances, model M1 can solve 685 instances to optimality, while it encounters memory problems in the remaining 428 instances. On the other hand, M2 generates and solves all of the instances to optimality within an average of 4.9 seconds. Therefore, we conclude that M1 is computationally inferior to model M2.

Table 3: First group experiment results comparing the solution performance of M1 and M2 models without any valid inequality.

Setting	Model	Optimization Status	Avg. Time (s)	Avg. Gap (%)	# Instances
0	M1	Optimal	121.0	0.00	685
		Memory/Feasible	908.0	21.29	33
		Memory/Unknown	538.2	-	428
1	M2	Optimal	4.9	0.00	1,146
		Memory/Feasible	0.0	0.00	0
		Memory/Unknown	0.0	0.00	0

7.3 The impact of valid inequalities on the computational performance of M2 model

The second group of experiments is designed to measure the impact of VIs on the computational performance of the M2 model. The results of the eight different VI settings are reported in Table 4. With the first column representing the setting number, the next three columns indicate whether VIs of various forms have been incorporated in the model M2 or

not. The fifth and sixth columns show the total number of instances and the percentage of instances solved to optimality respectively. The seventh column reports the maximum optimality gap in percentage. Finally, the average solution times in seconds are reported in the eighth column. Setting 3, which involves VI-2 only (constraints (23)), provides an average computational improvement of 21.64% with respect to setting 1, which corresponds to model M2 without any VIs, and setting 3 is the best-performing setting among the eight options. As such, setting 3 is used in the rest of the experimental runs.

Table 4: Second group experiment results comparing the solution performance of M2 model with different settings.

Setting	VI-1	VI-2	VI-3	# Instances	Solved Instances (%)	Maximum Optimality Gap (%)	Average Time (s)
1	0	0	0	5,328	100.00	0.00	29.2
2	1	0	0	5,328	99.98	1.71	30.1
3	0	1	0	5,328	100.00	0.00	22.9
4	0	0	1	5,328	99.98	0.79	28.1
5	1	1	0	5,328	100.00	0.00	23.7
6	0	1	1	5,328	100.00	0.00	26.0
7	1	0	1	5,328	100.00	0.00	26.3
8	1	1	1	5,328	99.98	1.71	27.0

Table 5 presents the results of the second group experiments conducted on "RE-3" networks with the best performing M2 model (3rd setting) for different n values and average budget levels. In general, the problem takes longer times to solve when $n < 1$. The average solution times are 55.0 s, 9.6 s and 4.0 s for $n = 0.25$, $n = 1.0$, and $n = 4.0$ respectively, which showcases that it grows exponentially as n decreases. Moreover, computational times vary in a concave fashion with respect to increasing budget levels. Excluding the budget B_0 in the case of $n = 0.25$, the computational time can nearly double from an average of 66.1s to 120.4 s, to solve an instance to optimality, until the threshold average budget range of $[1051.78, 1216.95]$ is hit, which is at least $[2.7, 3.1]$ times the minimum budget B_0. This can be explained as follows. Recall that a backup path is not always required as per the constraints (10). So, when the budget is very tight, the constraints (10) remain non-binding

for most of the demands, leaving O-D pairs connected through primal paths only, which are easier to identify as there are limited primal graph lbcuts to separate in comparison to backup lbcuts. Although, as the budget is relatively increased, the survivability aspect of the network can be prioritised and therefore, the network topology is designed to be robust against a larger number of edge failures. This leads to a larger value of k in the RHS of the constraints (21), which in turn increases the number of constraints to separate; yielding higher solution times. Beyond the threshold budget range of [1051.78, 1216.95], the solution time can sharply drop by a minimum factor of 1.6. This is also expected since the budget constraint (11) becomes increasingly redundant with increasing budget levels; thereby relaxing the formulation M2. This concave nature of solution time with respect to increasing average budget levels is also consistent for both $n = 1.0$ and $n = 4.0$. However, the instances in these cases are relatively solved faster due to diminishing prioritization of survivability as n increases.

Table 5: Second group experiment results of best performing M2 model (3rd setting) for "RE-3" networks with different average budget levels and n settings.

Average		Average Time (s)			
		n			
Budget Levels	**# Instances**	**0.25**	**1.0**	**4.0**	**Average Time (s)**
$B_0 = 391.93$	444	4.0	3.6	3.4	3.7
$B_1 = 556.53$	444	66.1	13.5	5.1	28.2
$B_2 = 721.62$	444	104.0	17.6	5.1	42.2
$B_3 = 886.71$	444	97.0	16.5	4.6	39.4
$B_4 = 1051.78$	444	120.4	14.9	4.2	46.5
$B_5 = 1216.95$	444	117.4	12.1	3.9	44.5
$B_6 = 1381.92$	444	74.2	8.6	3.9	28.9
$B_7 = 1547.09$	444	41.8	7.4	4.0	17.7
$B_8 = 1712.16$	444	16.4	6.3	3.9	8.9
$B_9 = 1877.25$	444	8.6	5.7	3.8	6.0
$B_{10} = 2042.34$	444	6.3	5.2	3.6	5.1
$B_{11} = 2207.82$	444	4.4	3.5	3.1	3.7
Average/Total	5,328	55.0	9.6	4.0	22.9

The primal and backup hop-limits, Δ^p and Δ^b, and the survivability limit Δ^s have a significant impact on solution efficiency. Table 6 presents the results of the second group of experiments with the best performing M2 model for different Δ^p, Δ^b and Δ^s settings on "RE-3" networks across different values of n. The maximum average time for $\Delta^s = 4$ is 311.7 s. Clearly, the average time increases as Δ^s takes higher values since there will be multiple edge failure instances to consider. The same scenario exists for larger values of Δ^b, which expands the hop-limit range set H_r^b. As finding a lbcut becomes increasingly harder due to its NP-hard nature for $h \geq 4$, eventually the solution time increases significantly. In fact, the solving instances with $\Delta^b = 2$ and $\Delta^s = 4$ on average can take a minimum of 180.2 s which is at least a 450 times multiplier of the average time of 0.4 s in the case of $(\Delta^b, \Delta^s) = (0,0)$. Another inference can be drawn that when $\Delta^p = \Delta^b$, there can be some minor dampening

effect since in such cases the detected lbcuts are applicable to both primal and backup paths. For example, the instances with settings $n = 0.25$, $\Delta^s = 3$ and $\Delta^b = \Delta^p$ have average solution times as $\{1.0, 19.8, 102.2\}$ as Δ^p takes values from $\{0, 1, 2\}$. As soon as there is a gap between Δ^b and Δ^p, the average solution time can increase significantly to 219.6 s, which occurs when $\Delta^b - \Delta^p = 1$ and $\Delta^p = 1$. This pattern is consistent for all n and $\Delta^b = \Delta^p$ values, with an exceptional case when $(\Delta^p, \Delta^b, \Delta^s) = (2, 2, 4)$ takes more time to solve than instances with setting $(\Delta^p, \Delta^b, \Delta^s) = (0, 2, 4)$ for $n = 0.25$.

