Les Cahiers du GERAD ISSN: 0711–2440

Network design with vulnerability constraints and probabilistic edge reliability

O. Arslan, G. Laporte

G–2023–50

November 2023

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2023
– Bibliothèque et Archives Canada, 2023

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2023
– Library and Archives Canada, 2023

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Network design with vulnerability constraints and probabilistic edge reliability

Okan Arslan [a]

Gilbert Laporte [a, b]

[a] *Department of Decision science, HEC Montréal & GERAD, Montréal, (Qc), Canada, H3T 2A7*

[b] *School of Management, University of Bath, Bath BA2 7AY, United Kingdom*

okan.arslan@hec.ca

November 2023
Les Cahiers du GERAD
G–2023–50

Abstract : The Network Design Problem with Vulnerability Constraints and Probabilistic Edge Reliability (NDPVC-PER) is an extension of the NDPVC obtained by additionally considering edge reliability. We consider the design of a telecommunication network in which every origin-destination pair is connected by a hop-constrained primal path, and by a hop-constrained backup path when certain edges in the network fail. The edge failures occur with respect to their reliability, and the network is designed by considering a minimum reliability level. Therefore, an hop-constrained backup path must be built by considering all simultaneous edge failures that have a certain probability of realization. While there exist models to solve the NDPVC without enumerating all edge subsets, edge reliability cannot be dealt with by applying the techniques applied to the NDPVC. Therefore, we develop models based on a new concept of *resilient length-bounded cuts*, and solve the NDPVC-PER without edge set enumerations. We perform extensive testing of the model to determine the best performing settings, and demonstrate the computational efficiency of the developed model. Our findings on these instances show that, in the dataset considered in this study, increasing the reliability level from 90% to 95% increases the average cost only by 12.4%, while increasing it from 95% to 99% level yields a cost increase of 93.9%.

Keywords : Network design, vulnerability, survivability, reliability, integer linear programming, length-bounded cut, resilient length-bounded cut

Acknowledgements: The authors gratefully acknowledge funding provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2015–06189 and 2022–04979.

1 Introduction

Telecommunication networks are designed to ensure the connectivity of a set of origin-destination (O-D) pairs. The links on these networks are prone to failures, and therefore the networks are designed by also ensuring that the connection is maintained when links some fail Wong (2021). The problem of ensuring connectivity in such networks has been handled from several standpoints. The Survivable Network Design Problem (SNDP) achieves connectivity by ensuring that, for every O-D pair, there exists k edge-disjoint paths or k node-disjoint paths, where k is the number of such paths, while ensuring that the setup cost is minimized. As a result, there exists a backup path when the primal path connecting an O-D pair fails. In order to ensure a certain level of quality in the network, the paths connecting the O-D pairs are also restricted to use a maximum number of edges, which is referred to as the *'hop bound'*. The Hop-Constrained Survivable Network Design Problem (kHSNDP) is similar to the SNDP and designs a network of minimum cost, but it additionally ensures that every O-D pair is connected using k edge-disjoint paths that contain at most H edges (Botton et al., 2013).

Note that designing a network with k edge-disjoint paths, implies that the network can maintain communication between an origin-destination (O-D) pair, even when *all* arcs on *all* of the primary $k-1$ paths between the same O-D pair *simultaneously* fail. However, this approach may be overly conservative in network design, even when $k = 2$, and can result in high setup costs. This fundamental assumption was later relaxed by Gouveia and Leitner (2017), who introduced the Network Design Problem with Vulnerability Constraints (NDPVC). In the NDPVC, each O-D pair is guaranteed to be connected by a hop-constrained primal path, and a hop-constrained backup path when $k \geq 2$ edges fail in the network (note that the parameter k is used for counting the failing *edges* in the NDPVC rather than the disjoint paths as in the SNDP or the kHSNDP). It is important to note that the NDPVC is less restrictive than the SNDP, as the backup path can utilize the working part of the primal path. Consequently, the NDPVC requires both a primal and backup hop bound to represent the quality of service between O-D pairs, along with an integer k that represents the network's survivability level.

In the current setting, after having solved the NDPVC, a telecommunication network manager may claim that the network is resilient to $(k-1)$-edge failures. However, the reliability of systems is typically expressed in terms of probabilities. In this paper, we associate probabilities to edge failures. Dahl and Stoer (1998) mention associating probabilities to edge failures, but argue that this problem would be computationally infeasible. In our paper, we develop models and algorithms that are feasible with today's modeling power and technology.

Indeed, the connectivity reliability of O-D pairs in a telecommunication network can be influenced by various factors, which include the quality, age, protection level and installation standards of the cables, as well as environmental conditions and maintenance frequencies. In particular, higher-quality cables are designed to better withstand environmental conditions, such as temperature variations, moisture, and mechanical stress compared with lower-quality cables. Furthermore, improper installation techniques, such as excessive bending, pulling with excessive force, or inadequate cable protection, can lead to cable damage and reduced reliability. For real-world implementations, we refer the reader to Alcoa Fujikura ltd. (2001) for different types of fiber-optical cables affecting reliability, to Roxtec Cable Seals (2023) for reliable cable seals for telecommunication networks, and to Eland Cables for the main causes of electrical cable failures. Similarly, Mashad Nemati (2017) present data-driven methods that utilize historical data in smart distribution grids for reliability evaluation. From this perspective, we associate probabilities with the edges, representing the likelihood of failures. These probabilities could be influenced by factors such as the quality of existing networks, historical data from the same region reflecting environmental conditions, or the quality of newly installed cables as indicated by manufacturers.

Note that, in the NDPVC, the number of failing edges is an input parameter. However, when the edge failure probabilities are considered, one or multiple edges may fail simultaneously, and the network needs to be protected against such failures. Therefore, the k parameter previously considered

is no longer a parameter but a variable of the problem. One can preprocess a network to enumerate all the edge sets that lead to a failure probability higher than the resilience level set by the managers, and we develop a model based on such an enumerative scheme, which is computationally very costly. In this paper, we also develop a mathematical model that does not require such an enumeration.

1.1 Literature review

The NDPVC problem has a strong connection with the k-edge-survivable network design problem (k-ESNDP), which involves finding a subset of edges that minimizes the total edge costs while ensuring the existence of k edge-disjoint paths for each O-D pair. Stoer (1992) studied the polyhedral properties of this problem for $k = 2$ and Grötschel et al. (1995) investigated it for general k values. For a comprehensive understanding of the k-HSNDP, please refer to Kerivin and Mahjoub (2005) and Bendali et al. (2010). Solutions to the k-ESNDP may result in relatively lengthy paths. To account for the quality of service in such networks, it is common practice to consider the number of edges on the path, often referred to as the 'hop bound' Balakrishnan and Altinkemer (1992). Hop bounds within spanning trees have been studied by Gouveia (1996) and Gouveia (1998). Recently, Fortz et al. (2022) compared the node-based and the arc-based hop-indexed formulations for the Steiner tree problem with hop constraints. The two-connected network with bounded meshes problem was studied by Fortz et al. (2000), in which the resulting network contains at least two vertex-disjoint paths between every pair of vertices, and each edge must belong to at least one cycle whose length is bounded by a given constant. The authors investigated the use of hop-constrained cycles connecting O-D pairs, which leads to two disjoint paths. Polyhedral results for two-connected networks with bounded rings are presented in Fortz and Labbé (2002) and Fortz et al. (2006). The k-HSNDP extends the k-ESNDP by introducing hop constraints for each O-D pair. Leveraging the layered network flow formulation developed by Gouveia (1998), Botton et al. (2013) introduced a Benders decomposition algorithm designed for the k-HSNDP, applicable to various values of k and specific length bounds. More recently, Diarrassouba and Mahjoub (2023) studied polyhedral properties of the k-ESNDP, and introduced several new classes of valid inequalities for the previously developed models.

The primary distinction between the k-HSNDP and the NDPVC lies in the way paths are constructed. In the k-HSNDP, the focus is on creating edge-disjoint paths, whereas in the NDPVC, the primary and backup paths do not need to be disjoint. The primary requirement is to ensure hop-constrained connectivity from sources to destinations in the event of the failure of any k edges in the network. Consequently, solutions of the k-HSNDP are more conservative than those of the NDPVC Gouveia and Leitner (2017). Branch-and-cut and Benders decomposition algorithms were developed for the NDPVC to accelerate its solution efficiency of the models Gouveia et al. (2018). These models depend on the explicit enumeration of edge sets, and on constructing paths. Observing that the network only needs to ensure the existence of such paths, but does not necessarily need to explicitly model them, Arslan et al. (2020) presented models based on the idea of length-bounded cuts. These models do not rely on edge set enumeration, but implicitly guarantee the existence of paths. They proved to be more efficient than the former models on a wide range of problem instances.

1.2 Scientific contributions and organization of the paper

We extend the network design problem with vulnerability constraints by considering failure probabilities associated with edges of the network. We introduce the Network Design Problem with Vulnerability Constraints and Edge Reliability (NDPVC-PER) and present mathematical models for solving it. The models in the literature developed for solving similar problems without any enumeration cannot be applied to the NDPVC-PER, and we therefore propose a novel idea based on a new cut type, referred to as *Resilient Length-bounded Cut*. This idea enables us to formulate a model that does not require the edge set enumeration. We conduct computational experiments to test the effectiveness of these models, and demonstrate that considering probabilities is important to measure the impacts on the design costs.

