Manuscript Click here to view linked References

Lexicographic Minimum-Weight Minimum-Cardinality Length-Bounded Cuts: Applications in Network Optimization

Weiquan Wang[a], Yossiri Adulyasak[b], Okan Arslan[b], Jean-François Cordeau[b], Bobin Wang[a,*]

[a]Smart and Sustainable Transportation Lab (SSTL), Department of Mechanical and Industrial Engineering, Université Laval, Québec G1V 0A6, Canada

[b]HEC Montréal and GERAD, Montréal H3T 2A7, Canada

This paper studies the minimum-weight minimum-cardinality length-bounded cut (MWMC-LBC) problem, which seeks to identify a set of arcs in a graph whose removal eliminates all origin-destination paths satisfying a length bound or a resource limit, while first minimizing the total cut weight and then, among all minimum-weight cuts, minimizing the cut cardinality. In many applications, such as the Refueling Station Location Problem (RSLP) and the Network Design with Service Requirements problem (NDSR), feasibility depends on the existence of paths satisfying such length or resource limits. We develop models and algorithms for generating length-bounded cut inequalities and embed them into exact algorithms for the RSLP and NDSR. Computational experiments show that, for the RSLP, the proposed enhancements solve all 240 benchmark instances from four realistic transportation networks to optimality and reduce average solution times from 137.87 seconds to 9.40 seconds, corresponding to a 93.2% reduction relative to the original branch-and-cut algorithm. They also reduce the average number of dynamically generated length-bounded cuts by 91.9%. For the NDSR, the explicit inclusion of a small number of pre-generated lb-cuts reduces the average solution time by 15.9% over the 85 instances solved to optimality, while also modestly reducing the average optimality gap over all 90 benchmark instances. These results demonstrate that length-bounded cuts can substantially improve the computational performance of exact algorithms for large-scale network optimization problems.

Key words: length-bounded cut; refueling station location; network design; branch-and-cut

1. Introduction

Network connectivity and cut problems constitute one of the fundamental topics in graph theory, combinatorial optimization, and network design. In classical network optimization, the minimum cut problem aims to identify a set of edges or nodes with minimum total weight whose removal disconnects an origin node from a destination node. Owing to the max-flow min-cut theorem (Ford Jr and Fulkerson 1956), minimum cut problems can be solved efficiently in polynomial time.

*Corresponding author. E-mail address: bobin.wang@gmc.ulaval.ca (Bobin Wang).

In many practical applications, however, connectivity is meaningful only when the connecting paths satisfy a predefined resource or length limit. Examples include transportation systems with travel-distance restrictions, telecommunications networks with hop limits, and survivable network design problems with service-quality requirements (Mahjoub and McCormick 2010). In such settings, connecting the network is not sufficient; instead, one must ensure the existence of source-destination paths whose lengths do not exceed a prescribed threshold. This requirement gives rise to the length-bounded cut (lb-cut) problem (Lovász et al. 1978, Exoo 1983, Baier et al. 2010), which is often used as a modeling tool for solving such network optimization problems.

Unlike the classical minimum cut problem, the minimum length-bounded cut problem is significantly more challenging. Several studies have shown that the minimum length-bounded cut problem becomes NP-hard for general length bounds and remains difficult to approximate (Baier et al. 2010, Golovach and Thilikos 2011). These results highlight the importance of developing fast-performing algorithms for solving length-bounded cut problems.

Length-bounded cuts arise naturally as separation subproblems in branch-and-cut algorithms designed to solve noncompact formulations for various problems. One example is the Refueling Station Location Problem with Routing (RSLP), in which an origin-destination (OD) pair can be covered only if there exists a feasible route whose length does not exceed a prescribed deviation tolerance. Arslan et al. (2019) showed that the separation problem of their branch-and-cut algorithm can be formulated as a length-bounded cut problem in a transformed network. Length-bounded cuts also play an important role in network design problems with service-quality requirements. In the Network Design Problem with Vulnerability Constraints (NDPVC), communication between OD pairs must be maintained through paths satisfying hop limits even after network failures. Arslan et al. (2020) developed noncompact branch-and-cut formulations whose separation procedures rely on identifying violated length-bounded cuts. Similar structures appear in the Network Design with Service Requirements (NDSR) problem, where feasible paths must satisfy prescribed service constraints (Balakrishnan et al. 2017, Gudapati et al. 2022). Gudapati et al. (2025) study the hop-constrained survivable network design problem (HCSNDP) using path-based and layered-graph flow formulations. Their work highlights the importance of path-length constraints in network design, but it does not solve minimum-weight lb-cut separation problems.

A common feature of these applications is that feasibility is determined by the existence of paths satisfying explicit length or resource limits. In large-scale instances, hundreds or even thousands of lb-cut separation problems may need to be solved during a single branch-and-cut run. Consequently, the efficiency of the underlying branch-and-cut algorithms depends critically on the weight and cardinality of the generated lb-cuts.

Existing studies such as Arslan et al. (2019) observe that multiple minimum-weight cuts often exist and that their cardinalities can differ substantially. Among cuts with the same weight, smaller-cardinality cuts often produce sparser inequalities and may lead to improved computational performance. Although minimum-weight lb-cuts have been studied extensively, cardinality has been treated only through heuristic reduction procedures as in Arslan et al. (2020). To the best of our knowledge, the lexicographic minimum-weight minimum-cardinality variant has never been formulated and studied as a standalone optimization problem. Motivated by these observations, we investigate the minimum-weight minimum-cardinality length-bounded cut (MWMC-LBC) problem, which seeks a length-bounded cut of minimum total weight and, among all such cuts, one with minimum cardinality.

The contributions of this paper are summarized as follows.

(1) We introduce the minimum-weight minimum-cardinality length-bounded cut (MWMC-LBC) problem and develop exact and heuristic solution methods for the problem.

(2) We apply the proposed lb-cut generation methods to two state-of-the-art exact algorithms for the RSLP and the NDSR. For the RSLP, we strengthen the branch-and-cut algorithm of Arslan et al. (2019) by adding pre-generated structural cuts and improving integer- and fractional-solution separation. For the NDSR, we show that lb-cut inequalities

are implicitly enforced by the linking and path-selection constraints of the path-based formulation of Gudapati et al. (2022), but that explicitly adding a small number of pre-generated structural lb-cuts as redundant valid inequalities can meaningfully improve computational performance.

(3) Extensive computational experiments on large-scale benchmark instances demonstrate the effectiveness of the proposed methods. For the RSLP, the proposed enhancements reduce average solution times by about 93.2% and substantially reduce the number of dynamically generated lb-cuts. For the NDSR, explicitly adding the pre-generated lb-cuts reduces the average solution time by 15.9% over the 85 commonly solved instances and modestly improves the final optimality gaps.

The remainder of the paper is organized as follows. Section 3 presents the problem description, formulation, and algorithms of MWMC-LBC. Section 4 presents the application to NDSR. Section 5 presents the application to RSLP. Section 6 reports computational results and comparisons with state-of-the-art exact algorithms, and discusses when and why the proposed cuts are effective in reducing solution times.

2. Literature Review

In this section, we review the literature related to the minimum-weight length-bounded cut problem, the Refueling Station Location Problem (RSLP), and the Network Design with Service Requirements problem (NDSR).

2.1. Minimum-Weight Length-Bounded Cut Problem

Given a directed graph, two terminals s and t, and a length bound L, the minimum-weight length-bounded cut problem seeks a subset of arcs or vertices whose removal eliminates all s-t paths of length at most L. Early theoretical work originated from bounded-length extensions of Menger's theorem. Lovász et al. (1978) compared the minimum number of vertices required to destroy all bounded-length s-t paths with the maximum number of internally vertex-disjoint bounded-length paths, showing that the classical min-cut–max-flow equality generally fails under length restrictions. The corresponding edge-disjoint setting was later studied by Exoo (1983), who derived bounds on the gap between minimum bounded-length edge cuts and maximum collections of edge-disjoint bounded-length paths. From a computational perspective, Itai et al. (1982) established polynomially solvable cases for small length bounds and NP-completeness for most larger bounds in both vertex- and edge-disjoint variants. Related work has also considered cardinality as a secondary criterion in classical cut and cut-selection problems. Hamacher (1982) studied the classical minimum-cut problem and sought a minimum-capacity cut containing the fewest arcs, proposing an exact approach based on lexicographical network flows. In the context of combinatorial Benders decomposition, Codato and Fischetti (2006) developed a general framework for generating combinatorial Benders cuts, while Fischetti et al. (2010) further investigated the selection of such cuts and proposed a heuristic strategy favoring cuts with smaller support. Their computational results highlight the potential benefit of sparse cuts. These studies are closely related to the cut-selection perspective of our work: when multiple separating cuts are available, reducing their cardinality may improve the resulting formulation and computational performance. However, they do not address the exact lexicographic selection of a minimum-cardinality cut among minimum-weight length-bounded cuts, which is the focus of this study. Subsequent research has studied length-bounded cuts and flows more directly. Mahjoub and McCormick (2010) investigated maximum flow and minimum cut problems under path-length restrictions and showed that many classical min-cut properties no longer hold when only bounded-length paths are considered. A more comprehensive complexity analysis was provided by Baier et al. (2010), who studied length-bounded cuts and flows from an approximation perspective and showed that minimum length-bounded cut problems are NP-hard and difficult to approximate for general length bounds. Golovach and Thilikos (2011) subsequently investigated the parameterized complexity of bounded-length disjoint-path and cut problems.