Table 6: Second group experiment results of best performing M2 model (3rd setting) for different Δ^p, Δ^b, Δ^s and n settings on "RE-3" networks.

				Average Time (s)			
				n			
Δ^p	Δ^b	Δ^s	**# Instances**	**0.25**	**1.0**	**4.0**	**Average Time (s)**
0	0	0	180	0.4	0.3	0.2	0.3
0	0	1	180	0.7	0.4	0.4	0.5
0	0	2	180	0.9	0.4	0.3	0.6
0	0	3	180	1.0	0.5	0.3	0.6
0	0	4	180	1.2	0.5	0.3	0.6
0	1	0	180	1.6	1.2	0.9	1.3
0	1	1	180	7.0	2.1	1.0	3.4
0	1	2	180	18.8	3.6	0.9	7.8
0	1	3	180	47.3	5.8	1.0	18.1
0	1	4	144	35.0	4.9	0.8	13.6
0	2	0	180	3.5	3.1	2.5	3.0
0	2	1	180	14.2	5.1	2.7	7.3
0	2	2	180	45.2	7.7	2.6	18.5
0	2	3	180	122.5	13.3	2.5	46.1
0	2	4	180	180.2	22.3	2.4	68.3
1	1	0	180	5.3	4.3	3.2	4.3
1	1	1	180	9.0	5.0	3.6	5.9
1	1	2	180	12.6	7.0	4.2	8.0
1	1	3	180	19.8	8.6	4.1	10.9
1	1	4	180	47.7	11.7	4.3	21.2
1	2	0	180	15.9	7.8	4.6	9.5
1	2	1	180	35.3	9.0	4.8	16.3
1	2	2	180	97.8	16.1	5.7	39.9
1	2	3	180	219.6	21.5	5.4	82.1
1	2	4	180	311.7	29.0	5.1	115.2
2	2	0	180	28.5	13.1	10.4	17.3
2	2	1	180	30.2	15.4	11.6	18.7
2	2	2	180	49.4	17.9	12.5	26.6
2	2	3	180	102.2	22.7	11.8	45.6
2	2	4	144	214.8	29.9	12.1	85.6
Average/Total			5,328	55.0	9.6	4.0	22.9

Table 7 reports the results of the second group of experiments for each of the "RE-3" networks separately. Clearly, solving the problem on some networks can be quite challenging. In particular, network "RE50-5-0.2-4" yields the highest average solution time of 172.7 s when $n = 0.25$. Note that the last four networks have $H_R^{min} = 3$, while "RE50-5-0.2-1" has $H_R^{min} = 2$. As $\Delta^p \geq 1$, identifying lbcuts for larger h is difficult and it even becomes NP-hard since $h \geq 4$ in this case. Consequently, the last four networks performed poorly, solving instances nearly $3-20$ times the average computational speed of the first network, when averaged across all n.

Table 7: Second group experiment results of best performing M2 model (3rd setting) for every "RE-3'" network for different values of n.

		Average Time (s)			
		n			
Network	**# Instances**	**0.25**	**1.0**	**4.0**	**Average Time (s)**
"RE50-5-0.2-1"	1,080	5.7	3.0	1.1	3.3
"RE50-5-0.2-2"	1,080	23.3	7.3	4.2	11.6
"RE50-5-0.2-3"	1,044	33.5	7.2	3.4	14.7
"RE50-5-0.2-4"	1,044	172.7	21.6	6.5	66.9
"RE50-5-0.2-5"	1,080	43.3	9.3	5.0	19.2
Average/Total	5,328	55.0	9.6	4.0	22.9

7.4 The evolution of various KPIs with respect to budget

The minimum and maximum network construction costs along with backup QoS_R^b and SL_R, averaged across all the "RE-3" networks for different n values in the second group of experiments have been reported in the Table 8. It can be observed that a network without any survivability requirements can be constructed at nearly one-fifth the cost of the networks designed to survive multi-edge failures and provide connectivity at a good QoS. It is clear that when $n = 0.25$, the network comprises longer connecting paths of 3.24 hops, while O-D pairs are connected through shorter paths consisting of an average of 2.27 hops when $n = 4.0$. A reverse scenario exists for the SL_R which decreases from 2.08 to 1.35 as n varies from 0.25

to 4.0. This is also reflected through the column "Avg. Connectivity (%)", which showcases that a relatively large number of O-D pairs are connected through backup paths, in the case of robust network configurations designed by $n = 0.25$ than tighter networks of $n = 4.0$. On the other hand, $n = 1.0$ yields a balanced network design, intermediate to the designs of $n = 0.25$ and $n = 4.0$, with $SL_R = 1.68$ and $QoS_R^b = 2.82$. Interestingly, the design of the network via $n = 4.0$, which provides the best QoS, requires a monetary budget of 2,158.33, which is costlier[1] than the most resilient networks of $n = 0.25$ built at the cost of 2,104.74. This is most likely due to the involvement of costly links in the construction of shortest paths for the O-D pairs. Furthermore, it can be even more costly to build networks when QoS_R^b and SL_R have to be balanced. Additionally, it can be observed that a minor deterioration of 8% in the backup QoS_R^b can yield a significant improvement of 24% in the SL_R as n is changed from 4.0 to 1.0. However, the deterioration rate has to be nearly doubled to 14% in QoS_R^b to further improve the SL_R by the same amount of 24%, as n changed from 1.0 (balanced network) to 0.25 (more resilient networks).