In Section 2, we provide the preliminaries and formally introduce the problem. In Section 3, we present an arc-flow model for solving this problem that is built on the idea of enumerating edge subsets. In Section 4, we present a model based on length bounded cuts inspired from the literature. In Section 5, we develop a model based on resilient length-bounded cuts, which does not require edge set enumeration. Section 6 describes the separation problem, and methods for solving this problem including several heuristics. Section 7 presents a computational study and results. We conclude the paper in Section 8.

2 Preliminaries and problem definition

We now present the preliminaries in Section 2.1, model the resilience in Section 2.2 and define the problem in Section 2.3.

2.1 Preliminaries

Let $G = (N, E)$ be an undirected graph, where N is the set of nodes and E is the set of edges $e = [i, j]$ with $i, j \in N$ and $i < j$. Parameter $c_e = c_{ij}$ is the cost of edge $e = [i, j] \in E$ with $0 \leq c_e \leq 1$, and $p_e = p_{ij}$ is the edge reliability. Consider the corresponding directed graph $\bar{G} = (N, A)$ where the arc set A contains arcs (i, j) and (j, i) for each edge $[i, j] \in E$. For arc $(i, j) \in A$, let d_{ij} be the minimum number of arcs in $\bar{G}$ from node i to node $j \in N$. A demand r is defined as a tuple $\langle o^r, d^r, H_p^r, H_b^r \rangle$, where o^r is the origin, d^r is the destination, and H_p^r and H_b^r are the primary and backup hop limits, respectively. Set R represents the set of all demands. Let $A_r^p = \{(i, j) \in A : d_{s_r i} + d_{j t_r} + 1 \leq H_r^p\}$ be the primal graph and $A_r^b = \{(i, j) \in A : d_{s_r i} + d_{j t_r} + 1 \leq H_r^b\}$ be the backup arc sets corresponding to demand $r \in R$. Let N_r^p and N_r^b be the node sets induced by the respective arc sets, A_r^p and A_r^b. Similarly, let E_r^p and E_r^b be the edge sets induced by the respective arc sets, A_r^p and A_r^b. We refer to the graphs $G_r^p = (N_r^p, A_r^p)$ and $G_r^b = (N_r^b, A_r^b)$ as the primary and backup graphs, respectively, of demand $r \in R$. For $r \in R$, and an edge set $\mathcal{C} \subset E_b^r$, let $G_r^b(\mathcal{C})$ be the graph induced by the arcs $A_r^b(\mathcal{C}) := \{(i, j) \in A_r^b : [i, j] \in \mathcal{C}\}$. When $\mathcal{C}$ contains only one edge $[i, j]$, we write $A_r^b([i, j])$ instead of $A_r^b(\{[i, j]\})$. In other words, $A_b^r([i, j])$ includes arcs (i, j) and (j, i), if they exist in A_b^r. The set $A_p^r([i, j])$ is similarly defined for the primal path. Let $\overline{A}_r^b(\mathcal{C}) := A_r^b \setminus A_r^b(\mathcal{C})$, $\overline{N}_r^b(\mathcal{C})$ be the set of nodes induced by the arc set $\overline{A}_r^b(\mathcal{C})$, and $\overline{G}_r^b(\mathcal{C}) := (\overline{N}_r^b(\mathcal{C}), \overline{A}_r^b(\mathcal{C}))$. The set $\overline{G}_r^p(\mathcal{C})$ is similarly defined.

2.2 Resilience definition

The Oxford English Dictionary defines reliability as *"the quality or fact of being able to recover quickly or easily ..."*. In the scope of this paper, for a network G and a *resilience level* $0 < P < 1$, the following three statements are equivalent:

- A network is $(100 \times P)$-resilient.
- A network is resilient to edge failures happening with at least probability P.
- All *simultaneous* edge failures that have a probability at least P can be effectively handled (i.e., there is a backup plan for such occurrences).

In other words, the network manager is concerned with all events with at least probability P. If the probability of failure is less than P, the network does not guarantee the existence of a backup plan for these cases (for example, when several edges fail simultaneously).

Let p_e be the probability of edge $e \in E$ being operational. The probability of failure is then given by $1 - p_e$. For a subset of the edge set, $\mathcal{C} \subset E$, the probability of simultaneous failures is

$$P(\mathcal{C}) := \prod_{e \in \mathcal{C}} (1 - p_e). \tag{1}$$

2.3 Problem definition

Definition 1. Given an undirected graph $G = (N, E)$, a demand set R, and a resilience level P, the Network Design Problem with Vulnerability Constraints and Probabilistic Edge Reliability (NDPVC-PER) is defined as finding a subset of edges with the minimum cost that ensures the connectivity of all demands from their origins to their destinations, respecting their primary hop limit, and ensuring that there exists a backup path between the same O-D pair when all edges in the edge set $\mathcal{C}$ are removed for every $\mathcal{C} \subset E : P(\mathcal{C}) \geq P$.

3 Model based on arc flows

We now present a mathematical model that explicitly builds the paths for every edge failure case in the network by enumerating all subsets of the edge set. Let x_e be a binary variable that equals 1 if edge $e \in E$ is selected, and 0 otherwise. We define binary variables y_{ij}^r equal to 1 if and only if arc $(i, j) \in A_p^r$ for $r \in R$ is on the path from o^r to d^r. Similarly, for every demand $r \in R$ and every subset of edges $\mathcal{C} \subset E_b^r$ such that $p(C) \geq P$, we define binary variables $z_{ij}^{r\mathcal{C}}$ equal 1 if and only if arc $(i, j) \in \overline{A}_r^b(\mathcal{C})$ is on the path from o^r to d^r when edges in $\mathcal{C}$ fail. In line with the resilience definition in Section 2.2, for $r \in R$, the edges that should be taken into account when designing the network are $\mathcal{C} \subset E : p(\mathcal{C}) \geq P$. However, the edges in $E \setminus E_b^r$ will not be used by either the primal or the backup paths. Therefore, we only consider the edge set E_b^r when determining the failing edges. Note that considering only the edges in E_p^r would not be correct because the backup path may still use the edges in $E_b^r \setminus E_p^r$ to ensure the backup connectivity and should use the edges that have not failed. In the following model, all such sets need to be enumerated before building the model. The model M1 is then formulated as follows.

$$\text{(M1) minimize} \sum_{e \in E} c_e x_e \tag{2}$$

subject to

$$\sum_{j:(i,j)\in A_p^r} y_{ij}^r - \sum_{j:(j,i)\in A_p^r} y_{ji}^r = \begin{cases} 1 & \text{if } i = o^r \\ -1 & \text{if } i = d^r \\ 0 & \text{otherwise} \end{cases} \qquad i \in N_p^r, r \in R \tag{3}$$

$$\sum_{j:(i,j)\in A_b^r \setminus A[\mathcal{C}]} z_{ij}^{r\mathcal{C}} - \sum_{j:(j,i)\in A_b^r \setminus A[\mathcal{C}]} z_{ji}^{r\mathcal{C}} = \begin{cases} 1 & \text{if } i = o^r \\ -1 & \text{if } i = d^r \\ 0 & \text{otherwise} \end{cases} \qquad i \in N_b^r, \mathcal{C} \subset E_b^r : p(\mathcal{C}) \geq P, r \in R \tag{4}$$

$$\sum_{(i,j)\in A_p^r} y_{ij}^r \leq H_r^p \qquad r \in R \tag{5}$$

$$\sum_{(i,j)\in \overline{A}_r^b(\mathcal{C})} z_{ij}^{r\mathcal{C}} \leq H_r^b \qquad \mathcal{C} \subset E_b^r : p(\mathcal{C}) \geq P, r \in R \tag{6}$$

$$\sum_{(k,l)\in A_p^r([i,j])} y_{kl}^r \leq x_e \qquad e = [i, j] \in E_p^r, r \in R \tag{7}$$

$$\sum_{(k,l)\in A_b^r([i,j])} z_{kl}^{r\mathcal{C}} \leq x_e \qquad \mathcal{C} \subset E_p^r \setminus \{e\} : p(\mathcal{C}) \geq P, \; e = [i, j] \in E_b^r, r \in R \tag{8}$$

$$x_e \in \{0, 1\} \qquad e \in E \tag{9}$$

$$y_{ij}^r \in \{0, 1\} \qquad (i, j) \in A_p^r, r \in R \tag{10}$$

$$z_{ij}^{r\mathcal{C}} \in \{0, 1\} \qquad (i, j) \in A_b^r \setminus A[\mathcal{C}], \; \mathcal{C} \subset E_p^r : p(\mathcal{C}) \geq P, r \in R. \tag{11}$$

Objective function (2) minimizes the total network design cost. Constraints (3) are node balance equations that yield a primary path for every demand $r \in R$. Similarly, constraints (4) are node bal-

ance equations for building backup paths for every subset of edges $\mathcal{C} \subset E_p^r$ such that $p(C) \geq P$. Constraints (5) ensure that the constructed primal and back paths respect the hop bounds. Constraints (6) and (7) ensure that an arc can only be used if the corresponding edge is selected. Constraints (9)–(11) define the domains of the variables.

Observe that the enumerative nature of Model M1 is restrictive. In the following section, we develop alternative models based on the notion of length-bounded cuts.

4 Model based on Length-bounded cuts

We now develop models based on length-bounded cuts.

Definition 2. Given a directed graph $\bar{G} = (N, A)$, a demand $r \in R$, a positive integer H as a length bound on the paths and an O-D pair (o^r, d^r), a set of arcs $\bar{S} \subset A$ is called an lbcut, if the removal of the arcs in $\bar{S}$ disconnects all paths of length at most H from o^r to d^r in G.