Although these studies established the theoretical foundations of the LBC problem, relatively limited attention has been devoted to exact algorithms for computing minimum-weight length-bounded cuts in large-scale optimization applications. More recently, Arslan et al. (2019) developed an integer-programming formulation for identifying minimum-weight lb node cuts. Building on this work, Arslan et al. (2020) proposed a corresponding arc-based formulation for the minimum-weight lb-cut problem. Recent computational studies (Gudapati et al. 2025) on hop-constrained survivable network design continue to rely on cut-based and path-based formulations to enforce path-length requirements, highlighting the importance of efficient separation procedures in large-scale network optimization.

These studies highlight the growing importance of minimum-weight length-bounded cuts as a fundamental separation problem in modern network optimization. To the best of our knowledge, no study has formally investigated the minimum-weight minimum-cardinality length-bounded cut problem or developed exact algorithms for its solution.

2.2. Refueling Station Location Problem

The Refueling Station Location Problem (RSLP) has been extensively studied as a strategic planning tool for supporting the deployment of alternative-fuel and electric vehicle infrastructures. Existing studies can be broadly classified into two categories according to their objective functions: maximum-covering models and minimum-cost models.

Maximum-covering models seek to maximize the amount of covered traffic, flow, or vehicle-miles traveled given a limited infrastructure budget. Early studies focused on flow-refueling formulations and their extensions to multi-period planning, capacity constraints, and heterogeneous vehicle technologies (Chung and Kwon 2015, Zhang et al. 2017). More advanced models incorporate route deviations, user route-choice behavior, and bilevel interactions between infrastructure providers and travelers (He et al. 2018). From an exact optimization perspective, several branch-and-cut, branch-and-price, and decomposition-based algorithms have been developed to improve the scalability of maximum-covering formulations (Capar et al. 2013, Arslan and Karaşan 2016, Yıldız et al. 2016, Arslan et al. 2018, 2019, Göpfert and Bock 2019). Recent studies further consider nonlinear charging processes, congestion effects, demand uncertainty, heterogeneous charging technologies, and demand evolution over time (Xu and Meng 2020, Mahmutoğulları and Yaman 2023, Park and Lee 2024, Mahmutoğulları and Yaman 2024).

In contrast, minimum-cost models aim to identify the smallest set of stations, or the minimum infrastructure investment, required to satisfy a prescribed coverage level. Representative studies include capacitated charging station location models, multistage deployment frameworks, and integrated location-sizing formulations (Zhang et al. 2016, Xie et al. 2018, Kınay et al. 2021, 2023, Plaza et al. 2026).

Despite their modeling differences, most RSLP variants share a common computational challenge: determining whether an OD demand can be served under vehicle-range and path-deviation constraints. Consequently, the existence of a feasible route within a prescribed length limit becomes a fundamental block of exact solution approaches. This observation establishes a direct connection between the RSLP and the lb-cut problem studied in this paper.

2.3. Network Design Problem with Service Requirements

The Network Design with Service Requirements (NDSR) was introduced by Balakrishnan et al. (2017) as a generalization of the classical multicommodity network design problem. In addition to selecting network links and routing commodities, the NDSR requires that each commodity be assigned to a single origin-destination path satisfying one or more end-to-end service requirements, such as limits on transit time, latency, hop count, or other quality-of-service metrics. As a result, the problem jointly optimizes network design and routing decisions while enforcing path-length restrictions for every commodity. To solve the NDSR, Balakrishnan et al. (2017) developed a polyhedral branch-and-cut framework based on several classes of valid inequalities and tailored separation procedures. Although the proposed approach significantly improved solvability, the problem remained computationally challenging for large-scale

instances. More recently, Gudapati et al. (2022) proposed a path-based formulation together with a branch-and-cut-and-price algorithm that exploits feasible paths satisfying the service requirements. Their approach substantially expanded the size of solvable instances and established a new state of the art for the NDSR. Nevertheless, the authors reported that many large-scale instances remain difficult to solve to proven optimality, highlighting the computational challenges posed by service-constrained path requirements.

Existing NDSR approaches share a common characteristic: the feasibility of each commodity depends on the existence of a path satisfying prescribed length or resource limits. This path-feasibility structure naturally gives rise to length-bounded cuts, which can be exploited either as separation inequalities or as explicitly added structural inequalities, depending on the underlying formulation. This observation establishes a natural connection between the NDSR and the lb-cut problem studied in this paper.

2.4. Summary

As shown in Table 1, lb-cuts have become an important separation tool in a variety of network optimization applications. Existing studies mainly focus on identifying minimum-weight lb-cuts, while cardinality is handled through heuristic reduction procedures. Consequently, no existing approach guarantees a minimum-cardinality solution among all minimum-weight cuts. Motivated by this gap, this paper investigates the MWMC-LBC problem and develops exact and heuristic solution methods and demonstrates how lb-cut generation can be exploited within state-of-the-art exact algorithms for the RSLP and NDSR.

Table 1 Comparison of existing studies and the proposed MWMC-LBC approach

Study	Problem	Use lb-cut	Min W	Card. Red.	Min C
Balakrishnan et al. (2017)	NDSR	✗	✗	✗	✗
Arslan et al. (2019)	RSLP	✓	✓	Heur.	✗
Arslan et al. (2020)	NDPVC	✓	✓	Heur.	✗
Gudapati et al. (2022)	NDSR	✗	✗	✗	✗
Mahmutoğulları and Yaman (2023, 2024)	RSLP	✓	✓	Heur.	✗
Gudapati et al. (2025)	HCSNDP	✗	✗	✗	✗
This paper	RSLP	✓	✓	Exact/Heur.	✓
	NDSR	✓			

Min W: solves the minimum-weight lb-cut problem; Card. Red: cardinality reduction procedure; Min C: guarantees a minimum-cardinality solution among all minimum-weight lb-cuts.

3. Problem Description, Formulation and Algorithms of MWMC-LBC

Let $G = (N, A)$ be a directed graph, where N denotes the set of nodes and A denotes the set of arcs. Each arc $a \in A$ is associated with two parameters: a nonnegative length l_a and a weight $w_a \in [0, 1]$. Given an origin node $s \in N$, a destination node $t \in N$, and a length bound L, let P_L denote the set of all s–t paths whose total length does not exceed L, i.e.,

$$P_L = \left\{ p : \sum_{a \in p} l_a \leq L \right\}.$$

A length-bounded cut (lb-cut) is defined as a subset of arcs $C \subseteq A$ whose removal disconnects all s–t paths in P_L. Let $\mathcal{C}_L$ denote the set of all length-bounded cuts, i.e.,

$$\mathcal{C}_L = \{C \subseteq A : C \cap p \neq \emptyset, \forall p \in P_L\}.$$

The objective of the MWMC-LBC problem is to find a minimum-weight minimum-cardinality lb-cut. Formally, for a given length bound L, the objective is to determine an lb-cut C^* with

$$C^* \in \arg\operatorname{lexmin}_{C \in \mathcal{C}_L} \left(\sum_{a \in C} w_a, |C| \right). \tag{1}$$

PROPOSITION 1 **(Computational complexity)**. *The MWMC-LBC problem is NP-hard.*

Proof. The minimum-weight length-bounded cut problem is a special case of the MWMC-LBC problem in which the secondary cardinality objective is ignored. More precisely, given any instance of the minimum-weight lb-cut problem, solving the corresponding MWMC-LBC problem on the same graph returns an lb-cut whose first objective value is the minimum possible cut weight. Therefore, a polynomial-time algorithm for MWMC-LBC would imply a polynomial-time algorithm for the minimum-weight lb-cut problem. Since the minimum-weight lb-cut problem is NP-hard for general length bounds (Baier et al. 2010), MWMC-LBC is also NP-hard. □

3.1. An Example

To provide intuition for the MWMC-LBC problem, Fig. 1 presents a representative example that may arise during the LP relaxation of a branch-and-cut algorithm. Consider the fractional solution as shown in the figure. For the graph associated with this OD pair, the length bound is $L = 7.5$. Note that all arcs have zero weights. We present three lb-cuts with the same total weight of 0. The first cut, denoted by $C_1 = \{a_3, a_4, a_5, a_6\}$, disconnects all o_k-d_k paths. Consequently, the shortest path length becomes infinite. The second cut, $C_2 = \{a_4, a_5, a_6\}$, is also a valid lb-cut. After removing these arcs, the shortest remaining path is $o_k \to 5 \to d_k$ with length 10, which exceeds the prescribed length bound. Similarly, the third cut, $C_3 = \{a_4, a_6\}$, is a valid lb-cut because the shortest remaining path, $o_k \to 1 \to 4 \to d_k$, has length 8, which also exceeds the length bound. It is important to note that all three lb-cuts have the same total weight, namely 0.0. Therefore, they are all optimal solutions to the classical minimum-weight lb-cut problem. However, their cardinalities differ substantially: $|C_1| = 4$, $|C_2| = 3$, and $|C_3| = 2$. The MWMC-LBC criterion therefore selects C_3 as the best of the three lb-cuts because it contains no redundant arcs and has the smallest cardinality.