Table 8: Values of different Metrics averaged across all the "RE-3" networks for different n values in the second group of experiments.

n	**Avg.** B^{min}	**Avg.** B_n^{max}	**Avg. Connectivity (%)**	**Avg.** QoS_R^b	**Avg.** SL_R^b
0.25	391.93	2,104.74	91.95	3.24	2.08
1.0	391.93	2,199.83	91.61	2.82	1.68
4.0	391.93	2,158.33	87.83	2.61	1.35
Average	391.93	2,154.33	90.50	2.88	1.82

Table 9 shows the averaged variation of the backup QoS_R^b and the SL_R across different average budget levels for all the "RE-3" networks. Note that the budget level B_0 is excluded since it is not sufficient enough to provide the backup connectivity even for at least one demand. In the case of $n = 0.25$, the backup QoS_R^b initially has an incremental trend, increasing

[1]However, it depends on the network topology on how the costs are distributed across its edges. It is possible that in some network configurations, the costs associated with O-D pair specific shortest paths are in fact cheaper than the edges of relatively longer paths connecting those O-D pairs. However in the case of "R" group, a reverse scenario exists such that longer connectivity paths are cheaper than the shortest paths, as confirmed through experimental results.

from 1.95 to 3.45 hops when the average budget nearly doubled from 559.07 to 1051.78. In the same budget range, the SL_R nearly tripled up from 0.60 to 1.94. This implies that when $B \in [559.07, 1051.78]$, the mechanism $n = 0.25$ sacrificed the QoS to strengthen the SL of the network, which was expected as the objective function maintains the tradeoff between $h \in H_r^b$ and $k \in K$. However, as the budget was relaxed, there is an opportunity to improve both the SL_R and QoS_R^b simultaneously. Consequently, both QoS_R^b and SL_R improved from 3.45 to 2.97 hops and 1.94 to 2.72 edges respectively. Similarly, in the design yielded by $n = 4.0$, the value of QoS_R^b and SL_R also increased as the budget increased until $B = 1216.95$. Although, the best SL_R in the case of $n = 4.0$ is around 2.11 edges which is 22% poorer than its value in the network design of $n = 0.25$. Although, the QoS_R of $n = 4.0$ networks are maintained at levels $16-38\%$ lower than the same in the case of $n = 0.25$. Again, $n = 1.0$ provides a balancing effect between the two metrics as evident from the averaged values of QoS_R^b and the SL_R, which are 2.83 hops and 1.68 edges and are in between the average values of their counterparts in the cases of $n = 0.25$ and $n = 4.0$. Note that a certain deterioration in the backup QoS is necessary to develop robustness in the network. The objective function will always try to strengthen both the QoS and SL for any budget level. It is the mechanism n that decides the importance of one network metric over another. Therefore, the QoS_R^b cannot be enhanced beyond a certain level.

Table 9: Values of different KPIs averaged across all the "RE-3" networks for different budget levels in the B_{set} and n values in the second group of experiments.

	n					
Average	**0.25**		**1.0**		**4.0**	
Budget Levels	**Avg.** QoS_R^b	**Avg.** SL_R	**Avg.** QoS_R^b	**Avg.** SL_R	**Avg.** QoS_R^b	**Avg.** SL_R
$B_1 = 559.07$	1.95	0.60	1.74	0.55	1.20	0.42
$B_2 = 721.62$	2.95	1.11	2.56	0.91	1.94	0.72
$B_3 = 886.71$	3.31	1.56	2.78	1.19	2.21	0.91
$B_4 = 1051.78$	3.45	1.94	2.78	1.41	2.42	1.11
$B_5 = 1216.95$	3.42	2.24	2.75	1.61	2.55	1.31
$B_6 = 1381.92$	3.34	2.49	2.74	1.82	2.50	1.40
$B_7 = 1547.09$	3.21	2.62	2.70	2.01	2.50	1.57
$B_8 = 1712.16$	3.06	2.67	2.69	2.16	2.50	1.70
$B_9 = 1877.25$	2.93	2.69	2.67	2.31	2.49	1.82
$B_{10} = 2042.34$	2.86	2.71	2.63	2.38	2.48	1.97
$B_{11} = 2207.82$	2.97	2.72	2.56	2.43	2.47	2.11
Average	3.24	2.08	2.82	1.68	2.61	1.35

7.5 Computational performance of the best performing algorithm on the complete dataset

The previous experiments were all executed on "RE-1" and "RE-3" networks from the "R" group. The third group of experiments are conducted on the complete set of 50 networks for different n, B, Δ^b, Δ^p, Δ^b and Δ^s settings, yielding a total of 53,208 instances. Table 10 shows the average solution time in seconds, the number of solved instances, the percentage of solved instances and the average gap per optimization status. Almost all instances were solved to optimality, with only 224 instances having feasible status. This yielded a total of 99.57% solved instances. The average solution time of all instances is 100.1 seconds and the average gap is 0.03%. Detailed computational results categorized by n for every SubGroup of networks are presented in Table 11. In general, the networks from the "R" group

are difficult to solve, in terms of computational time. The same results are categorized for different averaged budget levels B and n values in Table 12. Note that the algorithm performs better for smaller n values and when $B \leq 240.23$ or $B \geq 776.79$ as no feasible instances are found in this budget range.

Table 10: Overview of the computational results of the best performing algorithm on all 50 networks in the dataset in the third group of experiments.

Optimization Status	**Average Time (s)**	**# Solved Instances**	**% Solved Instances**	**Average Gap (%)**
Optimal	54.0	52,084	99.57	0.00
Feasible	TL (10800)	224	0.43	7.27
Average/Total	100.1	52,308	100.00	0.03

* Abbreviation: TL, time limit.

Table 11: Average percentage of the instances with "Optimal", and "Feasible" optimization status when solved for different datasets of networks considered in this study.

	Optimization Status					
	Optimal			**Feasible**		
Dataset	**Avg. Time (s)**	**Instances (%)**	**Avg. Gap (%)**	**Avg. Time (s)**	**Instances(%)**	**Avg.Gap (%)**
D-1	4.2	100.00	0.00	-	0.00	-
D-2	38.4	99.97	0.00	TL (10800)	0.03	4.35
E-1	8.7	100.00	0.00	-	0.00	-
E-2	136.6	99.53	0.00	TL (10800)	0.47	4.57
E-3	22.9	100.00	0.00	-	0.00	-
RE-1	12.1	100.00	0.00	-	0.00	-
RE-2	261.2	96.25	0.00	TL (10800)	3.75	7.77
RE-3	22.9	100.00	0.00	-	0.00	-
Average/Total	54.0	99.57	0.00	TL (10800)	0.43	7.27

Abbreviation: TL, time limit.

Table 12: Detailed results of the best performing algorithm on all 50 networks in the dataset in the third group of experiments.