Note that every cut in graph $\bar{G}$ is also an lbcut. There may be additional lbcuts in $\bar{G}$ that disconnect all path of certain length. Let $S \subset E$ be the set of *edges* induced by the arcs in $\bar{S} \subset A$, which is an lbcut in the corresponding undirected graph G. Let Γ_r^p be the set of all such edge sets S corresponding to the lbcuts $\bar{S}$ of length bound H_r^p in G_r^p. Similarly Γ_r^b is defined as the set of all edge sets $S \subset E$ corresponding to the lbcuts $\bar{S} \subset A$ of length bound H_r^p in G_r^b. We also define $\Gamma_r^b(\mathcal{C})$ as the set of all edge sets $S \subset E$ corresponding to the lbcuts $\bar{S} \subset \overline{A}_r^b(\mathcal{C})$ of length bound H_r^p in $\overline{G}_r^b(\mathcal{C})$.

We now develop a natural formulation for the NDPVC-PER, which we refer to as M2. In our formulation, we ensure the existence of a hop-constrained path using lbcuts Dahl et al. (2006); Arslan et al. (2020, 2019). For the sake of completeness, we repeat Proposition 1 in Arslan et al. (2020) here.

Proposition 1. ***[Proposition 1 in Arslan et al. (2020)]*** *For a given graph G, an O-D pair (s, t) and a hop bound H, there exists a path of length at most H from s to t if and only if every lbcut contains at least one edge of the path.*

For the proof, we refer the reader to Arslan et al. (2020). M2 ensures the existence of paths by implicitly through lbcuts.

$$\text{(M2)} \quad \text{minimize} \quad \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in S} x_e \geq 1 \qquad S \in \Gamma_r^p, r \in R \tag{12}$$

$$\sum_{e \in S} x_e \geq 1 \qquad S \in \Gamma_r^b(\mathcal{C}), \mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P, r \in R \tag{13}$$

$$x_e \in \{0, 1\} \qquad e \in E.$$

The objective function minimizes the total cost. Constraints (12) ensure that, for $r \in R$ there exists a primal path of length at most H_r^p between o^r and d^r. Constraints (13) ensure the existence of a path of length at most H_b^r between the same O-D pair when edges in the edge set $\mathcal{C}$ fails for every $\mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P$. The last set of constraints define the binary requirements.

There are exponentially many constraints corresponding to every lbcut for every demand. Hence we develop a branch-and-cut algorithm for solving the problem and the corresponding separation problem, and its results are presented in Section 6.

Observe that even though the enumeration of lbcuts is avoided by solving the model using a branch-and-cut algorithm, the failing edges $\mathcal{C}$ still need to be enumerated a priori in Model M2. In other words, Constraints (13) are added to the model for every lbcut in the graph that corresponds to removal of

edges in the edge set $\mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P$ and this set needs to be enumerated. When solving the NDPVC with k failing edges (without considering the edge reliability), the cardinality of the failing edges is always taken as $k-1$. Therefore, the constraints $\sum_{e\in S} x_e \geq k$ for every $S \subset \Gamma^p$ ensure the existence of hop-constrained backup paths because when $k-1$ edges fail, the remaining one edge still ensures connectivity. Nevertheless, this is no longer the case for the problem at hand when we consider edge reliability because the cardinality of the edge set $\mathcal{C}$ is not necessarily fixed. Therefore, the model developed by Arslan et al. (2020) for solving the NDPVC with k failing edges cannot solve the NDPVC-PER. Hence, even when M2 is solved by means of a branch-and-cut algorithm, the separation problem still needs to be solved for every edge set $\mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P$. In the following section, we present a novel idea that also avoids the enumeration of these sets.

5 Model based on resilient length-bounded cuts

We now present a natural model that does not require enumerating the failing edges $\mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P$. We first define a *resilient length-bounded cut (r-lbcut)*.

Definition 3. Given a directed graph $\bar{G} = (N, A)$, edge reliability p_{ij} for all $(i,j) \in A$, a resilience level P, a demand $r \in R$, a positive integer H as a length bound on the paths and an O-D pair (o^r, d^r), a set of arcs $\bar{T} \subset A$ is called a *Resilient Length-bounded Cut (r-lbcut)*, if the removal of the arcs in $\bar{T} \cup \mathcal{C}$ disconnects all paths of length at most H from o^r to d^r in G for a set $\mathcal{C} \subset A : p(\mathcal{C}) \geq P$.

Observe that, in the above definition, $\bar{T} \cup \mathcal{C}$ is an lbcut. The r-lbcut T is simply those edges that ensure connectivity of the O-D pair after the edges in $\mathcal{C}$ fails.

Remark 1. Given an lbcut S, set $T := S \setminus \mathcal{C}$ is an r-lbcut for every $\mathcal{C} \subset S : p(\mathcal{C}) \geq P$.

For $r \in R$ and a resilience level P, let $\Omega_r^b(P)$ be the set of all r-lbcuts. By definition, $\Omega_r^b(P) = \Gamma_r^b(\mathcal{C}), \mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P$. We can then model the NDPVC-PER using the idea of r-lbcuts, which does not require the enumeration of failing edge sets.

$$
\begin{aligned}
\text{(M3)} \quad \text{minimize} \quad & \sum_{e\in E} c_e x_e \\
\text{subject to} \quad & \\
& (12) \\
& \sum_{e\in S} x_e \geq 1 && S \in \Omega_r^b, r \in R && (14) \\
& x_e \in \{0,1\} && e \in E.
\end{aligned}
$$

The difference of Model M3 with respect to M2 lies in constraints (14), which ensure the existence of a hop-bounded backup path when edges with a certain simultaneous failure probability are removed from the graph. The key difference lies in the way the constraints are separated. For (13), the separation problem consists of finding an lbcut, whereas for (14), the separation problem is to identify an r-lbcut. In the following section, we elaborate on the solution methods of the separation problem.

6 The separation problem and its solution

Given a solution $x^* \in \mathbb{R}^{|E|}$ of NDPVC-PER, the separation problem for Constraints (14) is to identify an r-lbcut of weight 1 in the subgraph induced by x^*, or to conclude that none exists. For Constraints (12) and (13), the separation problem reduces to finding an lbcut of weight 1 on the same subgraph, or to prove that none exists. For $r \in \mathcal{R}$, we separate

- Constraints (14) by an r-lbcut $S \in \Omega_r^b$ with $x^*(S) < 1$, and
- Constraints (12) by an lbcut $S \in \Gamma_r^p$ with $x^*(S) < 1$.

Additionally, for $r \in \mathcal{R}$ and $\mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P$, we separate

- Constraints (13) by an lbcut $S \in \Gamma_r^b(\mathcal{C})$ with $x^*(S) < 1$.

Since the problem of finding an r-lbcut is a generalization of the problem of finding an lbcut, we focus on the former. We next present a linearization for modeling the multiplication of probabilities, and we then present a mathematical model for finding the minimum weight r-lbcut.

Proposition 2. *The following two statements are equivalent:*

$$P(\mathcal{C}) \geq P \tag{15}$$

$$\sum_{[i,j]\in\mathcal{C}} \log\frac{1}{(1-p_{ij})} \leq \log\frac{1}{(1-P)}. \tag{16}$$

Proof. By considering the logarithms and elementary operations, the proof is straightforward (see Exercise 4.39 in Ahuja et al. (1993)). □

Using this linear reformulation, we next present a mathematical model for solving the separation problem.

6.1 A mathematical model for the separation problem

Consider a directed graph $G = (\hat{N}, \hat{A})$, an O-D pair (o, d), edge weights $\hat{x}_{ij}$ for arc $(i, j) \in \hat{A}$ and a hop bound H. Let $u_{ij} = 1$ if and only if arc $(i, j) \in \hat{A}$ is in the minimum r-lbcut, and $t_{ij} = 1$ if and only if arc $(i, j) \in \hat{A}$ is in the failed arcs set $\mathcal{C}$. Additionally, let π_i be the shortest path from the destination node d to node $i \in \hat{N}$, if this path does not contain any arc on the r-lbcut S or on the set of failing edges $\mathcal{C}$, and a large number otherwise. These variables could be considered as node potentials. The mathematical model for solving the minimum r-lbcut problem is

$$\text{(r-lbcutM)} \quad \text{minimize} \quad \sum_{(i,j)\in\hat{A}} \hat{x}_{ij}u_{ij} \tag{17}$$

subject to

$$\pi_d = 0 \tag{18}$$

$$\pi_i \leq \pi_j + 1 + M(u_{ij} + t_{ij}) \qquad (i,j) \in \hat{A} \tag{19}$$

$$\pi_o \geq H + \varepsilon \tag{20}$$

$$\sum_{(i,j)\in\hat{A}} \log(\frac{1}{1-p_{ij}})t_{ij} \leq \log(\frac{1}{1-P}) \tag{21}$$

$$\pi_i \geq 0 \qquad i \in \hat{N} \tag{22}$$

$$u_{ij}, t_{ij} \in \{0, 1\} \qquad (i,j) \in A. \tag{23}$$

The objective function (17) minimizes the weight of the r-lbcut. Constraints (18) and (19) ensure that the node potentials are correctly calculated by the π variables. Constraint (20) ensures that the destination cannot be reached from the origin using H hops. The parameter ε is a very small number. Constraint (21) ensures that the failing edges that form the set $\mathcal{C}$ yields $\mathcal{C} \subset A_r^b : p(\mathcal{C}) \geq P$ (due to Proposition 2). Finally, constraints (22) and (23) define the domains of the variables.