This example illustrates that minimizing cut weight alone may lead to multiple optimal solutions with significantly different cardinality. By additionally minimizing cardinality, the MWMC-LBC problem eliminates redundant arcs and generates more compact separation inequalities, which can potentially improve the computational performance of branch-and-cut algorithms.

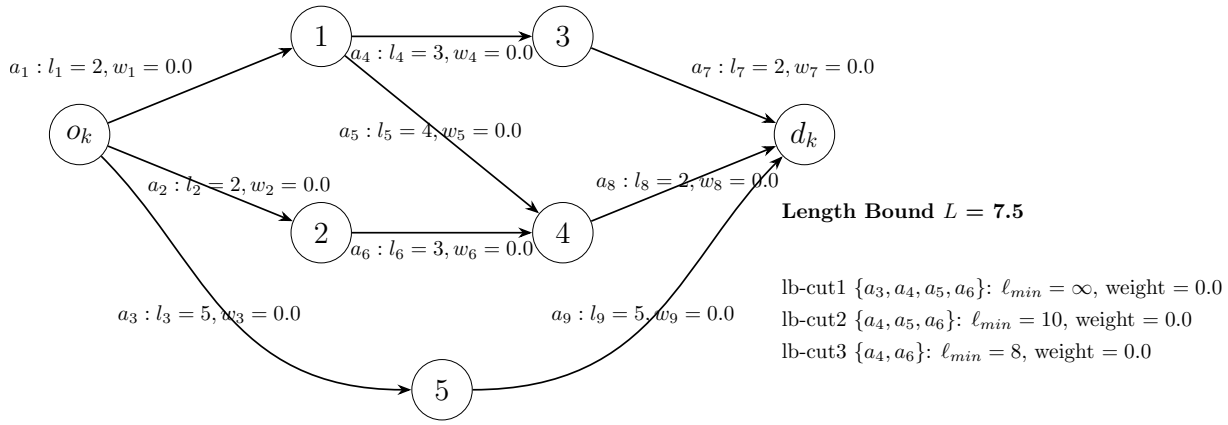

Figure 1 An Example of MWMC-LBC

3.2. Lexicographic Formulation

The MWMC-LBC problem employs a lexicographic objective that prioritizes cut weight before cut cardinality. The proposed MWMC-LBC formulation is referred to as MWMC-LEX. Based on the minimum-weight lb-cut formulation

proposed by Arslan et al. (2019), with modifications to accommodate the MWMC-LBC setting, we introduce the following binary variables: $u_a = 1$ if arc a belongs to the lb-cut, and 0 otherwise. We define the continuous variables π_i as distance potentials associated with the nodes. The formulation of MWMC-LBC is given as follows.

$$\text{[MWMC-LEX]} \quad \min_{\text{lex}} \left(\sum_{a \in A} w_a u_a, \sum_{a \in A} u_a \right) \tag{2}$$

$$\text{s.t. } \pi_t = 0 \tag{3}$$

$$\pi_i \leq \pi_j + l_a + M u_a \qquad \forall a = (i,j) \in A \tag{4}$$

$$\pi_s \geq L + \theta \tag{5}$$

$$\pi_i \geq 0 \qquad \forall i \in N \tag{6}$$

$$u_a \in \{0,1\} \qquad \forall a \in A, \tag{7}$$

where θ is a sufficiently small positive value and M is a sufficiently large constant. The objective function (2) first minimizes the total weight of the lb-cut and then its cardinality. Constraint (3) fixes the distance potential of the destination node. Constraints (4) are the shortest-path propagation constraints. When arc a is not selected in the cut ($u_a = 0$), they reduce to the standard Bellman inequalities. When $u_a = 1$, the corresponding constraint is relaxed by the big-M term. Constraint (5) ensures that every remaining s–t path has length strictly greater than L. This strict inequality is necessary because an lb-cut must eliminate all s–t paths of length at most L, including those of length exactly L. Since strict inequalities cannot be imposed directly in a mixed-integer linear formulation, we replace the condition that the shortest remaining path length is greater than L by Constraint (5), as justified in Proposition 2. Constraints (6)–(7) define the domains of the variables.

PROPOSITION 2. *Since the arc lengths and the length bound are represented with finite precision, there exists a common scaling factor D such that $Dl_a \in \mathbb{Z}$ for all $a \in A$ and $DL \in \mathbb{Z}$. Then, setting*

$$\theta = \frac{1}{D}$$

ensures that the shortest remaining s–t path has length strictly greater than L if and only if its length is at least $L + \theta$. Consequently, Constraint (5) correctly enforces the length-bounded-cut requirement. Moreover, setting $M = L + \theta$ is sufficient for the formulation.

Proof. Since $Dl_a \in \mathbb{Z}$ for all $a \in A$ and $DL \in \mathbb{Z}$, the length of any finite s–t path is an integer multiple of $1/D$. Therefore, any finite s–t path whose length is strictly greater than L must have length at least

$$L + \frac{1}{D} = L + \theta.$$

The converse follows immediately since $\theta > 0$. The equivalence also holds when the remaining graph contains no s–t path, in which case its shortest-path length is interpreted as $+\infty$. Without loss of generality, the distance-potential variables can be bounded above by $L + \theta$, because values exceeding this threshold are unnecessary for enforcing Constraint (5). Therefore, setting $M = L + \theta$ is sufficient for the model. □

To achieve the lexicographic objective (2), we use a two-stage solution method. We first use a formulation to obtain the optimal value of the minimum-weight lb-cut (MW-LBC) as follows.

$$\text{[MW-LBC]} \quad \min \sum_{a \in A} w_a u_a \tag{8}$$

$$\text{s.t. } (3) - (7).$$

Then, let Z^* denote the optimal value obtained from the first-stage model. Given Z^*, the second stage seeks the lb-cut among all minimum-weight solutions. Specifically, the total cut weight is fixed at its optimal value Z^*, and the cut cardinality is subsequently minimized. The resulting minimum-cardinality formulation (MC-LBC) is given below.

$$\text{[MC-LBC]} \quad \min \sum_{a \in A} u_a \tag{9}$$

$$\text{s.t.} \sum_{a \in A} w_a u_a \leq Z^* \tag{10}$$

$$(3) - (7).$$

3.3. Weighted-Objective Formulation

The MWMC-LBC problem can also be solved through a weighted-objective formulation. This idea originates from Arslan et al. (2020), who assign a small positive value $\epsilon > 0$ only to zero-weight arcs to encourage the removal of redundant arcs. However, this approach does not guarantee equivalence to the MWMC-LBC problem. For example, consider two lb-cuts, C_1 and C_2, with arc weights $(0, 1)$ and $(0.3, 0.3, 0.4)$, respectively. Both lb-cuts have the same total weight of 1. Under the lexicographic objective of the MWMC-LBC problem, C_1 is preferred because it has a smaller cardinality ($|C_1| = 2$ versus $|C_2| = 3$). However, if ϵ is added only to zero-weight arcs, the weighted objective values become $1 + \epsilon$ for C_1 and 1 for C_2. Consequently, C_2 is selected for any $\epsilon > 0$, contradicting the lexicographic optimum. Therefore, adding a small value to only zero-weight arcs cannot, in general, reproduce the lexicographic objective of the MWMC-LBC problem, regardless of how small the value is.

To achieve such equivalence, a small positive value $\epsilon > 0$ must be added to the weight of every arc. The modified arc weights are defined as $\tilde{w}_a = w_a + \epsilon$ $(\forall a \in A)$. The resulting formulation, denoted by [MWMC-ϵ], is given as follows.

$$\text{[MWMC-}\epsilon\text{]} \quad \min \sum_{a \in A} \tilde{w}_a u_a \tag{11}$$

$$\text{s.t. } (3) - (7).$$

Since each arc contributes an extra penalty of ϵ, the formulation implicitly favors cuts with smaller cardinality among cuts having the same original weight. We have the following proposition to guarantee the equivalence to the MWMC-LBC problem.