Average		n			
Budget Levels	**Optimization Status**	**0.25**	**1.0**	**4.0**	**Total**
$B_0 = 240.23$	Optimal	1,453	1,453	1,453	4,359
	Feasible	0	0	0	0
$B_1 = 305.03$	Optimal	1,418	1,450	1,453	4,321
	Feasible	35	3	0	38
$B_2 = 371.03$	Optimal	1,396	1,443	1,453	4,292
	Feasible	57	10	0	67
$B_3 = 448.01$	Optimal	1,411	1,453	1,453	4,316
	Feasible	42	1	0	43
$B_4 = 552.48$	Optimal	1,413	1,453	1,453	4,319
	Feasible	40	0	0	40
$B_5 = 604.58$	Optimal	1,424	1,453	1,453	4,330
	Feasible	29	0	0	29
$B_6 = 694.64$	Optimal	1,447	1,453	1,453	4,346
	Feasible	7	0	0	7
$B_7 = 776.79$	Optimal	1,453	1,453	1,453	4,359
	Feasible	0	0	0	0
$B_8 = 853.52$	Optimal	1,453	1,453	1,453	4,359
	Feasible	0	0	0	0
$B_9 = 930.23$	Optimal	1,453	1,453	1,453	4,359
	Feasible	0	0	0	0
$B_{10} = 1006.95$	Optimal	1,453	1,453	1,453	4,359
	Feasible	0	0	0	0
$B_{11} = 1084.10$	Optimal	1,453	1,453	1,453	4,359
	Feasible	0	0	0	0
Total		17,436	17,436	17,436	52,308

7.6 The tradeoff mechanism

We will now discuss how our new objective function can capture the tradeoff between QoS_R and SL_R of a network. As discussed in Section 7.4, when $n = 0.25$, the network's QoS_R^b is consistently sacrificed to maintain a high level of resiliency in the network as long as the increasing budget levels are below a threshold budget level. This scenario also exists in the case of $n = 4.0$, but the deterioration in QoS_R^b is almost stagnant, which is achieved at the cost of lower edge failure resiliency (though it improves at a very slow rate with increasing

budget levels) in the network. Overall, this results in longer (or shorter) paths leading to poor (or better/stagnant) QoS_r^b of larger (or smaller/constant) values while making O-D pairs $r \in R$ more (or less) survivable against multiple-edge failures as reflected by larger (or smaller) values of SL_r in a network when $n = 0.25$ (or 4.0). In other words, both QoS_R^b and SL_R have opposite behaviours for smaller budgets (below a threshold budget level), even though, in terms of magnitude both QoS_R^b and SL_R are increasing with the budget. As a larger QoS_R^b implies a poor network connectivity, we will define the following new KPI:

$$\overline{QoS}_R^b = H_R^{min} + \Delta^b - QoS_R^b \tag{35}$$

Note that in any instance with a particular setting $(\Delta^p, \Delta^b, \Delta^s)$, the size of the largest backup path connecting an O-D pair $r \in R$ will be $H_R^{min} + \Delta^b$, which represents the poorest QoS offered by the network in that particular setting. This is reflected as a value of 0 in Equation 35, which is basically the opposite of QoS_R^b. This implies, that the higher (or lower) the value of $\overline{QoS}_R^b$, the better (or poorer) the backup QoS. We will use $\overline{QoS}_R^b$ and SL_R to show the tradeoff between the backup QoS and SL of a network.

We consider the dataset "D7x7-5-1_10_10_20-1" to show the QoS and SL tradeoff for the parameter settings presented in Table 2, when the survivability limit $\Delta^s = 3$ which implies that the designed network configuration may survive the failure of at most any 4 edges. In a total of 18 subplots, Figure 2 shows the variation in the $\overline{QoS}_R^b$, and SL_R along the y-axis with respect to different budget B levels represented on the x-axis. In Figure 2 each row represents a particular (Δ^p, Δ^b) setting, with subfigures showing a tradeoff plot for $n = 0.25, 1.0$ and 4.0 respectively as one move from left to right. For instance, Figures 2a to 2c are the tradeoff plots for $n = 0.25, 1.0$ and 4.0 respectively when $\Delta^p = \Delta^b = 0$.

The scale of SL_R is fixed in the range [0, 4], while the range of $\overline{QoS}_R$ depends on Δ^b and can vary from 0 to $6-8$ units when Δ^b is changed from $0-2$. The budget B has a range of [80, 550]. The first observation is that a clear tradeoff exists between $\overline{QoS}_R^b$ and SL_R for the network design irrespective of the value n, indicating that one requirement improves at the expense of the other. Secondly, as Δ^b increases, which implies longer connectivity paths are allowed, then there is an improvement of 40% in SL_R from 2 (Figure 2a) to 2.5 (Figures 2d, 2g, 2j, 2m and 2p) edges as when $n = 0.25$ and the budget is high with $B = 550$. Although, this effect is less pronounced when $n > 1.0$ and it even depends on the value of Δ^b, but this strengthens the observation that $n = 0.25$ prioritises the survivability aspect of the

network. However, Figures 2a, 2d, 2g, 2j and 2m shows that the backup QoS can deteriorate significantly by a minimum of 25% as $\overline{QoS}_R^b$ drops from $3-4$ units to close to $1-3$ units. This pattern is also consistent for all (Δ^p, Δ^b) combinations when $n = 0.25$. On the other hand, network configurations designed by the mechanism $n = 4.0$ have poor $SL_R < 1.5$, yet, the QoS offered by these networks is relatively better than the network of $n = 0.25$ as reflected through high values of $\overline{QoS}_R^b$, staying close to $5-7$ units when the budget is low, while never dropping below 1.5 units, which is evident from the Figures 2c, 2f, 2i, 2l, 2o and 2r. Also, observe that the deterioration in QoS is delayed as n increases. Note that $\overline{QoS}_R^b$ is close to 4 units for the budget $B \leq 170$ when $n = 0.25$, while $n = 4.0$ maintained the same level until B crosses 268 (a minor exception in the case of Figure 2c), which further indicates that $n = 4.0$ focuses on preserving the QoS over SL of a network, making it less resilient to multi-edge failures, while $n = 0.25$ does the opposite. Focusing on $n = 1.0$ highlights that the two metrics are well balanced as evident by their curves which are nearly stacked near each other in the Figures 2b, 2e, 2h, 2k, 2n and 2q. The mechanism $n = 1.0$ tried to construct a network that can survive the simultaneous failure of at most 2 edges, which is in between the SL_R of 1.5 edges and 2.5 edges for $n = 4.0$ and $n = 0.25$ respectively. Also, it controls the value of $\overline{QoS}_R^b$ around an average of $2-4$ units across all budget levels and all possible parameter settings. It can also be inferred that once SL_R and $\overline{QoS}_R$ crosses their paths, the QoS may also slowly improve along with SL, particularly when $n = 0.25$. In the case of $n \geq 1.0$, the $\overline{QoS}_R$ will remain almost stable, which comes at the cost of slowing down the growth of SL_R. In addition to this, it can be noted that the values of Δ^b can impact the network design cost for any n. At $\Delta^b = 0$, the minimum cost of construction was 125 (close to the average of 80 and 174), but it gradually shifts to 80 as Δ^b increases to 2 which allows relatively longer paths in comparison to the case of $\Delta^b = 0$. This implies that the cost of constructing longer paths is cheaper than establishing the O-D pair specific shortest paths, which contributes to the reduction of overall network cost. Finally, it is also important to highlight that while SL_R changed from 0 to at most 3.5 edges, the $\overline{QoS}_R$ never dropped to 0, which represents the worst QoS for a particular Δ^b setting. As discussed in Section 7.3, once a sufficient budget is allocated, both QoS and SL can be improved simultaneously. Hence, there is no incentive for the objective function (2) to yield lengthy connectivity paths when the budget is high.