For graph G and demand $r \in R$, we refer to preceding mathematical model as r-lbcutM(G) when the minimum weight r-lbcut problem is solved on graph G. Observe that since we are searching for a minimum weight r-lbcut, the arcs with zero weights do not contribute to the objective function and the optimal solution of the mathematical model may potentially include many such arcs. To avoid this, a straightforward idea is to change the weight of such variables with zero coefficients to a small value ε so that the cut size is eventually finalized. Note that this may spoil the optimality of the cut,

particularly when the weights are also too small. Therefore, we use this idea of adding an ε coefficient to the variables with zero weights only when using the mathematical model as a heuristic. We also impose a time limit when using this model as a heuristic, and we refer to it as ε-r-lbcutM-h(G).

6.2 An ε-minimum cut heuristic

All minimum weight cuts are also length bounded cuts, and a length bounded cut is a superset of a resilient length bounded cut. Therefore, finding a minimum weight cut may help identify any violations of Constraints (12), (13) or (14). Furthermore, it can be used for both fractional and integer separation. As in the ε-r-lbcutM-h(G) heuristic, assigning an ε value to those arc with zero weights yields cuts with minimal cardinality, and therefore the cut may be stronger. Similar strengthening ideas were also implemented by Koch and Martin (1998) and Arslan et al. (2020).

6.3 An ε-resilient minimum cut heuristic

Observe that the minimum cut generated by the ε-minimum cut heuristic can further be strengthened when resilience is considered. Let S be a cut generated by the ε-minimum cut heuristic. By definition of a resilient length-bounded cut, a subset $\mathcal{C} \subset S$ may fail, and the edges in $S \setminus \mathcal{C}$ should still ensure the connectivity. The following model minimizes the weight of the r-lbcut by maximizing the weight of the edges that are removed from the cut as failing edges. Let t_{ij} be equal 1 if and only if edge (i, j) in a given cut S is considered as a failing edge. Note that those failing edges are not part of the resulting r-lbcut. Therefore, given a cut S, we present knapsack heuristic model as

$$\text{(Knapsack-h)} \quad \text{maximize} \quad \sum_{(i,j)\in S} \hat{x}_{ij} t_{ij} \tag{24}$$

$$\text{subject to}$$

$$\sum_{(i,j)\in \hat{A}} \log(\frac{1}{1-p_{ij}}) t_{ij} \leq \log(\frac{1}{1-P}) \tag{25}$$

$$t_{ij} \in \{0, 1\} \qquad (i, j) \in S. \tag{26}$$

The objective function maximizes the weight of the set of edges that are "failing" edges. The remaining edges in the cut form the r-lbcut. Constraint (25) ensures that the probability of simultaneous failure of the edges is at least P (due to Proposition 2). Constraints (26) state the binary requirements on the variables.

6.4 Length-bounded cuts with hop limits of at most three

A minimum weight length-bounded cut can be found in polynomial time if the bound is at most three units Mahjoub and McCormick (2010). The authors of this paper build a linear time network transformation, and solving a minimum weight cut problem on the transformed graph corresponds to a minimum weight length-bounded cut in the original graph. We again assign an ε value for zero-valued arcs, which helps minimalize the size of the minimum cut. Observe that the generated cut is an lbcut, and is an exact separation for Constraints (12) when the hop limit $H^p \leq 3$. On the other hand, it is a heuristic for the separation problem corresponding to Constraints (13) and (14). In the latter case when the algorithm is used as a heuristic, the Knapsack-h heuristic can also be applied to the generated cut in order to construct a resilient length-bounded cut. We refer to this technique as *ε-lbcut3(G)* when solved on an input graph G.

7 Computational study

We now present the data, the implementation details, the experimental design, and the results.

7.1 Data

Table 1 shows the NDPVC instances in Gouveia and Leitner (2017), which include a total of 350 networks with grid and random structures. In this study, we use the same set of instances. Additional details on the generation of these networks are discussed in Gouveia and Leitner (2017). The first five columns in Table 1 show the name, the node, edge and demand counts and the number of instances in each set, respectively. The parameter H_{min} represents the minimum hop limit among the demands in the particular set of instances. Columns 6−8 report the minimum, average and maximum value of the H_{min} of these instances.

Table 1: Properties of the instance sets.

					H_{min}		
Set	$\|N\|$	$\|E\|$	$\|\mathcal{R}\|$	Number	Minimum	Average	Maximum
C-1	100	342	5	20	3	4.7	7
C-2	100	342	10	20	4	5.4	7
C-3	400	1482	5	20	3	4.7	7
C-4	400	1482	10	20	4	5.05	7
C-5	400	1482	20	20	4	5.75	7
C-6	900	3422	5	20	3	4.65	7
C-7	900	3422	10	20	4	5.4	7
C-8	900	3422	20	20	4	5.6	7
C-9	900	3422	30	20	4	5.9	7
D-1	25	72	10	10	3	3.8	4
D-2	49	156	10	10	4	5.2	6
D-3	100	342	10	10	7	8.2	9
D-4	100	342	45	10	6	8.1	9
D-5	400	1482	10	10	12	14.6	18
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
E-4	50	245	45	5	4	4.8	6
E-5	75	277	10	5	5	5.6	6
E-6	75	277	45	5	6	8.6	12
E-7	75	555	10	5	3	4.4	6
E-8	75	555	45	5	4	4.6	5
E-9	100	495	10	5	5	5.2	6
E-10	100	495	45	5	6	7.2	9
E-11	100	990	10	5	3	4.0	5
E-12	100	990	45	5	3	4.8	6
R-1	50	122	10	5	3	4.6	6
R-2	50	122	45	5	4	5.2	6
R-3	50	245	10	5	2	2.8	3
R-4	50	245	45	5	3	3.0	3
R-5	75	277	10	5	3	3.8	4
R-6	75	277	45	5	4	4.2	5
R-7	75	555	10	5	2	2.6	3
R-8	75	555	45	5	2	2.8	3
R-9	100	495	10	5	3	3.6	5
R-10	100	495	45	5	4	4.0	4
R-11	100	990	10	5	2	2.0	2
R-12	100	990	45	5	3	3.0	3

In the experimental design, we used 10 selected networks to test the performance of the models and to determine the best parameter settings. These networks are referred to as the "10 networks" in the remainder of this paper, and they come of C-1 and C-2 datasets. All 10 networks have 100 nodes, 342 edges and 5 demands. They are the first five networks in *C10x10-5-1_10_10_50-2_5* and *C10x10-5-1_10_10_50-2_7* networks in the dataset.

In addition to the parameters provided in the dataset, we need reliability for all the edges in the networks. In this study, we assign the reliability of edge $(i, j) \in E$ as follows:

$$p_{ij} = p_{base} + (1 - p_{base} - \varepsilon) \times f_{ij},$$

where $0 < p_{base} < 1$ is the minimum probability value, ε is a very small value, which we take as 10^{-7}, and f_{ij} is a parameter between 0 and 1 that controls the distribution of the edge probability between p_{base} and $1 - 10^{-7}$. This ensures that $0 < p_{ij} < 1$. In this study, we generate f_{ij} using two methods. We refer to the first one as *cost-dependent*, in which $f_{ij} = c_{ij}/c_{max}$ where c_{max} is the maximum cost for the particular instance. This ensures that the instances can easily be replicated, and also the underlying idea is that the cost would increase for more reliable arcs, and hence the function. In the second method, we randomly generate f_{ij} using Java with seed number 1000. This also ensures that p_{ij} is distributed uniformly between p_{base} and $1-10^{-7}$. The instances can again be replicated, and this instance generation yields a more random edge reliability to test the efficiency of the methods. We use the "*Random generator = new Random(1000)*" command to generate f_{ij} values. In the experimental design, we refer to the reliability distribution as F, and $F = c$ implies that the reliability is a function of *c*ost, and $F = r$ implies that it is randomly distributed.

Table 2 presents the maximum number of edge subsets for which a backup path should be built for varying values of P and p_{base} among the 10 networks. Observe that the number reaches to more than 30.5 million paths when $P = 0.99$ and $p_{base} = 0.65$. Clearly, enumerating these subsets and building a model is not practical (or even infeasible) with the existing computational power. Table 3 reports the minimum, the average and the maximum numbers of edges in these failing edge sets for different values of P and p_{base} among the 10 networks. Note that the average number of edges in each set varies between 1 and 4 when $p_{base} = 0.65$ and $P = 0.99$, and the average set cardinality is 3.35. These values show the difficulty of enumeration and justify the development of methods that do not require explicit enumeration.

Table 2: Maximum number of edge subsets for which a backup path should be built for varying values of P and p_{base} among 10 selected networks.

	Reliability level (P)									
p_{base}	0.90	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.65	18,565	31,411	40,040	47,416	54,652	63,065	74,107	1,150,847	2,745,684	30,504,945
0.70	843	876	6,875	25,566	38,878	48,923	58,966	71,065	777,273	3,504,831
0.75	811	833	861	881	916	21,803	41,083	55,388	72,685	1,596,046
0.80	734	768	811	833	876	893	951	30,113	53,735	81,418
0.85	628	681	711	768	811	843	881	951	6,973	58,996
0.90	0	275	556	613	681	734	811	876	951	1,006
0.95	0	0	0	0	0	0	556	681	811	951
Maximum	18,565	31,411	40,040	47,416	54,652	63,065	74,107	1,150,847	2,745,684	30,504,945

7.2 Implementation details

We implemented our algorithms using Java, and all the experiments were conducted on the Cedar cluster of Digital Research Alliance of Canada using single thread and 20GB of RAM on a Linux environment. We used CPLEX 22.1.0.0 and its built-in constraint callback functions. The time limit for each of the experiments was set to two hours.