PROPOSITION 3. *If $\epsilon < \frac{\Delta_w}{|A|}$, where Δ_w is the minimum positive difference between distinct lb-cut weights, then the lexicographic formulation and the weighted-objective formulation are equivalent.*

Proof. For any two lb-cuts with different original weights, their weight difference is at least Δ_w. Since any cut contains at most $|A|$ arcs, the total perturbation introduced by ϵ is at most $|A|\epsilon$. If $\epsilon < \Delta_w/|A|$, this perturbation cannot change the ordering of cuts with different original weights. Therefore, model [MWMC-ϵ] first preserves the minimum-weight criterion. Among cuts with the same original weight, the perturbation penalizes each selected arc by ϵ, and thus minimizes cut cardinality. Hence, the model returns an optimal MWMC-LBC solution. □

Proposition 3 is primarily a theoretical equivalence result rather than a practical rule for selecting ϵ. In general, Δ_w is unknown in practice. Our preliminary experiments indicate that the computational performance of [MWMC-ϵ] is sensitive to the choice of ϵ. These observations motivate the two-stage approaches developed in the following sections, which avoid the need to estimate Δ_w.

3.4. Two-Stage Heuristic

The lexicographic method requires solving two mixed-integer programs, which can be computationally expensive when hundreds or thousands of lb-cuts need to be generated within a branch-and-cut algorithm. Therefore, we develop a two-stage heuristic procedure inspired by the cardinality-reduction strategy of Arslan et al. (2020), referred to as MWMC-HEURISTIC.

In the first stage, model [MW-LBC] is solved to obtain a minimum-weight lb-cut. Let $C^{MW} = \{a \in A : u_a^* = 1\}$, where u^* denotes an optimal solution of model [MW-LBC]. In the second stage, we restrict the search to subsets of C^{MW} and solve the following formulation:

$$\text{[MWMC-HEURISTIC]} \quad \min \quad \sum_{a \in C^{MW}} u_a \tag{12}$$

$$\text{s.t.} \quad (3) - (7),$$

$$u_a = 0 \qquad \forall a \in A \setminus C^{MW}. \tag{13}$$

The second-stage problem finds a minimum-cardinality lb-cut among all lb-cuts contained in C^{MW}. In particular, since C^{MW} itself is a feasible lb-cut, the second-stage problem is always feasible and exactly solves

$$\min\{|C| : C \subseteq C^{MW},\ C \text{ is an lb-cut}\}.$$

Hence, the second stage removes as many redundant arcs as possible from the minimum-weight lb-cut obtained in the first stage while preserving the lb-cut property.

The procedure remains heuristic with respect to the MWMC-LBC problem because it restricts the second-stage search to subsets of a single minimum-weight lb-cut C^{MW}. A minimum-cardinality cut among all minimum-weight lb-cuts need not be contained in this particular C^{MW}. Therefore, MWMC-HEURISTIC does not guarantee a globally minimum-cardinality solution among all minimum-weight lb-cuts. Nevertheless, as demonstrated in the computational experiments, this restricted search produces compact lb-cuts while requiring substantially less computational effort than the exact lexicographic formulation.

3.5. An Exact Branching Algorithm for MW-LBC

We further develop an exact Shortest-Path Branching Algorithm (SPBA) to solve the MW-LBC problem. The algorithm is based on a simple observation: if the current residual graph contains an s–t path with length at most L, then any feasible lb-cut must remove at least one arc from this path. Therefore, the algorithm repeatedly identifies the shortest s–t path in the residual graph and branches on the arcs of this path.

Before the branching process starts, we adopt the idea of Arslan et al. (2020) that all zero-weight arcs are removed from the graph. Since removing zero-weight arcs does not increase the cut weight, this preprocessing step is always valid for the minimum-weight objective. If the shortest s–t path in the resulting graph already exceeds L, then a zero-weight lb-cut is obtained. Otherwise, the algorithm initializes a list of partial cuts. For each partial cut, a shortest path in the graph induced by the arcs with positive weights is computed. If its length exceeds L, the partial cut is feasible, and the incumbent solution is updated. Otherwise, the algorithm creates one branch for each arc on this path. The algorithm is guaranteed to terminate and be exact because each branching operation permanently adds one arc to the current cut and removes one arc from the current graph, and the number of arcs in the graph is finite. Consequently, the search tree is finite, and the algorithm eventually converges to an optimal MW-LBC solution.

Algorithm 1 summarizes the SPBA. The algorithm maintains a set $\mathcal{Q}$ of unexplored search nodes. Each search node is represented by a triple (C, G_C, p_C), where C denotes the current partial lb-cut, $G_C = G \setminus C$ is the corresponding residual graph obtained by removing the arcs in C, and p_C is the shortest s–t path in G_C. Starting from the empty cut,

the algorithm repeatedly selects one search node from $\mathcal{Q}$. If the current shortest path satisfies $\ell(p_C) \leq L$, the algorithm branches by creating one child node for each arc on p_C. Specifically, each child adds one arc from p_C to the current partial cut, producing a new residual graph and a new shortest path. If the shortest path in the new residual graph has length greater than L, the corresponding partial cut is a feasible lb-cut and is used to update the incumbent solution. Otherwise, the new search node is inserted into $\mathcal{Q}$ for further exploration. To improve efficiency, SPBA incorporates an incumbent-based pruning strategy. Whenever a better feasible lb-cut is found, the incumbent value Z^* is updated. Any search node whose accumulated cut weight is no smaller than Z^* is immediately discarded, since none of its descendants can produce a better solution. Although the worst-case complexity remains exponential, this pruning strategy substantially reduces the search tree in practice, as confirmed by the computational experiments.

Algorithm 1 Shortest-Path Branching Algorithm (SPBA)

Input: Graph $G = (N, A)$, source s, destination t, length bound L, arc weights w_a
Output: A minimum-weight lb-cut C^*, and the corresponding minimum-weight Z^*
Let $X^0 = \{a \in A : w_a = 0\}$, remove the set X^0 of all zero-weight arcs from G: $G \leftarrow G - X^0$
Compute the shortest s–t path p in the residual graph G
if $\ell(p) > L$ **then**
    Return the zero-weight lb-cut X^0 and the minimum weight $Z^* = 0$
end if
Initialize $\mathcal{Q} \leftarrow \{(\emptyset, G, p)\}$ and $Z^* \leftarrow +\infty$
while $\mathcal{Q} \neq \emptyset$ **do**
    Select and remove a partial solution (C, G_C, p_C) from $\mathcal{Q}$
    for each arc $a \in p_C$ **do**
        $C' \leftarrow C \cup \{a\}$, $G' \leftarrow G_C - \{a\}$
        Compute the shortest s–t path p' in G'
        if $w(C') < Z^*$ **then**
            if $\ell(p') > L$ **then**
                $C^* \leftarrow C'$, $Z^* \leftarrow w(C')$
            else
                Add (C', G', p') to $\mathcal{Q}$
            end if
        end if
    end for
    Remove from $\mathcal{Q}$ all partial cuts with weight at least Z^*
end while
$C^* \leftarrow C^* \cup X^0$
Return C^* and Z^*

3.6. Polynomial-Time Heuristics

To further improve computational efficiency, we develop three solver-independent polynomial-time heuristics for generating lb-cuts. The key observation underlying these methods is that every classical s–t cut is also a valid lb-cut. Let $\mathcal{C}_{st}$ denote the set of classical s–t cuts in $G = (N, A)$. For any $C \in \mathcal{C}_{st}$, removing the arcs in C disconnects s from t and therefore eliminates, in particular, all s–t paths whose length is at most L. Hence, C is a valid lb-cut.

Based on this observation, we consider three simple procedures. Let $\delta^+(s) = \{(s, j) \in A\}$ denote the set of outgoing arcs of the source node s, and let $\delta^-(t) = \{(i, t) \in A\}$ denote the set of incoming arcs of the destination node t. The first procedure, denoted by S-CUT, returns

$$C^s = \delta^+(s),$$

whereas the second procedure, denoted by T-CUT, returns

$$C^t = \delta^-(t).$$

Both C^s and C^t disconnect s from t and are therefore valid lb-cuts. These cuts can be obtained directly from the adjacency structure of the graph without solving an optimization problem. The third procedure, denoted by MIN-CUT, seeks a classical s–t cut with minimum cardinality:

$$C^{\min} \in \arg\min_{C \in \mathcal{C}_{st}} |C|.$$

It can be computed by assigning unit capacity to every arc and solving a classical maximum-flow/minimum-cut problem. Since every arc has unit capacity, the capacity of a cut equals its cardinality, and the resulting minimum cut therefore contains the minimum possible number of arcs among all classical s–t cuts.