Similar findings can be derived for different datasets and different values of $\Delta^s \in \{0, 1, 2, 3, 4\}$,

although the discussed effects of changing Δ^b or n will be more pronounced in the case of larger Δ^s as it will incorporate multiple edge failures. The given configuration of the network also plays a crucial role in deciding the level of tradeoff. The inherent topology may potentially prevent O-D pairs from surviving multi-edge failures even if the budget B or Δ^s or Δ^b are set to higher values. Overall, these observations demonstrate the power of our proposed objective functions, which can capture the tradeoff between backup QoS_R^b and SL_R, and can shift the tradeoff to design the network in favour of one requirement over another.

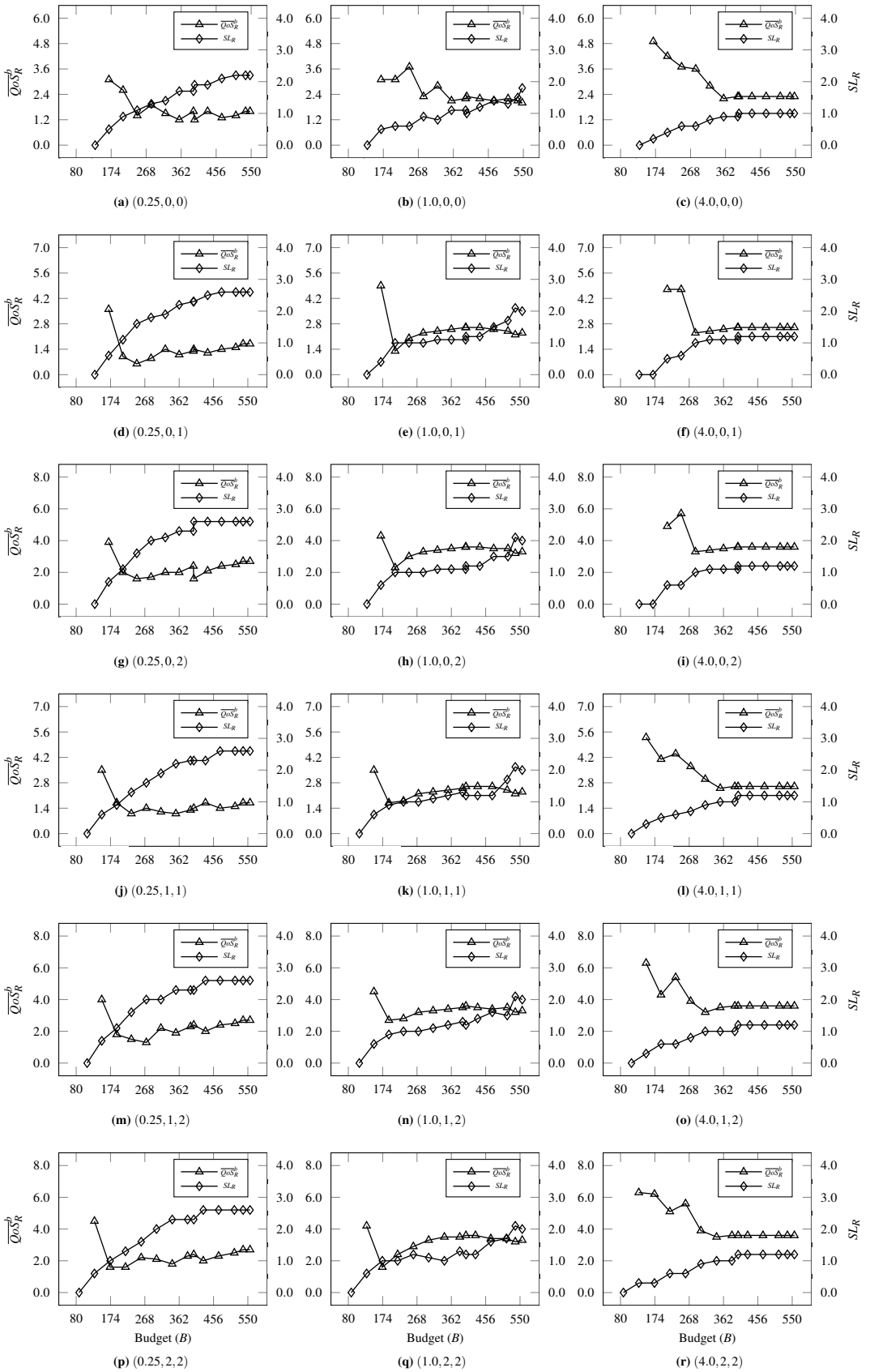


Figure 2: The tradeoff between the backup QoS and SL of the network "D7x7-5-1_10_10_20-1" for different combinations of (n, Δ^p, Δ^b) and $\Delta^s = 3$.

8 Conclusion

We have introduced, modeled and solved the network design problem with vulnerability and budget constraints (NDPVBC). This problem is a natural extension of the NDPVC by considering local-level survivability and quality-of-service for every O-D pair subject to a budget. The network is designed with the objective of either ensuring the existence of longer backup paths in the scenario of multi-edge failures to keep undisrupted yet poor connectivity or sacrificing the resilience to be built in the network, to enhance the connectivity among O-D pairs.

In this study, we proposed a new objective function that can establish the tradeoff between the Quality-of-Service (QoS) and Survivability Level (SL) of a network. A naive model was built that explicitly constructs paths for all the demands through the enumeration of multiple edge failure scenarios. We also developed a model that avoids the enumeration of edge sets and involves building length-bounded cuts (lbcuts), which destroys all the paths of certain length bound connecting a given O-D pair. By exploiting their structural properties, we generalised the applicability of a lbcut detected for a hop-limit over a range of smaller hop-limits. We have presented this strengthened version of the separation problem (Arslan et al. 2020) for our model and we have solved it through several heuristic algorithms while guaranteeing an exact separation via the solution of a mathematical model.

We have also derived three families of valid inequalities that have been extensively tested and demonstrated to improve the computational performances of the model by up to 22%, using 94,932 instances. We have identified the best performance settings for our algorithm and solved the problem for a large variety of input parameters. We also show that networks can be designed in favour of one objective of enhancing the robustness over tightening the O-D pair connectivity or vice-versa through our new objective function. Our findings show that for the graphs and settings we have considered, a degradation in the backup QoS by 8% can yield an improvement of 24% in the SL while improving it further by another 24%, requires the deterioration rate in QoS to be nearly doubled to 14%.