The pseudo-code of the separation algorithm is presented in Algorithm 1. The time limit for solving the ε-r-lbcutM-h heuristic is set to 0.1 seconds. When separating at fractional solutions, we only implemented it at the root node. To avoid tailing-off effects, we stopped adding cuts at the root node when the LP relaxation has not improved over the previous three iterations by more than 0.01%. Finally, when solving the exact separation in line 14−18 of the pseudo-code, we broke the loop as soon as a violation is detected in order not to spend time on separating the solutions exactly when this may be done heuristically in the next iteration.

Table 3: (Minimum, average, maximum) edge numbers in failing edge sets to be considered in the problem for different values of P and p_{base}.

	Reliability level (P)									
p_{base}	0.90	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
0.65	(1, 1.75, 2)	(1, 1.85, 2)	(1, 1.89, 2)	(1, 1.93, 2)	(1, 1.94, 2)	(1, 1.96, 2)	(1, 1.98, 3)	(1, 2.6, 3)	(1, 2.85, 3)	(1, 3.35, 4)
0.70	(1, 1, 1)	(1, 1, 1)	(1, 1.52, 2)	(1, 1.8, 2)	(1, 1.9, 2)	(1, 1.92, 2)	(1, 1.95, 2)	(1, 1.96, 2)	(1, 2.5, 3)	(1, 2.88, 3)
0.75	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1.03, 2)	(1, 1.77, 2)	(1, 1.9, 2)	(1, 1.94, 2)	(1, 1.96, 2)	(1, 2.66, 3)
0.80	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1.81, 2)	(1, 1.93, 2)	(1, 1.97, 2)
0.85	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1.49, 2)	(1, 1.94, 2)
0.90	(0, 0, 0)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)
0.95	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)	(1, 1, 1)
(min, avg, max)	(0, 0.82, 2)	(0, 0.98, 2)	(0, 1.06, 2)	(0, 1.10, 2)	(0, 1.12, 2)	(0, 1.24, 2)	(1, 1.40, 3)	(1, 1.62, 3)	(1, 1.82, 3)	(1, 2.11, 4)

Algorithm 1: Separation Algorithm

Input: Demand set R
Output: A set of violated constraints ($CutSet$)

begin
  /* Heuristic cut generation */
  for $i \in \{p, b\}$ **do** // For 'primal' (p) and 'backup' (b)
    $CutSet^i \leftarrow \emptyset$
    for $r \in R$ **do**
      $G \leftarrow G_r^i$
      if $H_r^i \leq 3$ **then**
        $tempCutSet^i$ += ε-lbcut3(G) // Section 6.4 (this algorithm is exact when $i =$ 'p')
      else
        $tempCutSet^i$ += ε-minCut-h(G) // Section 6.2
        $CutSet^i$ += ε-r-lbcutM-h(G) // Section 6.1
  for $cut \in tempCutSet^b$ **do** // Strengthening the backup cuts
    $CutSet^b$ += Knapsack-h(cut, G_r^b) // Section 6.3
  if $CutSet == null$ **then** // If no cut is found by the heuristics
    /* Exact separation */
    for $r \in R$ **do**
      if $H_r^p > 3$ **then**
        $CutSet^p$ += r-lbcutM(G_r^p) // Section 6.1
    for $r \in R$ **do**
      $CutSet^b$ += r-lbcutM(G_r^b) // Section 6.1
  return $CutSet^p \cup CutSet^b$

7.3 Experimental design

Our experimental design consists of three groups, which involves a total of 90,540 experiment runs. In the first group, we compare the solution efficiency Models M1 and M3, and test different parameter settings of the separation problem. Observe that M2 is an intermediate model between M1 and M3. It is inferior to Model M3, because it is based on the same enumerative scheme as M1. Therefore, in this first group of computational study, we compare the performances of M1 and M3. Following the convention used in the literature, we use the same hop limit for all demands for the same problem instance. For each instance, we use $(H^p, H^b) = (H_{min} + \Delta^p, H_{min} + \Delta^p + \Delta^b)$ for all $\Delta^p, \Delta^b \in \{0, 1, 2\}$. Furthermore, we test a resilience level $P \in \{0.9, 0.95, 0.99\}$, $p_{base} \in \{0.85, 0.9, 0.95\}$ and $F \in \{c, r\}$. This setting, as presented in Table 4, gives 162 instances for each network. We carried out tests on *10 networks*, as described in Section 7.1, yielding 1620 instances per method.

Using these 1620 instances, we tested the performance of model M1, and M3 under 12 different algorithmic settings, as presented in Table 5. In this table, the first column represents the setting number, 'Frac.Sep.' in the second column indicates if fractional separation is implemented at the root node or not. Columns 3−5 indicate whether the ε-lbcutM-h, ε-minCut-h and Knapsack-h heuristics are implemented.

Table 4: Parameter settings for M1 and M3 models in the first group of experiments.

Parameter	Possible values
Δ^p	$\{0, 1, 2\}$
Δ^b	$\{0, 1, 2\}$
P	$\{0.9, 0.95, 0.99\}$
p_{base}	$\{0.85, 0.9, 0.95\}$
F	$\{c, r\}$

Table 5: Hyperparameter testing for the M3 model in the first group of experiments.

Setting	Frac.Sep.	ε-lbcutM-h	ε-minCut-h	Knapsack-h
1	0	0	0	0
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	1	0
6	0	1	1	1
7	1	0	0	0
8	1	0	1	0
9	1	0	1	1
10	1	1	0	0
11	1	1	1	0
12	1	1	1	1

0 indicates exclusion, and 1 inclusion.

The first experiments demonstrate the computational superiority of M3 over M1, and also yield the best performing setting for the algorithm. Using this setting, in the second part of the experiments, we carried out experiments on a wider range of input values, as shown in Table 6. In this table, the same hop limits as in the first group of experiments are tested, but additionally resilience levels from 0.90 to 0.99 with increments of 0.1 are also tested. The values tested for the base probability p_{base} vary between 0.6 and 0.95 with increments of 0.05. These yields 1440 different settings, which were tested on the previously introduced '10 networks'. Thus, in the second group of experiments, the total number of runs is 14,400.

Table 6: Parameter settings for M3 model in the second group of experiments.

Parameter	Possible values
Δ^p	$\{0, 1, 2\}$
Δ^b	$\{0, 1, 2\}$
P	$\{0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99\}$
p_{base}	$\{0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95\}$
F	$\{c, r\}$

In the third and last group of experiments, we tested the best performing algorithm on the complete set of 350 networks using the parameter settings in Table 4. Since 10 networks were already tested in the first group, we ran experiments with these 162 parameter settings in Table 4 on the remaining 340 networks, yielding 55,080 instances.

7.4 Computational performance comparison of the models

The first group experiment results that compare the solution performance of M1 model versus M3 model under 12 different settings are presented in Table 7. The first two show present the setting and the model solved, where setting 0 represents Model M1 with default settings of CPLEX, and settings $1-12$ represents the settings in Table 5 corresponding to different separation algorithm implementations. For each row, there are 1620 solved instances, as indicated in the third column. The fourth column shows the percentage of solved instances (which includes those solved to optimality, and those that are found

to be infeasible). The average optimality gap percentage and the average solution time in seconds are presented under the fifth and sixth columns, respectively. The number of instances solved to optimality and the number of instances shown to be infeasible are presented in seventh and eighth columns. The number of instances with a feasible solution but with a positive gap is presented under column nine. If no integer feasible solution is found, it is classified as unknown in the 10th column. Finally, computer memory may be a problem for M1, and the number of instances for which the solution was terminated due memory limit is shown under the 11th column. The first observation in this table is that M3 performs poorly for settings 1−6, which involves instances with integer separation only. In settings 7−12, fractional separation is also included. Among these six settings, 12th performs the best with 94.20% solved instances, 2.45% average gap, and 581.5 seconds of solution time. Note that these are the best performances among all settings. When compared to Model M1, we observe a clear superiority of M3. Model M1 could solve 64.20% of the instances, and CPLEX could not identify an integer solution in 294 instances among 1620, and ran into a memory problem in 109 of them. From this point on, the 12th setting is used as our algorithm. Recall that this setting involves separation at fractional solutions at the root node, and implementing the ε-lbcutM-h heuristic, ε-minCut-h heuristic, and Knapsack-h heuristic when separating solutions.

Table 7: First group experiment results comparing the solution performance of M1 model versus M3 model under 12 different settings.

						Optimization Status				
Setting	Model	# instances	Solved (%)	Avg.Gap (%)	Avg.Sol.Time (s)	Optimal	Infeasible	Feasible	Unknown	Memory
0	M1	1620	64.20%	2.78%†	2828.7	1034	6	177	294	109
1	M3	1620	7.41%	80.89%	6687.8	114	6	1500	0	0
2	M3	1620	34.20%	47.36%	4773.4	548	6	1066	0	0
3	M3	1620	34.26%	47.14%	4765.6	549	6	1065	0	0
4	M3	1620	7.35%	80.93%	6709.4	113	6	1501	0	0
5	M3	1620	33.52%	47.42%	4830.6	537	6	1077	0	0
6	M3	1620	33.70%	47.78%	4834.0	540	6	1074	0	0
7	M3	1620	90.62%	4.79%	921.8	1462	6	152	0	0
8	M3	1620	91.17%	4.16%	888.6	1471	6	143	0	0
9	M3	1620	90.93%	4.03%	914.6	1467	6	147	0	0
10	M3	1620	92.90%	3.49%	672.0	1499	6	115	0	0
11	M3	1620	93.33%	3.37%	687.3	1506	6	108	0	0
12	M3	1620	94.20%	2.45%	581.5	1520	6	94	0	0

† The optimality gap reported does not include 403 instances with 'Unknown' and 'Memory' status.