PROPOSITION 4. *The* S-CUT, T-CUT, *and* MIN-CUT *all return valid lb-cuts and can be executed in polynomial time.*

Proof. The sets C^s and C^t contain, respectively, all arcs leaving s and all arcs entering t. Removing either set therefore disconnects s from t, which implies that both are valid lb-cuts. They can be constructed by scanning the outgoing arcs of s and the incoming arcs of t, respectively, and thus require at most $O(|A|)$ time. For MIN-CUT, the returned set $C^{\min}$ is a classical s–t cut and is therefore also a valid lb-cut. It is obtained by solving a maximum-flow/minimum-cut problem with unit arc capacities, which is polynomially solvable. Hence, all three procedures run in polynomial time. □

These procedures provide inexpensive alternatives for generating valid lb-cuts without solving the MWMC-LBC problem. In particular, S-CUT and T-CUT require only simple graph operations, whereas MIN-CUT requires a single maximum-flow/minimum-cut computation. These cuts are primarily used as pre-generated structural inequalities at negligible computational cost.

3.7. Summary of the Methods

Table 2 summarizes the solution approaches proposed in this paper. Among them, [MWMC-LEX] is the only method that guarantees optimality for the MWMC-LBC problem. [MWMC-HEURISTIC] preserves the optimal minimum-weight objective but does not guarantee minimum cardinality. [MWMC-ϵ] provides a single-stage alternative whose optimality for the MWMC-LBC problem is guaranteed when the condition in Proposition 3 is satisfied, whereas [MIN-CUT], [S-CUT], and [T-CUT] are polynomial-time heuristic approaches that do not guarantee MWMC optimality. Finally, [SPBA] is an exact algorithm for the MW-LBC problem.

In the application sections, these methods are used differently depending on the structure of the underlying formulation. For the NDSR, inexpensive structural lb-cuts are pre-generated and explicitly added to the path-based formulation. For the RSLP, the proposed methods are combined within a branch-and-cut framework through pre-generated cuts, fast cut compression, heuristic separation, and time-limited minimum-weight separation.

Table 2 Summary of the MWMC-LBC solution approaches

Method	Description	Computation	MW-OPT	MWMC-OPT
MWMC-LEX	Exact two-stage formulation	Two MIP solves	Yes	Yes
MWMC-ϵ	One-stage ϵ formulation	One MIP solve	Conditional	Conditional
MWMC-HEURISTIC	Two-stage heuristic	Two MIP solves	Yes	No
MIN-CUT,S-CUT,T-CUT	Polynomial-time	Polynomial	No	No
SPBA	Shortest-path branching algorithm	Exponential (worst case)	Yes	No

4. Pre-Generated lb-Cuts for the NDSR

The Network Design with Service Requirements (NDSR) is a fundamental network optimization problem in which each commodity must be routed through a single path satisfying prescribed end-to-end service requirements, such as hop, distance, or latency limits. These service requirements make the problem substantially more challenging than the

classical multicommodity network design problem, because feasibility is determined not only by network connectivity but also by the existence of a path satisfying a resource or length bound.

The two main algorithmic streams for the NDSR are the arc-flow polyhedral approach of Balakrishnan et al. (2017), which strengthens the formulation through valid inequalities and separation procedures, and the path-based branch-and-cut-and-price approach of Gudapati et al. (2022), which exploits service-feasible path variables. Although lb-cut inequalities are implicitly enforced by the linking and path-selection constraints of the path-based formulation, explicitly pre-generating and adding a small number of these inequalities can still provide meaningful computational benefits. As demonstrated by our computational experiments, their explicit inclusion improves the overall computational efficiency of the path-based NDSR formulation.

4.1. Problem Definition of the NDSR

Let $G = (N, A)$ denote a directed network, where N is the set of nodes and A is the set of arcs. Let K denote the set of commodities, and let $\mathcal{M}$ denote the set of service metrics. For each commodity $k \in K$, let o_k and d_k denote its origin and destination, respectively. Each arc $a \in A$ is associated with an activation cost F_a, a commodity-dependent usage cost c_a^k, and a metric-dependent weight w_a^{km} for commodity $k \in K$ and metric $m \in \mathcal{M}$. The NDSR requires that each commodity k be routed on a feasible path from o_k to d_k. A path p is feasible for commodity k if, for each service metric $m \in \mathcal{M}$, the total weight of the arcs on the path does not exceed the corresponding limit W^{km}, i.e., $\sum_{a \in p} w_a^{km} \leq W^{km}$. The objective of the NDSR is to determine a set of activated arcs and a feasible path for each commodity such that all selected paths use only activated arcs and the total cost is minimized. The total cost consists of the activation costs of the selected arcs and the routing usage costs incurred by all commodities.

4.2. Path-Based Formulation for the NDSR

To the best of our knowledge, the path-based algorithm of Gudapati et al. (2022) is currently the state-of-the-art exact approach for the NDSR. Their modeling choice eliminates the need to explicitly impose service constraints, since each path variable already satisfies all service requirements by construction. Because the number of feasible paths can be exponential, Gudapati et al. (2022) first employ a labeling algorithm to enumerate all feasible paths when full enumeration is computationally viable. When enumeration becomes impractical, they embed a column generation procedure within a branch-and-cut-and-price (BCP) framework. Their computational results on the 90 benchmark instances introduced by Balakrishnan et al. (2017) show that the path-based model with explicit path enumeration generally outperforms the BCP approach, solving 80 instances to proven optimality. It significantly scales up the size of solvable NDSR instances compared with the arc-flow-based approach of Balakrishnan et al. (2017), and provides the strongest benchmark currently available for this problem. The BCP framework becomes advantageous mainly when the service limits W^{km} are large, in which case the number of feasible paths grows substantially and full enumeration becomes less effective.

Since the purpose of this paper is not to develop a new exact solution framework for the NDSR from scratch, but rather to investigate whether explicitly incorporating pre-generated lb-cuts can improve the computational performance of the current state-of-the-art path-based algorithm, we primarily build on the formulation of Gudapati et al. (2022) as follows.

Let $\mathcal{P}_k$ denote the set of feasible paths for commodity $k \in K$. For each path $p \in \mathcal{P}_k$, we introduce a binary variable α_p, which equals 1 if commodity k is routed along path p, and 0 otherwise. Let parameter c_p denote the routing cost of path p. For each arc $a \in A$, we introduce a binary variable z_a, which equals 1 if arc a is activated, and 0 otherwise. Using these variables, the resulting path-based formulation is given as follows.

$$\min \quad \sum_{a \in A} F_a z_a + \sum_{k \in K} \sum_{p \in P_k} c_p \alpha_p \tag{14}$$

$$\text{s.t.} \quad z_a \geq \sum_{p \in P_k : a \in p} \alpha_p \qquad \forall a \in A, k \in K \tag{15}$$

$$\sum_{p \in P_k} \alpha_p = 1 \qquad \forall k \in K \tag{16}$$

$$z_a \in \{0,1\}, \alpha_p \in \{0,1\} \qquad \forall k \in K, p \in P_k, a \in A. \tag{17}$$

The objective function (14) minimizes the total cost. Constraints (15) are arc-activation linking constraints. They ensure that if a path selected for commodity k uses arc a, then arc a must be activated. Constraints (16) require that exactly one feasible path be selected for each commodity. Finally, constraints (17) define the domains of the decision variables.

4.3. Pre-Generated lb-Cuts for the Path-Based NDSR Model

We introduce lb-cuts into the path-based NDSR formulation as a class of valid inequalities. For a commodity $k \in K$, a path is feasible only if it satisfies all service requirements simultaneously. Therefore, we define an lb-cut for commodity k as an arc set whose removal eliminates all o_k–d_k paths satisfying all service metrics simultaneously. In other words, after removing an lb-cut C, there exists no remaining path from o_k to d_k whose accumulated weight satisfies $\sum_{a \in p} w_a^{km} \leq W^{km}$ for every metric $m \in M$. Since every service-feasible path must contain at least one arc in C, any feasible NDSR solution must activate at least one arc in C.

Let Γ_k denote the set of all lb-cuts for commodity k. The corresponding lb-cut inequalities are

$$\sum_{a \in C} z_a \geq 1 \qquad \forall k \in K, C \in \Gamma_k. \tag{18}$$

It is important to note that inequalities (18) are already implied by the linking and path-selection constraints of the path-based formulation. Indeed, for any $k \in K$ and $C \in \Gamma_k$,

$$\begin{aligned} \sum_{a \in C} z_a &\geq \sum_{a \in C} \sum_{p \in P_k : a \in p} \alpha_p \\ &= \sum_{p \in P_k} |p \cap C| \alpha_p \\ &\geq \sum_{p \in P_k} \alpha_p = 1, \end{aligned}$$

where the second inequality follows from $|p \cap C| \geq 1$ for every service-feasible path $p \in P_k$. Therefore, inequalities (18) do not tighten the LP relaxation of the path-based formulation.