Bibliography

[1] Fatmah Almathkour, Youcef Magnouche, Ali Ridha Mahjoub, and Raouia Taktak. "Design of survivable networks with low connectivity requirements". In: *International Transactions in Operational Research* (2024).

[2] Okan Arslan, Ola Jabali, and Gilbert Laporte. "A flexible, natural formulation for the network design problem with vulnerability constraints". In: *INFORMS Journal on Computing* 32.1 (2020), pp. 120–134.

[3] Okan Arslan and Gilbert Laporte. "Network design with vulnerability constraints and probabilistic edge reliability". In: *Networks* 84 (2024), pp. 181–199.

[4] Anantaram Balakrishnan and Kemal Altinkemer. "Using a hop-constrained model to generate alternative communication network design". In: *ORSA Journal on Computing* 4.2 (1992), pp. 192–205.

[5] Benita M Beamon. "Supply chain design and analysis: Models and methods". In: *International Journal of Production Economics* 55.3 (1998), pp. 281–294.

[6] Fatiha Bendali, Ibrahima Diarrassouba, Ali Ridha Mahjoub, M Didi Biha, and Jean Mailfert. "A branch-and-cut algorithm for the k-edge connected subgraph problem". In: *Networks* 55.1 (2010), pp. 13–32.

[7] Fatiha Bendali, Ibrahima Diarrassouba, Ali Ridha Mahjoub, and Jean Mailfert. "The k edge-disjoint 3-hop-constrained paths polytope". In: *Discrete Optimization* 7.4 (2010), pp. 222–233.

[8] Bilge Bilgen and Irem Ozkarahan. "Strategic tactical and operational production-distribution models: a review". In: *International Journal of Technology Management* 28.2 (2004), pp. 151–171.

[9] Quentin Botton, Bernard Fortz, Luis Gouveia, and Michael Poss. "Benders decomposition for the hop-constrained survivable network design problem". In: *INFORMS Journal on Computing* 25.1 (2013), pp. 13–26.

[10] Ivan Contreras and Elena Fernández. "General network design: A unified view of combined location and network design problems". In: *European Journal of Operational Research* 219.3 (2012), pp. 680–697.

[11] Alysson M Costa, Jean-François Cordeau, and Gilbert Laporte. “Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints”. In: *Networks* 53.2 (2009), pp. 141–159.

[12] Geir Dahl and Luis Gouveia. “On the directed hop-constrained shortest path problem”. In: *Operations Research Letters* 32.1 (2004), pp. 15–22.

[13] Geir Dahl and Bjarne Johannessen. “The 2-path network problem”. In: *Networks* 43.3 (2004), pp. 190–199.

[14] Ibrahima Diarrassouba, Virginie Gabrel, Ali Ridha Mahjoub, Luıs Gouveia, and Pierre Pesneau. “Integer programming formulations for the k-edge-connected 3-hop-constrained network design problem”. In: *Networks* 67.2 (2016), pp. 148–169.

[15] Ibrahima Diarrassouba and Ali Ridha Mahjoub. “Polyhedral investigation of k edge-connected l-hop-constrained network design problem”. In: *hal-04051494* (2023).

[16] Bernard Fortz, Luis Gouveia, and Pedro Moura. “A comparison of node-based and arc-based hop-indexed formulations for the Steiner tree problem with hop constraints”. In: *Networks* 80.2 (2022), pp. 178–192.

[17] Bernard Fortz and Martine Labbé. “Polyhedral results for two-connected networks with bounded rings”. In: *Mathematical Programming* 93 (2002), pp. 27–54.

[18] Bernard Fortz, Martine Labbé, and Francesco Maffioli. “Solving the two-connected network with bounded meshes problem”. In: *Operations Research* 48.6 (2000), pp. 866–877.

[19] Bernard Fortz, Ali Ridha Mahjoub, S Thomas McCormick, and Pierre Pesneau. “Two-edge connected subgraphs with bounded rings: Polyhedral results and Branch-and-Cut”. In: *Mathematical Programming* 105 (2006), pp. 85–111.

[20] Petr A Golovach and Dimitrios M Thilikos. “Paths of bounded length and their cuts: Parameterized complexity and algorithms”. In: *Discrete Optimization* 8.1 (2011), pp. 72–86.

[21] Luis Gouveia. “Multicommodity flow models for spanning trees with hop constraints”. In: *European Journal of Operational Research* 95.1 (1996), pp. 178–190.

[22] Luis Gouveia. “Using variable redefinition for computing lower bounds for minimum spanning and Steiner trees with hop constraints”. In: *INFORMS Journal on Computing* 10.2 (1998), pp. 180–188.

[23] Luis Gouveia, Martim Joyce-Moniz, and Markus Leitner. “Branch-and-cut methods for the network design problem with vulnerability constraints”. In: *Computers & Operations Research* 91 (2018), pp. 190–208.

[24] Luis Gouveia and Markus Leitner. “Design of survivable networks with vulnerability constraints”. In: *European Journal of Operational Research* 258.1 (2017), pp. 89–103.

[25] Luis Gouveia, Pedro Patricio, and Amaro de Sousa. “Hop-constrained node survivable network design: An application to MPLS over WDM”. In: *Networks and Spatial Economics* 8 (2008), pp. 3–21.

[26] Martin Grötschel, Clyde L Monma, and Mechthild Stoer. “Design of survivable networks”. In: *Handbooks in Operations Research and Management Science* 7 (1995), pp. 617–672.

[27] Terry P Harrison. *The practice of supply chain management: where theory and application converge*. Springer Science & Business Media, 2005.

[28] David Huygens and Ali Ridha Mahjoub. “Integer programming formulations for the two 4-hop-constrained paths problem”. In: *Networks* 49.2 (2007), pp. 135–144.

[29] Hervé Kerivin and Ali Ridha Mahjoub. “Design of survivable networks: A survey”. In: *Networks* 46.1 (2005), pp. 1–21.

[30] Thorsten Koch and Alexander Martin. “Solving Steiner tree problems in graphs to optimality”. In: *Networks* 32.3 (1998), pp. 207–232.

[31] Ali Ridha Mahjoub and S Thomas McCormick. “Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation”. In: *Mathematical Programming* 124 (2010), pp. 271–284.

[32] Riccardo Mangiaracina, Guang Song, and Alessandro Perego. “Distribution network design: a literature review and a research agenda”. In: *International Journal of Physical Distribution & Logistics Management* 45.5 (2015), pp. 506–531.

[33] Mary J Meixell and Vidyaranya B Gargeya. “Global supply chain design: A literature review and critique”. In: *Transportation Research Part E: Logistics and Transportation Review* 41.6 (2005), pp. 531–550.

[34] Karl Menger. “Zur allgemeinen kurventheorie”. In: *Fundamenta Mathematicae* 10.1 (1927), pp. 96–115.