Table 8 presents the results of the first group experiments with the best performing M3 model (12th setting) for different resilience levels (P) and base probability (p_{base}) settings on 10 networks with cost-dependent and random edge probability settings. The problem is generally harder to solve on graphs with cost-dependent edge probabilities ($F = c$) than those with randomly assigned edge probabilities ($F = r$). The average solution time for $F = c$ and $F = r$ is 749.7 and 413.2 seconds, respectively, while the average gap is 3.60% and 1.29%, respectively. We observe that the instances are harder to solve for an increasing resilience level (P) and for a decreasing base probability (p_{base}. Clearly, this leads to more potential cases to address, and increasing numbers of constraints to separate, which yields higher solution times and higher optimality gaps.

The primal and backup hop limits, Δ^p and Δ^b, have a significant impact on solution efficiency. Table 9 presents the results of the first group of experiments with the best performing M3 model for different Δ^p and Δ^b settings on 10 networks with cost-dependent and random edge probability settings. Solving the instances with $\Delta^p = 2$ and $\Delta^b = 2$ on average takes 1952.9 seconds and yields an average optimality gap of 11.73%, which is significantly higher than the values obtained with $\Delta^p = 0$ and $\Delta^b = 0$.

Table 10 presents the results for the first group of experiments for each of the 10 networks. Clearly, solving the problem on some networks can be quite challenging. In particular, network C10x10-5-1_10_10_50-2_7-2 yields the highest average gap, and network C10x10-5-1_10_10_50-2_7-3 yields the

Table 8: First group experiment results of best performing M3 model (12th setting) for different resilience level (P) and base probability (p_{base}) settings on 10 networks with cost-dependent and random edge probability settings.

Resilience level (P)	Base probability (p_{base})	Cost-dependent edge probability ($F = c$)		Random edge probability ($F = r$)		Average	
		Avg.Sol. Time (s)	Avg.Gap (%)	Avg.Sol. Time (s)	Avg.Gap (%)	Avg.Sol. Time (s)	Avg.Gap (%)
0.90	0.85	143.6	0.00%	94.1	0.00%	118.8	0.00%
	0.90	7.5	0.00%	8.0	0.00%	7.7	0.00%
	0.95	7.8	0.00%	7.4	0.00%	7.6	0.00%
0.95	0.85	474.6	1.39%	299.6	0.11%	387.1	0.75%
	0.90	227.6	0.00%	164.1	0.06%	195.8	0.03%
	0.95	7.5	0.00%	7.3	0.00%	7.4	0.00%
0.99	0.85	3755.1	20.25%	1952.0	6.78%	2853.6	13.51%
	0.90	1226.0	7.24%	674.7	3.09%	950.3	5.16%
	0.95	898.0	3.52%	511.4	1.62%	704.7	2.57%
Average		749.7	3.60%	413.2	1.29%	581.5	2.45%

Table 9: First group experiment results of best performing M3 model (12th setting) for different Δ^p and Δ^b settings on 10 networks with cost-dependent and random edge probability settings.

Δ^p	Δ^b	Cost-dependent edge probability ($F = c$)		Random edge probability ($F = r$)		Average	
		Avg.Sol. Time (s)	Avg.Gap (%)	Avg.Sol. Time (s)	Avg.Gap (%)	Avg.Sol. Time (s)	Avg.Gap (%)
0	0	65.1	0.00%	33.3	0.00%	49.2	0.00%
0	1	525.1	1.22%	124.7	0.03%	324.9	0.63%
0	2	443.7	1.66%	146.6	0.04%	295.1	0.85%
1	0	410.1	0.87%	157.7	0.05%	283.9	0.46%
1	1	657.4	2.51%	268.9	0.58%	463.1	1.54%
1	2	943.3	5.98%	581.8	2.08%	762.6	4.03%
2	0	610.3	2.36%	330.0	0.35%	470.2	1.36%
2	1	1139.7	6.08%	775.0	2.20%	957.3	4.14%
2	2	1952.9	11.73%	1300.4	6.31%	1626.7	9.02%
Average		749.7	3.60%	413.2	1.29%	581.5	2.45%

highest average solution time. The last five networks have $H_{min} = 7$ while the first five networks have $H_{min} = 5$, which yields more combinations to be considered for the former five networks. This is the main reason for the solution performance difference between the first five and second five networks.

Finally, Table 11 presents the average objective function value (the cost) for different reliability level (P) and base probability (p_{base}) in the first group of experiments. Clearly, when the base probability is greater than or equal to the reliability level requested, no backup path is required, which leads to a minimum cost. For a base probability level of 0.85, the costs are 155.5, 174.5 and 339.0 for $P = 0.90$, $P = 0.95$, and $P = 0.99$, respectively. Increasing the reliability level from 0.90 to 0.95 level increases the average cost only by 12.4%; increasing it from 0.95 to 0.99 level yields a cost increase of 93.9%.

7.5 Computational performance of the best performing algorithm under various settings

Table 12 shows the computational results of the best performing algorithm in the second group of experiments. Among 14,400 instances, 11,500 are solved to optimality, and 347 are proven to be infeasible, yielding a 82.3% solution performance. There are two unknown solutions among 14,400 instances. Note that the instances in the second group are significantly harder than those of the first group, because we consider base probabilities of 0.60 and higher with increments of 0.05. Detailed computational results for the same set of instances are presented in Table 13. When $p_{base} = 0.95$,

meaning a highly reliable network, the problem becomes easier (1774 instances are solved to optimality among 1800 instances). On the other hand, the number of solved instances decreases to only 958 when $p_{base} = 0.60$. Clearly, increasing the reliability level makes the problem more difficult. Observe that the experiment set with $p_{base} = 0.6$, and $P = 0.99$ is the hardest, with only 26 instances solved to optimality and 47 instances infeasible among 180 instances. Since these 26 instances are solved to optimality across all settings for P and p_{base}, we next investigate the optimal objective function values of these instances by varying the P and p_{base} values. Table 14 reports the average optimal objective function values of these 26 instances for all P and p_{base} values considered in the second group of experiments. The same values are shown in a three dimensional bar chart in Figure 1. Observe that when the base probability is high ($p_{base} \geq 0.90$), an increase in reliability level does not significantly increase the cost, whereas when the base probability is low, the cost is high, and the impact of increasing reliability level is more pronounced.

Table 10: First group experiment results of best performing M3 model (12th setting) on 10 networks with cost-dependent and random edge probability settings.

	Cost-dependent edge probability ($F = c$)		Random edge probability ($F = r$)		Average	
Network	Avg.Sol. Time (s)	Avg.Gap (%)	Avg.Sol. Time (s)	Avg.Gap (%)	Avg.Sol. Time (s)	Avg.Gap (%)
C10x10-5-1_10_10_50-2_5-1	171.6	0.31%	75.6	0.00%	123.6	0.16%
C10x10-5-1_10_10_50-2_5-2	233.1	0.27%	71.7	0.00%	152.4	0.13%
C10x10-5-1_10_10_50-2_5-3	22.5	0.00%	13.0	0.00%	17.8	0.00%
C10x10-5-1_10_10_50-2_5-4	1360.6	8.05%	521.0	2.02%	940.8	5.03%
C10x10-5-1_10_10_50-2_5-5	300.3	0.49%	137.4	0.00%	218.9	0.25%
C10x10-5-1_10_10_50-2_7-1	701.1	2.48%	265.5	0.51%	483.3	1.49%
C10x10-5-1_10_10_50-2_7-2	1637.4	10.79%	831.7	3.66%	1234.6	7.22%
C10x10-5-1_10_10_50-2_7-3	1852.8	9.10%	1772.3	5.93%	1812.5	7.51%
C10x10-5-1_10_10_50-2_7-4	976.6	4.06%	402.8	0.83%	689.7	2.44%
C10x10-5-1_10_10_50-2_7-5	241.3	0.46%	40.4	0.00%	140.8	0.23%
Average	749.7	3.60%	413.2	1.29%	581.5	2.45%

Table 11: Average objective function value for different reliability levels (P) and base probabilities (p_{base}) in the first group of experiments.

	Reliability level (P)		
p_{base}	0.90	0.95	0.99
0.85	155.5	174.8	339.0
0.90	81.9	163.6	200.4
0.95	81.9	81.9	184.5

Table 12: Computational results of the second group experiments.

Optimization Status	# instances	Avg.Sol.Time (s)	Avg.Gap (%)
Optimal	11500	422.4	0.0%
Infeasible	347	103.4	0.0%
Feasible	2551	TL† (7200)	43.8%
Unknown	2	TL† (7200)	-
Total	14400	1616.6	7.8%

† 'TL' is for Time Limit.

Table 13: Detailed computational results of the second group experiments reporting the number of instances solved per optimization status (Optimal, Infeasible, Feasible, Unknown) for different reliability level (P) and base probability (p_{base}).