Nevertheless, explicitly adding a small number of these inequalities can still provide a meaningful computational benefit. Rather than relying on dynamic separation, we pre-generate a small set of lb-cuts for each commodity and add them to the initial formulation. These cuts can be generated at negligible computational cost, and our computational experiments show that their inclusion improves the overall computational performance of the path-based NDSR formulation, including a reduction in solution time and a modest improvement in the final optimality gaps of the unsolved instances.

4.3.1. Transformed Network

For each commodity k and metric m, let γ_{ij}^{km} denote the minimum accumulated value of metric m from node i to node j. We adopt the transformation of MirHassani and Ebrazi (2013) and construct a transformed network $G_k = (N_k, A_k)$ for each k, where

$$A_k = \{a = (i,j) \in A : \gamma_{o_k i}^{km} + w_a^{km} + \gamma_{j d_k}^{km} \leq W^{km},\ \forall m \in \mathcal{M}\}, \tag{19}$$

$$N_k = \{i \in N : \exists j \in N \text{ such that } (i,j) \in A_k \text{ or } (j,i) \in A_k\}. \tag{20}$$

The arc set A_k contains all arcs that may appear on a service-feasible path for commodity k. Indeed, if an arc $a = (i,j)$ belongs to a service-feasible o_k-d_k path, then the accumulated value of each metric along the corresponding prefix, arc a, and suffix must not exceed W^{km}. Hence, the above condition is satisfied for every metric $m \in \mathcal{M}$.

4.3.2. Pre-Generated lb-Cuts. We pre-generate three classes of lb-cuts for each commodity, namely S-CUT, T-CUT, and MIN-CUT, and add them to the initial formulation. Based on the transformed network $G_k = (N_k, A_k)$, the S-CUT and T-CUT are defined as $C_k^s = \{a = (o_k, j) \in A_k\}$ and $C_k^t = \{a = (i, d_k) \in A_k\}$. The MIN-CUT is obtained by assigning unit capacity to each arc in A_k and solving a classical maximum-flow/minimum-cut problem on the network $G_k = (N_k, A_k)$. Let the resulting minimum o_k-d_k cut be denoted by C_k^m. We therefore have the following valid lb-cut inequalities:

$$\sum_{a \in C} z_a \geq 1 \qquad \forall k \in K, C \in \{C_k^m, C_k^s, C_k^t\}. \tag{21}$$

These three classes of cuts are pre-generated before the branch-and-cut procedure starts. They are inexpensive to compute and can be explicitly added to the initial formulation at negligible computational cost.

5. MWMC-LBC Separation for the RSLP

The Refueling Station Location Problem (RSLP) is a classical transportation network design problem that seeks to determine the locations of refueling stations so that origin–destination demands can be served under vehicle range and length bound constraints. Since its introduction by Kuby and Lim (2005), the problem has received considerable attention, with several exact formulations and algorithms subsequently proposed (Yıldız et al. 2016, Arslan et al. 2019). In particular, Arslan et al. (2019) developed a branch-and-cut algorithm in which violated length-bounded cuts are dynamically separated to enforce route-feasibility requirements. Because a large number of separation problems must be solved throughout the optimization process, the efficiency of the overall algorithm depends heavily on the generated cuts. Motivated by this observation, we investigate how the proposed MWMC-LBC method can strengthen the separation routine in the branch-and-cut algorithm of Arslan et al. (2019).

5.1. Problem Definition of RSLP

We continue to use the notation $G = (N, A)$ and K introduced in Section 4, but reinterpret them in the context of the RSLP. Here, N is the set of candidate refueling locations and A is the set of road segments. Let K denote the set of OD demands. For each demand $k \in K$, the origin node, destination node, traffic flow, vehicle range, and maximum travel distance that drivers are willing to tolerate are denoted by o_k, d_k, f_k, r_k and λ_k, respectively. A demand k is considered covered if there exists a feasible route connecting o_k and d_k such that the distance traveled between any two consecutive refueling stations does not exceed r_k, and the total route length does not exceed λ_k. We adopt the round-trip setting from the existing literature (Yıldız et al. 2016, Arslan et al. 2019, Kınay et al. 2021): (1) a vehicle departs from its origin with half of its fuel capacity, unless a refueling station is located at the origin, in which case it can depart with a full tank; and (2) upon reaching its destination, the vehicle must retain at least half of its fuel capacity unless a refueling station is located at the destination, in which case it can arrive with an empty tank. The objective of the RSLP is to maximize the total covered flow, given a limit of at most p open refueling stations.

5.2. Transformed Network

To satisfy the round-trip setting, we adapt the network transformation of MirHassani and Ebrazi (2013) and construct a transformed network as follows. Let δ_{ij} denote the shortest-path distance between nodes $i, j \in N$. We construct a transformed network $G_k = (N_k, A_k)$ for each demand k, where $N_k = \{s_k, t_k\} \cup \{i \in N : \delta_{o_k,i} + \delta_{i,d_k} \leq \lambda_k\}$. Here, s_k and t_k are dummy nodes representing the origin o_k and the destination d_k, respectively. The arc set $A_k = A_k^1 \cup A_k^2 \cup A_k^3$ is defined as follows:

$$A_k^1 = \left\{(s_k, j) : \delta_{o_k j} \leq r_k/2, j \in N_k \setminus \{s_k, t_k\}\right\},$$
$$A_k^2 = \left\{(i, t_k) : \delta_{i d_k} \leq r_k/2, i \in N_k \setminus \{s_k, t_k\}\right\},$$
$$A_k^3 = \left\{(i, j) : \delta_{ij} \leq r_k, i, j \in N_k \setminus \{s_k, t_k\}, i \neq j\right\}.$$

The arc set A_k is partitioned into three subsets. Arcs in A_k^1 and A_k^2 represent the refueling logic for vehicles departing from the origin and arriving at the destination, respectively, whereas arcs in A_k^3 capture the refueling logic at intermediate locations along the route. The length of each transformed arc is defined by the corresponding shortest-path distance in the original network, i.e., $\ell^k_{s_kj} = \delta_{o_kj}$ for $(s_k, j) \in A_k^1$, $\ell^k_{it_k} = \delta_{id_k}$ for $(i, t_k) \in A_k^2$, and $\ell^k_{ij} = \delta_{ij}$ for $(i, j) \in A_k^3$.

5.3. Branch-and-Cut Algorithm of RSLP

Arslan et al. (2019) proposed a branch-and-cut algorithm for the RSLP. Let x_i be the binary decision variable that equals 1 if a refueling station is installed at node i, and 0 otherwise. Let y_k be the binary decision variable that equals 1 if OD pair k is covered, and 0 otherwise. For each OD pair k, let Γ_k denote the set of all lb-cuts associated with that demand. The formulation is as follows.

$$\max \quad \sum_{k \in K} f_k y_k \tag{22}$$

$$\text{s.t.} \quad y_k \leq \sum_{i \in S} x_i \qquad \forall k \in K, S \in \Gamma_k \tag{23}$$

$$\sum_{i \in N} x_i \leq p \tag{24}$$

$$x_i \in \{0,1\}, y_k \in \{0,1\} \qquad \forall k \in K, i \in N. \tag{25}$$

The objective function (22) maximizes the total covered flow. Constraints (23) enforce that a demand k cannot be covered if none of the nodes in its lb-cuts are opened. Constraint (24) limits the number of open stations to at most p. Note that the constraints (23) are exponential in number and cannot be enumerated explicitly. Therefore, the key component of this approach is the separation of violated lb-cuts (23), which are iteratively added to the model. To solve the separation problem associated with constraints (23), Arslan et al. (2019) proposed different procedures for fractional and integer solutions. For an integer solution of each OD pair, they verify the existence of a feasible path by solving a shortest-path problem. If no feasible path exists, a violated lb-cut is identified and subsequently strengthened through a cut-minimization procedure. For a fractional solution of each OD pair, they formulate the separation problem as an MW-LBC problem and solve it exactly using a mixed-integer programming formulation. To further reduce computational effort, they also proposed an ϵ-MIN heuristic for generating violated cuts.

5.4. MWMC-LBC-Enhanced Branch-and-Cut Algorithm for the RSLP

Building on the B&C algorithm of Arslan et al. (2019), we introduce three MWMC-LBC-based enhancements to improve the separation of lb-cuts. These enhancements are designed to strengthen the initial formulation, generate more compact violated cuts during the search, and improve the efficiency of separation at integer and fractional solutions.

5.4.1. Pre-Generated lb-Cuts. Before the branch-and-cut procedure starts, we use the polynomial-time heuristics in Section 3.6 to pre-generate a small set of valid lb-cuts for each OD pair. These cuts are derived only from the network structure and added to the initial formulation. Their purpose is to strengthen the initial relaxation at negligible computational cost and reduce the burden of dynamic separation.