[35] Sebastian Orlowski and Roland Wessäly. *The Effect of Hop limits on Optimal cost in Survivable Network Design*. Ed. by S Raghavan and G Anandalingam. Vol. 33. Springer, Heidelberg, 2006, pp. 151–166.

[36] Mauricio GC Resende and Panos M Pardalos. *Handbook of Optimization in Telecommunications*. Springer Science & Business Media, 2008.

[37] Hanif D Sherali and Patrick J Driscoll. “Evolution and state-of-the-art in integer programming”. In: *Journal of Computational and Applied Mathematics* 124.1-2 (2000), pp. 319–340.

[38] Markus Sinnl and Ivana Ljubić. “A node-based layered graph approach for the Steiner tree problem with revenues, budget and hop-constraints”. In: *Mathematical Programming Computation* 8 (2016), pp. 461–490.

[39] Mechthild Stoer. *Design of survivable networks*. Vol. 1531. Berlin Heidelberg: Springer, 1992.

[40] Carlos J Vidal and Marc Goetschalckx. “Strategic production-distribution models: A critical review with emphasis on global supply chain models”. In: *European Journal of Operational Research* 98.1 (1997), pp. 1–18.

[41] Andrzej P Wierzbicki and Wojciech Burakowski. “A conceptual framework for multiple-criteria routing in QoS IP networks”. In: *International Transactions in Operational Research* 18.3 (2011), pp. 377–399.

[42] Richard T Wong. *A survey of network design problems*. Massachusetts Institute of Technology, Operations Research Center, 1976.

A Appendix for Nomenclature

A.1 Sets

$\mathbb{N} =$ Set of natural numbers

$V =$ Set of vertices

$E = \{[i,j] \mid$ Set of edges formed between two vertices i, j where $i, j \in V\}$

$\overline{G} = (N,E)$, An undirected graph

$A = \{(i,j),(j,i) \mid$ Set of two opposite arcs $[i,j] \in E\}$

$R = \{(o_r, d_r, H_r^{min}) \mid$ Set of O-D pairs in a network$\}$

$H_r^p = \{H_r^{min}, H_r^{min}+1, \ldots, H_r^{min}+\Delta^p \mid$ The primal hop-limit range set$\}$

$H_r^b = \{H_r^{min}, H_r^{min}+1, \ldots, H_r^{min}+\Delta^p \mid$ The backup hop-limit range set$\}$

$K = \{\kappa, \kappa+1, \ldots, \kappa+\Delta^s \mid$ The survivability set$\}$

$A_{rh}^p = \{(i,j) \in A \mid d_{o_r i} + d_{jd_r} + 1 \leq h; h \in H_r^p, r \in R\}$

$A_{rh}^b = \{(i,j) \in A \mid d_{o_r i} + d_{jd_r} + 1 \leq h; h \in H_r^b, r \in R\}$

$V_{rh}^p = \{i \mid$ Vertices induced by arcs in $A_{hr}^p\}$

$V_{rh}^b = \{i \mid$ Vertices induced by arcs in $A_{hr}^b\}$

$E_{rh}^p = \{[i,j] \mid$ Edges induced by arcs in $A_{hr}^p\}$

$E_{rh}^b = \{[i,j] \mid$ Edges induced by arcs in $A_{hr}^b\}$

$E_{rh}^b(i) = \{[i,j] \mid$ Set of edges $e \in E_{rh}^b$ that are incident to the vertex $i \in V_{rh}^b$ in the backup graph $E_{rh}^b\}$

$G_{rh}^p = (V_{hr}^p, A_{hr}^p)$, a directed primal graph where hop-limit is h for a O-D pair $r \in R$

$G_{rh}^b = (V_{hr}^b, A_{hr}^b)$, a directed backup graph where hop-limit is h for a O-D pair $r \in R$

$\overline{S_h} = \{\overline{S} \in A \mid$ Set of edges of a lbcut that destroys all paths of length at most h edges$\}$

$\Gamma_{rh}^p = \{\overline{S_h} \mid$ Set of lbcuts of length bound h in the primal graph $G_{rh}^p\}$

$\Gamma_{rh}^b = \{\overline{S_h} \mid$ Set of lbcuts of length bound h in the backup graph$G_{rh}^b\}$

$C_k =$ Set of all possible combinations of $k-1$ edges derived from E_{rh}^b

$A^b_{rh}(C_k) =$ $\{(i,j) \in A^b_{rh} : [i,j] \in C_k\}$

$\overline{V}^b_{rh} =$ $\{i \mid$ Vertices induced by arcs in $\overline{A}^b_{rh}\}$

$\overline{G}^b_{rh} =$ $(\overline{V}^b_{rh}, \overline{A}^b_{rh})$

$\overline{G}^b_{rh} =$ $(\overline{V}^b_{rh}, \overline{A}^b_{rh})$

$P^b_{rh} =$ $\{[i,j] \mid$ Set of edges that are required to construct the minimum cost-weighted shortest path connecting $r \in R$ in the graph G^b_{rh} induced by the hop-limit $h \in H^b_r\}$

$B_{set} =$ $\{B \mid$ The set of budget levels determined for a particular parameter setting and dataset$\}$

A.2 Parameters

$o_r =$ The origin vertex of a demand $r \in R$

$d_r =$ The destination vertex of a demand $r \in R$

$H^{min}_r =$ The minimum hop-distance to ensure connectivity of a demand $r \in R$

$H^{min}_R =$ $\max_{r \in R}\{H^{min}_r \mid$ The minimum global hop-distance required to connect all the O-D pairs R in a network$\}$

$\Delta^b =$ A non-negative constant that quantifies the extent of deterioration in the QoS offered by the backup paths for the demand set R

$\Delta^p =$ A non-negative constant that quantifies the extent of deterioration in the QoS offered by the primal paths for the demand set R

$\Delta^s =$ A non-negative constant that describes the extent of improvement in the SL of demand set R

$\Delta^B =$ Number of budget levels to consider for a network

$\kappa =$ The smallest non-negative integer-based constant which signifies that an O-D pair r can have a minimum SL of $\kappa - 1$ edges

$\varepsilon =$ A very small number

$t =$ Time limit assigned to the models or separation algorithms

$B =$ A non-negative Budget assigned for the construction of a network

$n =$ A non-negative decimal number that controls the trade-off between QoS and SL

$B^{min} =$ The minimum non-negative integer Budget required to construct the a network with primal paths only

$B_n^{max} =$ The minimum non-negative integer Budget that is required to construct the "Maximum Cost Network" for a particular n

$\beta_{rh}^b =$ Construction cost of graph G_{rh}^b

A.3 Variables

$\hat{x}_{ij} =$ Edge weights detected during integer/fractional separation for the edge $(i, j) \in E$