Optimization		Reliability level (P)										
Status	p_{base}	0.90	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	Total
Optimal	0.60	137	133	125	119	119	97	81	70	51	26	958
	0.65	156	139	139	122	121	118	109	87	60	35	1086
	0.70	174	169	163	147	139	127	116	108	87	52	1282
	0.75	175	175	170	170	173	150	132	120	109	71	1445
	0.80	179	176	174	171	165	168	164	130	117	102	1546
	0.85	179	177	179	177	175	171	172	168	154	114	1666
	0.90	180	180	180	179	179	177	171	173	164	160	1743
	0.95	180	180	180	180	180	180	177	177	176	164	1774
Infeasible	0.60	1	3	6	6	6	8	11	22	28	47	138
	0.65	0	1	1	3	6	6	8	11	22	46	104
	0.70	0	0	0	1	1	3	6	8	11	25	55
	0.75	0	0	0	0	0	0	1	6	8	11	26
	0.80	0	0	0	0	0	0	0	1	6	11	18
	0.85	0	0	0	0	0	0	0	0	0	6	6
	0.90	0	0	0	0	0	0	0	0	0	0	0
	0.95	0	0	0	0	0	0	0	0	0	0	0
Feasible	0.60	42	44	49	55	55	75	88	88	100	107	703
	0.65	24	40	40	55	53	56	63	82	98	99	610
	0.70	6	11	17	32	39	50	58	64	82	103	462
	0.75	5	5	10	10	7	30	47	54	63	98	329
	0.80	1	4	6	9	15	12	16	49	57	67	236
	0.85	1	3	1	3	5	9	8	12	26	60	128
	0.90	0	0	0	1	1	3	9	7	16	20	57
	0.95	0	0	0	0	0	0	3	3	4	16	26
Unknown	0.60	0	0	0	0	0	0	0	0	1	0	1
	0.70	0	0	0	0	1	0	0	0	0	0	1
Total		1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	14400

Table 14: The average optimal objective function values of the 26 instances which are solved to optimality for all p_{base} and P values in the second group of experiments (see Table 13).

	Reliability level (P)										
p_{base}	0.90	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	Average
0.60	202.42	206.62	212.85	216.38	225.15	262.69	290.54	338.19	374.42	478.73	280.80
0.65	172.81	186.88	198.96	207.04	214.08	219.46	245.58	284.77	331.73	412.85	247.42
0.70	151.08	152.54	164.92	184.77	197.04	208.77	216.73	241.08	286.96	357.31	216.12
0.75	145.04	151.08	151.88	153.35	156.88	181.12	203.46	216.19	247.96	309.58	191.65
0.80	135.65	140.62	145.04	151.08	152.54	156.23	157.42	189.92	215.81	262.73	170.70
0.85	120.81	129.31	133.08	136.69	145.15	151.19	154.27	157.42	168.77	221.31	151.80
0.90	74.88	92.81	112.50	118.12	129.42	135.62	145.04	152.54	157.42	159.19	127.75
0.95	74.88	74.88	74.88	74.88	74.88	74.88	112.50	129.31	145.04	157.42	99.36
Average	134.70	141.84	149.26	155.29	161.89	173.75	190.69	213.68	241.01	294.89	185.70

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

The previous experiments were all executed on 10 networks. The third group of experiments are on the complete set of 350 networks for different P, p_{base}, Δ^p, and Δ^b settings, yielding a total of 55,080 instances. Table 15 shows the average solution time in seconds, the number of solved instances, the percentage of solved instances and the average gap per optimization status. Among all instances, 36,719 are solved to optimality, and 1343 are proven to be optimal, yielding a total of 69.1% solved instances. The average solution time of all instances is 2486.3 seconds and the average gap is 17.61%. Detailed results classified by P and p_{base} are presented in Table 16. Note that the algorithm performs best

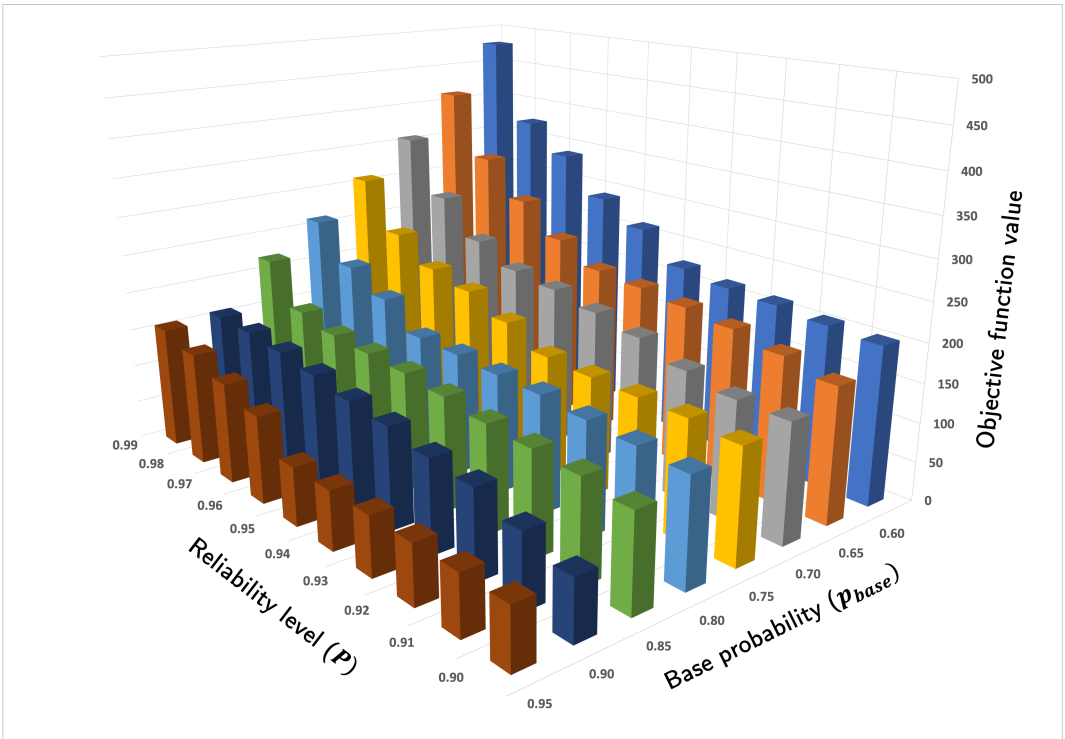

Figure 1: The average optimal objective function values of the 26 instances which are solved to optimality for all p_{base} and P values in the second group of experiments (see Table 14).

Table 15: Overview of the computational results of the best performing algorithm on all 350 networks in the dataset in the third group of experiments.

Optimization Status	Avg.Sol.Time (s)	# Solved Instances	% Solved Instances	Avg.Gap (%)
Optimal	380.9	36719	66.7%	0.00%
Infeasible	16.2	1343	2.4%	0.00%
Feasible	TL† (7200)	16316	29.6%	55.15%
Unknown	TL† (7200)	702	1.3%	-
Total	2486.3	55080	100.0%	17.61%

† 'TL' is for Time Limit.

for higher p_{base} values and lower P values among the values considered in this group of experiments. The results are categorized per set of networks in Table 17. While there is no clear difference between sets C, D, E and R in terms of computational results, it is more difficult to solve the model for some instances in each set than others. The same results as categorized by P, p_{base}, Δ^p, and Δ^b are shown in Table 18. We observe that smaller Δ^p, and Δ^b values lead to a computational better performance. In particular, the algorithm performs the best when $P \leq 0.95$, $p_{base} = 0.95$, $\Delta^p = 0$, and $\Delta^b \leq 1$ among the settings considered in this study on the networks under consideration.

Table 16: Detailed results of the best performing algorithm on all 350 networks in the dataset in the third group of experiments.

		Reliability Level (P)			
p_{base}	Optimization status	0.90	0.95	0.99	Total
0.85	Optimal	4553	3261	1847	9661
	Infeasible	113	183	324	620
	Feasible	1454	2673	3265	7392
	Unknown	0	3	684	687
0.90	Optimal	5903	3941	2574	12418
	Infeasible	54	142	219	415
	Feasible	163	2037	3318	5518
	Unknown	0	0	9	9
0.95	Optimal	5890	5905	2845	14640
	Infeasible	54	54	200	308
	Feasible	175	159	3072	3406
	Unknown	1	2	3	6
Total		18360	18360	18360	55080

Table 17: Average percentage of the instances with 'Optimal', 'Infeasible', 'Feasible', and 'Unknown' optimization status when solved by the best performing algorithm for different sets of networks considered in this study.