For each OD pair $k \in K$, we pre-generate two reachability-based cuts, S-CUT and T-CUT, and one MIN-CUT. Since the RSLP separation problem considered by Arslan et al. (2019) is node-based, we explicitly define the two reachability-based cuts as sets of station nodes:

$$S_k^s = \{j \in N_k \setminus \{s_k, t_k\} : (s_k, j) \in A_k\}, \tag{26}$$

$$S_k^t = \{i \in N_k \setminus \{s_k, t_k\} : (i, t_k) \in A_k\}. \tag{27}$$

Thus, S_k^s contains the station nodes directly reachable from the origin s_k, whereas S_k^t contains the station nodes from which the destination t_k can be reached directly. These sets correspond to the S-CUT and T-CUT, respectively.

To generate the MIN-CUT, we use a node-splitting transformation. Each station node $i \in N_k \setminus \{s_k, t_k\}$ in G_k is replaced by two nodes i^- and i^+ connected by a dummy arc (i^-, i^+). We assign weight one to every dummy arc and infinite weight to all other arcs. A minimum-weight arc cut in the transformed network therefore corresponds to a minimum-cardinality set of station nodes whose removal disconnects s_k from t_k in the original network.

These three cuts can be pre-generated for every OD pair in negligible time and provide an inexpensive way to strengthen the initial formulation. As shown in the computational experiments, the enhanced algorithm that incorporates these pre-generated cuts, together with the proposed separation strategies, substantially reduces the number of dynamically generated lb-cuts and improves the overall solution time.

5.4.2. Separation at Integer Solutions. Let $(\bar{x}, \bar{y})$ be an integer solution obtained during the branch-and-cut procedure. For each OD pair k with $\bar{y}_k = 1$, we check whether a feasible refueling route exists under the open-station set $N(\bar{x}) = \{i \in N_k \setminus \{s_k, t_k\} : \bar{x}_i = 1\}$ in G_k. If no such route exists, demand k is incorrectly covered, and a violated lb-cut $\bar{S}_k = (N_k \setminus \{s_k, t_k\}) \setminus N(\bar{x}) = \{i \in N_k \setminus \{s_k, t_k\} : \bar{x}_i = 0\}$ must be generated: $y_k \leq \sum_{i \in \bar{S}_k} x_i$. The original algorithm of Arslan et al. (2019) applies a cut-minimization procedure after identifying a violated lb-cut $\bar{S}_k$. This procedure examines the closed station set $\bar{S}_k$ one by one and removes a station whenever the remaining set is still a valid lb-cut in G_k. Although effective, it can be expensive when the initial violated cut contains many closed stations, because each deletion test requires a shortest-path computation.

We therefore propose an adaptive cut-minimization strategy. Suppose that a violated lb-cut $\bar{S}_k$ is obtained at the current integer solution. Let S_k^m be the pre-generated MIN-CUT for OD pair k, and define $\eta_k = |S_k^m|$.

If $|\bar{S}_k| \leq \eta_k$, we directly apply the original cut-minimization procedure to $\bar{S}_k$.

If $|\bar{S}_k| > \eta_k$, we first compress $\bar{S}_k$ using a polynomial-time ϵ-MIN procedure. For each dummy arc associated with station i, we set $w_i = \epsilon$ if $i \in \bar{S}_k$, and $w_i = 1 + \epsilon$ otherwise, with ϵ chosen such that $\epsilon < 1/|N_k|$. The resulting maximum-flow/minimum-cut problem returns a candidate cut $\tilde{S}_k$. If the weight of $\tilde{S}_k$ is at least one, then $\tilde{S}_k$ contains at least one open station and the corresponding inequality is not violated at the current integer solution. In this case, we discard $\tilde{S}_k$ and fall back to the original cut-minimization procedure applied to $\bar{S}_k$. Otherwise, $\tilde{S}_k$ consists only of closed stations and is used as the input to the cut-minimization procedure. Thus, whenever the compression step succeeds, the expensive deletion-based minimization is applied to a substantially smaller cut.

5.4.3. Separation at Fractional Solutions. Separating at fractional solutions is computationally more challenging than at integer solutions. Conceptually, fractional separation involves two tasks: identifying a violated lb-cut with small separation weight and reducing its cardinality while preserving violation. Solving both tasks exactly at every callback would be computationally prohibitive. Therefore, solving the MW-LBC model and the MC-LBC model exactly for every OD pair at every fractional solution would be impractical, especially because the separation routine must be invoked many times during the branch-and-cut process.

For this reason, we adopt a fast heuristic separation strategy at fractional solutions. Instead of solving the MW-LBC model, we first apply the ϵ-MIN procedure, which runs in polynomial time and is computationally very efficient. If ϵ-MIN fails to identify a violated cut, we then apply the SPBA procedure to search for a violated lb-cut. As shown in Section 6.1.1, SPBA can find a minimum-weight lb-cut faster than the MW-LBC model in about 95% of the tested instances. Nevertheless, SPBA still has exponential worst-case complexity. To avoid excessive separation time, we impose a time limit on SPBA and denote this time-limited variant by SPBA-t. Once an lb-cut is obtained, we apply the cut-min procedure to reduce its cardinality substantially, rather than solving the MC-LBC model exactly. We also adopt the quick-exit strategy.

The proposed combination of ϵ-MIN, SPBA-t, cut-min, and quick-exit provides an efficient way to generate compact violated length-bounded cuts at fractional solutions. The individual and combined effects of these components are evaluated computationally in Section 6.2.

5.5. Summary of Algorithmic Enhancements

Table 3 summarizes the main differences between the original branch-and-cut algorithm of Arslan et al. (2019) and our enhanced implementation. The proposed enhancements are threefold. Fig. 2 shows the flowchart of the proposed separation procedure. First, before the branch-and-cut procedure starts, we add three classes of pre-generated lb-cuts, namely S-CUT, T-CUT, and MIN-CUT, to strengthen the initial formulation. Second, during integer-solution separation, we introduce an ϵ-MIN-based compression step before applying the original cut-minimization procedure. This avoids applying deletion-based cut minimization directly to large violated cuts. Third, for fractional solutions, we employ a hybrid separation procedure that combines ϵ-MIN, SPBA-t, cut-min, and quick-exit strategies.

Table 3 Summary of Enhancements to the RSLP Branch-and-Cut Algorithm

Algorithm	Pre-generated lb-cuts	Separation	
		Integer	Fractional
B&C (Arslan et al. 2019)	✗	cut-min	ϵ-MIN, cut-min
Enhanced B&C	S-CUT, T-CUT, MIN-CUT	ϵ-MIN, cut-min	ϵ-MIN, SPBA-t, cut-min, quick-exit

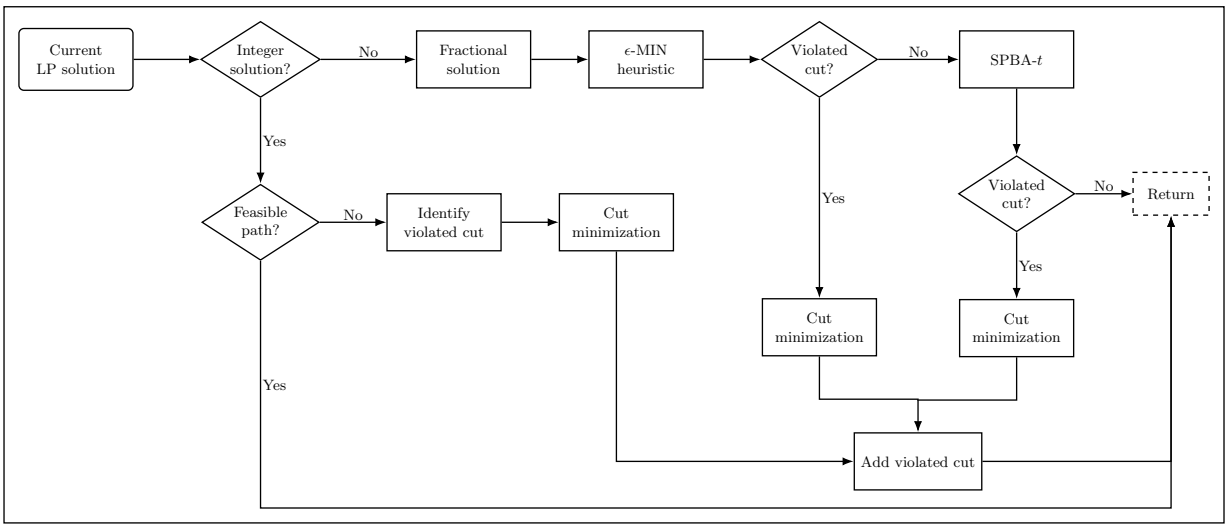

Figure 2 Flowchart of the Proposed Separation Procedure at Integer/Fractional Solutions

6. Computational Experiments

This section evaluates the computational performance of the proposed MWMC-LBC methods and their impact on two network optimization applications. The experiments are organized into three parts. First, we assess the solution methods developed in Section 3 for the MWMC-LBC problem itself. Second, we evaluate the proposed lb-cut enhancements on the RSLP benchmarks. Third, we evaluate the computational impact of pre-generated lb-cuts on the NDSR benchmarks. All computational experiments are conducted on the Narval cluster of the Digital Research Alliance of Canada (Compute Canada). The experiments are run on compute nodes equipped with AMD EPYC 7532 processors using Gurobi Optimizer 12.0. Unless otherwise specified, Gurobi is run with its default thread setting. All proposed

algorithms are implemented in Java. The source code used for the RSLP and NDSR experiments is publicly available in our GitHub repository at `https://github.com/wwq-uibe/MWMC`.