$\hat{z}_{rh} =$ Primal weight detected during integer/fractional separation for the O-D pair r in the graph G_{rh}^b

$\hat{y}_{rhk} =$ Backup weight detected during integer/fractional separation for the O-D pair r in the graph G_{rh}^b for any $k \in K$

$h =$ A strictly positive integer that signifies the number of hops in a path connecting an O-D pair

$h' = (h - H_r^{min} + 1)$

$k =$ A non-negative integer that signifies an O-D pair can survive the failure of any $k-1$ edges

$$x_e = \begin{cases} 1, & \text{if the edge } e \in E \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{rhk} = \begin{cases} 1, & \text{if edge connectivity of demand } r \in R \text{ is maintained using } h \text{ edges after} \\ & \text{the failure of any } k-1 \text{ edges,} \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{rh} = \begin{cases} 1, & \text{if edge connectivity of demand } r \in R \text{ is maintained using } h \text{ edges with-} \\ & \text{out any failure of edges,} \\ 0, & \text{otherwise.} \end{cases}$$

$$u_{ij}^{rh} = \begin{cases} 1, & \text{if arc } (i,j) \in A_{rh}^{p} \text{ for every } r \in R \text{ is on the path of hop-limit } h \text{ from } o_r \\ & \text{to } d_r, \\ 0, & \text{otherwise.} \end{cases}$$

$$v_{ij}^{rhC_k} = \begin{cases} 1, & \text{if arc } (i,j) \in \overline{A}_{rh}^{b}(C_k) \text{ for every } r \in R \text{ is on the path of hop-limit } h \text{ from} \\ & o_r \text{ to } d_r \text{ when any } k-1 \text{ edges in } C_k \text{ fails,} \\ 0, & \text{otherwise.} \end{cases}$$

$$f_{ij} = \begin{cases} 1, & \text{if arc } (i,j) \text{ belongs to the minimum weighted lbcut detected during int-} \\ & \text{eger/fractional separation,} \\ 0, & \text{otherwise.} \end{cases}$$

$SL_r = \sum_{h \in H_r^b} \sum_{k \in K} (k-1) y_{rhk}^{*}$, A non-negative integer that signifies the SL of an O-D pair in terms of number of edges it can survive

$SL_R = \frac{\sum_{r \in R} SL_r}{|R|}$, The average SL of a network

$QoS_r^b = \sum_{h \in H_r^b} \sum_{k \in K} (h) y_{rhk}^{*}$, A non-negative integer that signifies the QoS of the backup path(s) connecting an O-D pair in terms of the number of hops or edges

$QoS_R^b = \frac{QoS_R^b}{|R|}$, The average backup QoS of a network

$\overline{QoS}_R^b = H_R^{min} + \Delta^b - QoS_R^b$, The opposite of QoS_R^b

B Dataset

Table 13 provides the details of instances considered in this study, with the first two columns reporting the Group and their corresponding SubGroup sets. The next 4 columns show the sizes of vertex, edge and demand sets and the total number of instances associated with each (Group, SubGroup) combination. Finally, the minimum, average and maximum values of the parameter H_r^{min} have been provided in the rightmost columns. There are a total of 350 networks, but for the current study, only 50 networks were considered.

Table 13: Properties of Instance Sets.

						H^{min}		
Group	**SubGroup**	$\|V\|$	$\|E\|$	$\|R\|$	**Number**	**Minimum**	**Average**	**Maximum**
D	D-1	25	72	10	10	3	3.8	4
	D-2	49	156	10	10	4	5.2	6
E	E-1	50	122	10	5	6	7.6	9
	E-2	50	122	45	5	7	8.6	11
	E-3	50	245	10	5	4	4.6	6
R	RE-1	50	122	10	5	3	4.6	6
	RE-2	50	122	45	5	4	5.2	6
	RE-3	50	245	10	5	2	2.8	3

C Appendix for the alternative form of the separation algorithm

Note that Algorithm 2 is based on common separation such that it can yield up to two cuts, one for each primal and backup graph when $h \in H_r^p \cap H_r^b$. However, these cuts can be determined exclusively during primal and backup cut separation as described in the following Algorithm 3.

Algorithm 3 Algorithm for Independent Primal and Backup Cut Separation

Input: Demand Set R, the limits (Δ^p, Δ^b, Δ^k), κ, primal $\hat{z}_{rh}$ and backup $\hat{y}_{rhk}$ weights during a callback

Output: A set of violated constraints

for $i \in \{p, b\}$ **do**
  $H^i \leftarrow \{H_R^{min} + \Delta^i, H_R^{min} + \Delta^i - 1, \ldots, \min_{r \in R}\{H_r^{min}\}\}$
  for $h \in H^i$ **do**
    $G^i \leftarrow G^i_{rh}$
    for $r \in R$ **do**
      if $h \geq H_r^{min}$ **then**
        if $i = p$ **then**
          $R_{cut}^{i_{rh}} \leftarrow \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^p} \hat{z}_{rh}$ // Get the RHS value of 'p' graph Cut
        else if $i = b$ **then**
          $R_{cut}^{i_{rh}} \leftarrow \sum_{g=H_r^{min}}^{H_R^{min}+\Delta^b} \sum_{k=\kappa}^{\kappa+\Delta^k} (k)\hat{y}_{rhk}$ // 'b' graph Cut's RHS value
        if $R_{cut}^{i_{rh}} > 0$ **then** // Cuts with $R_{cut}^{i_{rh}} = 0$, never violates
          if $h \leq 3$ **then**
            /* Exact Separation */
            $CutSet^i$ += ε-lbcut3(G^i) // Section 5.1
          else
            /* Heuristic Separation */
            $CutSet^i$ += ε-minCut(G^i) // Section 5.2
            $tempCutSet^i$ += lbcutA(G^i, 0.016) // Section 5.3
            $tempCutSet^i$ += lbcutM(G^i, 0.1) // Section 5.4
            $tempCutSet^i \leftarrow$ CutEnhancer($tempCutSet^i$, 0.1) // Section 5.5
            $CutSet^i \leftarrow CutSet^i \cup tempCutSet^i$
    /* Exact Separation */
    if $CutSet^i = NULL$ **then**
      for $r \in R$ **do**
        if $h \geq H_r^{min}$ **and** $h > 3$ **then**
          if $R_{cut}^{i_{rh}} > 0$ **then** // Cuts with $R_{cut}^{i_{rh}} = 0$, never violates
            $CutSet^i$ += lbcutM(G^i, 10800) // Section 5.4
            $CutSet^i \leftarrow$ CutEnhancer($CutSet^i$, 0.1) // Section 5.5
return $CutSet^p \cup CutSet^b$