Dataset name	Optimal	Infeasible	Feasible	Unknown
C-1	89.5%	0.1%	10.2%	0.2%
C-2	68.6%	0.1%	31.0%	0.4%
C-3	87.7%	0.3%	11.8%	0.2%
C-4	67.7%	0.0%	31.9%	0.4%
C-5	42.4%	0.2%	55.0%	2.4%
C-6	87.5%	0.2%	12.3%	0.0%
C-7	66.4%	0.1%	32.9%	0.6%
C-8	50.6%	0.0%	47.2%	2.2%
C-9	40.1%	0.0%	56.2%	3.7%
D-1	99.1%	0.9%	0.0%	0.0%
D-2	98.5%	1.2%	0.3%	0.0%
D-3	74.1%	0.1%	24.9%	0.9%
D-4	52.8%	0.1%	42.8%	4.4%
D-5	20.8%	0.7%	74.0%	4.4%
E-1	91.4%	8.6%	0.0%	0.0%
E-2	75.8%	24.2%	0.0%	0.0%
E-3	92.1%	6.2%	1.7%	0.0%
E-4	72.7%	3.2%	23.3%	0.7%
E-5	86.0%	7.5%	6.4%	0.0%
E-6	62.5%	3.3%	32.3%	1.9%
E-7	74.4%	0.5%	24.9%	0.1%
E-8	55.8%	0.1%	40.9%	3.2%
E-9	80.4%	3.3%	16.3%	0.0%
E-10	52.2%	4.3%	40.1%	3.3%
E-11	62.0%	1.7%	36.3%	0.0%
E-12	29.6%	2.1%	64.3%	4.0%
R-1	93.8%	6.2%	0.0%	0.0%
R-2	90.7%	6.3%	3.0%	0.0%
R-3	87.7%	8.1%	4.2%	0.0%
R-4	63.7%	2.1%	34.0%	0.2%
R-5	92.2%	3.1%	4.6%	0.1%
R-6	59.4%	3.6%	35.9%	1.1%
R-7	58.8%	13.3%	27.8%	0.1%
R-8	41.0%	6.9%	48.0%	4.1%
R-9	78.0%	4.4%	17.3%	0.2%
R-10	48.1%	3.8%	44.9%	3.1%
R-11	36.9%	33.3%	29.8%	0.0%
R-12	34.6%	0.0%	60.7%	4.7%
Total	66.7%	2.4%	29.6%	1.3%

Table 18: Average percentage of the instances with 'Optimal' and 'Infeasible' optimization status when solved by the best performing algorithm on all 350 networks in the dataset in the third group of experiments for varying P, p_{base}, Δ^p, and Δ^b settings.

	Reliability	$\Delta^p = 0$			$\Delta^p = 1$			$\Delta^p = 2$			
p_{base}	Level (P)	$\Delta^b = 0$	$\Delta^b = 1$	$\Delta^b = 2$	$\Delta^b = 0$	$\Delta^b = 1$	$\Delta^b = 2$	$\Delta^b = 0$	$\Delta^b = 1$	$\Delta^b = 2$	Average
0.85	0.90	90.00%	87.65%	85.15%	79.85%	75.29%	72.50%	68.09%	63.38%	64.26%	76.24%
	0.95	77.94%	75.44%	69.26%	58.68%	51.32%	50.00%	44.41%	40.29%	39.12%	56.27%
	0.99	65.00%	48.97%	42.06%	38.09%	30.29%	26.47%	26.76%	22.21%	19.41%	35.47%
0.90	0.90	99.71%	99.71%	99.71%	98.82%	98.09%	98.24%	94.41%	93.24%	94.12%	97.34%
	0.95	82.79%	80.59%	79.41%	68.24%	64.71%	63.82%	57.94%	51.32%	51.62%	66.72%
	0.99	73.97%	66.62%	55.74%	48.38%	40.00%	33.97%	35.74%	30.15%	26.18%	45.64%
0.95	0.90	99.71%	99.71%	99.71%	98.09%	97.50%	98.09%	93.97%	93.53%	93.82%	97.12%
	0.95	99.71%	99.71%	99.85%	98.38%	97.94%	98.24%	95.00%	93.68%	93.82%	97.37%
	0.99	75.59%	70.29%	62.65%	52.65%	43.53%	38.97%	39.56%	34.26%	30.29%	49.75%
Average		84.93%	80.96%	77.06%	71.24%	66.52%	64.48%	61.76%	58.01%	56.96%	69.10%

8 Conclusion

We have introduced, modeled and solved the Network Design Problem with Vulnerability Constraints and Edge Reliability (NDPVC-PER). The problem is a natural extension of the NDPVC by considering edge failures in the network. The network designer is assumed to have a certain reliability level requirement, and the network is designed by ensuring that there exists a backup path for all edge failure scenarios having a higher probability of occurrence than the level dictated by the designer. The problem is a generalization of the NDPVC, and is computationally more difficult to solve. In particular, the models built for NDPVC which do not require enumeration do not solve the NDPVC-PER.

In this paper, we have built a naive model that enumerates all edge failure scenarios and explicitly builds paths for them. We also developed a model that does not require edge set enumeration, and involves building resilient length-bounded cuts. These cuts disconnect all paths of a certain length connecting a given O-D pair after the failure of edges in the network. We have presented this new separation problem for our model, and we have solved it using various techniques. We have developed some heuristic techniques for this separation problem, and we have guaranteed an exact separation by solving the problem by a mathematical model.

We have thoroughly tested the computational efficiency of the models using 90,540 instances. We have identified the best performing settings for our algorithm, and we have solved the problem for a large variety of input parameters. Our findings show that for the graphs and settings we have considered, increasing the reliability level from 0.90 to 0.95 level increases the average cost only by 12.4%, while increasing it from 0.95 to 0.99 level yields a much larger cost increase of 93.9%. Therefore, we have shown that very high requirements dictated by network designers in terms of reliability may lead to excessive costs.

References

Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Upper Saddle River, NJ, 1993.

Alcoa Fujikura ltd. Reliability of fiber optic cable systems, 2001. URL https://www.southern-telecom.com/content/dam/southern-telecom/pdfs/AFL-Reliability.pdf. Accessed June 12, 2023.

Okan Arslan, Oya Ekin Karaşan, A Ridha Mahjoub, and Hande Yaman. A branch-and-cut algorithm for the alternative fuel refueling station location problem with routing. Transportation Science, 53(4):1107–1125, 2019.

Okan Arslan, Ola Jabali, and Gilbert Laporte. A flexible, natural formulation for the network design problem with vulnerability constraints. INFORMS Journal on Computing, 32(1):120–134, 2020.

Anantaram Balakrishnan and Kemal Altinkemer. Using a hop-constrained model to generate alternative communication network design. ORSA Journal on Computing, 4(2):192–205, 1992.

Fatiha Bendali, I Diarrassouba, Ali Ridha Mahjoub, M Didi Biha, and Jean Mailfert. A branch-and-cut algorithm for the k-edge connected subgraph problem. Networks, 55(1):13–32, 2010.

Quentin Botton, Bernard Fortz, Luis Gouveia, and Michael Poss. Benders decomposition for the hop-constrained survivable network design problem. INFORMS Journal on Computing, 25(1):13–26, 2013.

Geir Dahl and Mechthild Stoer. A cutting plane algorithm for multicommodity survivable network design problems. INFORMS Journal on Computing, 10(1):1–11, 1998.

Geir Dahl, Luis Gouveia, and Cristina Requejo. On formulations and methods for the hop-constrained minimum spanning tree problem. In Mauricio G. C. Resende and Panos M. Pardalos, editors, Handbook of Optimization in Telecommunications, pages 493–515. Springer US, Boston, MA, 2006.

Ibrahima Diarrassouba and Ridha A Mahjoub. Polyhedral investigation of k edge-connected l-hop-constrained network design problem. HAL, 2023. hal-04051494.

Eland Cables. What are the main causes of electrical cable failure? URL https://www.elandcables.com/the-cable-lab/faqs/faq-what-are-the-main-causes-of-electrical-cable-failure. Accessed June 12, 2023.

Bernard Fortz and Martine Labbé. Polyhedral results for two-connected networks with bounded rings. Mathematical Programming, 93(1):27–54, 2002.

Bernard Fortz, Martine Labbé, and Francesco Maffioli. Solving the two-connected network with bounded meshes problem. Operations Research, 48(6):866–877, 2000.

Bernard Fortz, Ali Ridha Mahjoub, S Thomas McCormick, and Pierre Pesneau. Two-edge connected subgraphs with bounded rings: Polyhedral results and branch-and-cut. Mathematical Programming, 105(1):85–111, 2006.

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. Networks, 80(2):178–192, 2022.

Luis Gouveia. Multicommodity flow models for spanning trees with hop constraints. European Journal of Operational Research, 95(1):178–190, 1996.

Luis Gouveia. Using variable redefinition for computing lower bounds for minimum spanning and Steiner trees with hop constraints. INFORMS Journal on Computing, 10(2):180–188, 1998.

Luis Gouveia and Markus Leitner. Design of survivable networks with vulnerability constraints. European Journal of Operational Research, 258(1):89–103, 2017.

Luis Gouveia, Martim Joyce-Moniz, and Markus Leitner. Branch-and-cut methods for the network design problem with vulnerability constraints. Computers & Operations Research, 91:190–208, 2018.

Martin Grötschel, Clyde L Monma, and Mechthild Stoer. Design of survivable networks. In Michael Ball, Tom Magnanti, Clyde Monma, and George Nemhauser, editors, Network Models, volume 7 of Handbooks in Operations Research and Management Science, pages 617–672. Elsevier, Amsterdam, 1995.

Hervé Kerivin and A Ridha Mahjoub. Design of survivable networks: A survey. Networks, 46(1):1–21, 2005.

Thorsten Koch and Alexander Martin. Solving steiner tree problems in graphs to optimality. Networks, 32(3): 207–232, 1998.

A Ridha Mahjoub and S Thomas McCormick. Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. Mathematical programming, 124(1-2):271–284, 2010.

Hassan Mashad Nemati. Data-Driven Methods for Reliability Evaluation of Power Cables in Smart Distribution Grids. PhD thesis, Halmstad University Press, 2017.

Roxtec Cable Seals. Reliable cable seals for telecom networks, 2023. URL https://www.roxtec.com/en/knowledge-library/application-areas/reliable-cable-seals-for-telecom-networks/. Accessed June 12, 2023.

Mechthild Stoer. Design of survivable networks, volume 1531, Lecture Notes in Mathematics, Springer, Berlin Heidelberg, 1992.

Richard T Wong. Telecommunications network design: Technology impacts and future directions. Networks, 77(2):205–224, 2021.