6.1. Computational Performance of MWMC-LBC Solution Methods

We first evaluate the computational performance of the MWMC-LBC solution methods developed in Section 3. This experiment focuses on the separation problem itself and compares the proposed methods in terms of cut weight, cut cardinality, and runtime. The goal is to understand the trade-offs between cut quality and computational effort before examining how the proposed methods can be exploited in the RSLP and NDSR applications.

The lb-cut instances used in this experiment are recorded during the exploration of the branch-and-bound trees of the RSLP algorithms, using callback functions. To avoid a test set dominated by easy cases, we only store fractional separation instances. Integer separation instances are not stored because an integer solution can be viewed as a special case of a fractional solution, and for each OD pair or commodity, violation can be checked by solving a single shortest-path or feasibility problem. In addition, we do not store separation instances that can be handled by the polynomial-time ϵ-MIN heuristic. Such cases are, therefore, relatively easy and do not provide a meaningful stress test for the proposed MWMC-LBC methods. We keep only those instances for which ϵ-MIN fails to identify a violated cut, while exact lb-cut separation succeeds. These instances represent the more challenging separation cases where advanced MWMC-LBC methods are expected to be most useful.

Using this callback-based collection procedure, we randomly store 1,500 lb-cut instances. For each instance, we store the graph, source node, destination node, length bound, and arc weights. We then solve these instances using the methods proposed and compare the resulting cut weight, cut cardinality, and computational time.

To avoid numerical issues in the implementation of [MWMC-LEX] and [MWMC-ϵ], all arc weights are scaled by 10^6 and represented as integers when constructing the corresponding weight objectives and constraints. The MIP models are solved with `MIPGap`=0, `MIPGapAbs`=0, `FeasibilityTol`=10^{-9}, and `IntFeasTol`=10^{-9}. We classify a solution as minimum-weight optimal if its scaled original cut weight is equal to the optimal benchmark weight obtained by [MWMC-LEX]. This integer scaling also ensures that the second-stage weight constraint in [MWMC-LEX] is imposed exactly with respect to the six-decimal precision of the input arc weights.

6.1.1. SPBA versus MW-LBC for Minimum-Weight lb-Cut Separation.

We first compare SPBA with the mixed-integer programming formulation [MW-LBC] for the minimum-weight lb-cut problem. Both methods aim to identify a minimum-weight lb-cut, which is the key step for determining whether a violated lb-cut can be generated efficiently for a given OD pair or commodity. Therefore, this comparison focuses on the ability of the two approaches to solve the minimum-weight separation problem.

Fig. 3 shows that SPBA solves approximately 95% of the instances within 0.02 seconds, whereas [MW-LBC] requires more time on the great majority of these instances. This indicates that the shortest-path branching strategy is highly effective for the majority of separation instances encountered in practice.

However, Fig. 3 only displays the running-time distribution between 0 and 0.1 seconds. Over the full set of instances, the most time-consuming instance for SPBA requires 43 seconds, whereas the maximum running time of [MW-LBC] is only 0.6 seconds. This suggests that, although SPBA is significantly faster on most instances, its worst-case behavior can be less stable than that of the MIP-based formulation. Overall, these results suggest that SPBA is a very effective separation tool for quickly identifying minimum-weight lb-cuts in most cases, but it should be used with a time limit when embedded in larger branch-and-cut algorithms. Similar results were also observed in Arslan et al. (2020). This observation motivates the time-limited SPBA variant used in the RSLP experiments.

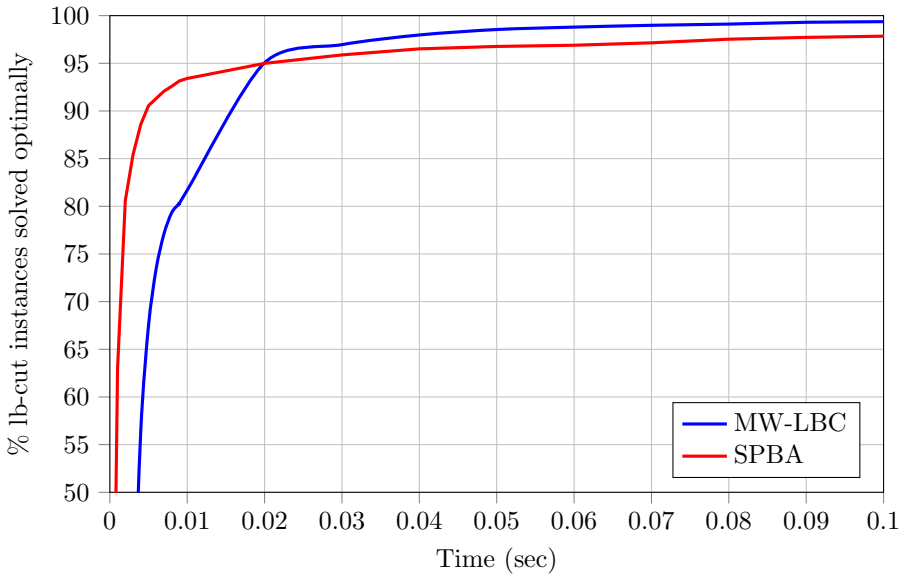

Figure 3 Cumulative percentage of lb-cut instances solved to optimality by SPBA and MW-LBC

6.1.2. Comparison of MWMC-LBC Solution Methods. We next compare the proposed solution methods for the MWMC-LBC problem, including the exact two-stage formulation [MWMC-LEX], the perturbation-based single-stage formulation [MWMC-ϵ], the two-stage heuristic [MWMC-HEURISTIC], and the SPBA-based variants combined with cut minimization. Since only [MWMC-LEX] is designed to solve the lexicographic MWMC-LBC problem exactly, an instance is counted as MWMC-optimal only when both the minimum cut weight and the minimum cardinality among all minimum-weight lb-cuts are attained.

Table 4 reports the computational performance of the MWMC-LBC solution methods on 1,500 lb-cut instances. The results show that [MWMC-LEX] is the most reliable method, as it solves all 1,500 instances to MWMC optimality, obtaining an average cut weight of 0.436776 and an average cardinality of 5.64, which serve as benchmark values for evaluating the remaining methods. However, this high solution quality comes at a non-negligible computational cost: the average runtime is 0.61 seconds, and the maximum runtime reaches 780 seconds. Therefore, although [MWMC-LEX] is appropriate for evaluating solution quality, directly invoking it many times inside a large-scale branch-and-cut algorithm may be computationally expensive.

Table 4 Computational performance of MWMC-LBC solution methods on 1,500 lb-cut instances

	Avg W	Avg C	Avg Time (s)	Max Time (s)	# MWMC-Opt	# MW-Opt
MWMC-LEX	0.436776	5.64	0.61	780.47	1500/1500	1500/1500
MWMC-ϵ (1e-3)	0.436860	5.49	0.10	36.17	1445/1500	1445/1500
MWMC-ϵ (1e-4)	0.436785	5.62	0.07	23.21	1477/1500	1477/1500
MWMC-ϵ (1e-5)	0.436785	5.62	0.06	15.11	1477/1500	1477/1500
MWMC-HEURISTIC	0.436776	6.72	0.07	6.27	958/1500	1500/1500
ϵ-MIN	0.828218	2.67	negligible	negligible	0/1500	0/1500
SPBA	0.436776	170.04	0.09	43.19	0/1500	1500/1500
SPBA-0.01	0.439932	170.00	0.001	0.01	0/1500	1476/1500
SPBA-0.01 + cut-min	0.439932	7.61	0.014	0.37	633/1500	1476/1500

Avg W: average weight of lb-cut; Avg C: average cardinality of lb-cut
#MWMC-Opt: number of min-weight min-cardinality optimal solutions obtained
#MW-Opt: number of min-weight optimal solutions obtained
cut-min: heuristic cardinality reduction via sequential element examination

The formulation [MWMC-ϵ] provides a substantially faster alternative to [MWMC-LEX], but its ability to preserve the primary minimum-weight objective depends on the choice of ϵ. With $\epsilon = 10^{-3}$, the method solves all instances to optimality in an average of 0.10 seconds and obtains MWMC-optimal solutions for 1,445 of the 1,500 instances. Its average cardinality of 5.49 is even smaller than the benchmark value of 5.64. This does not indicate better MWMC solutions; rather, for 55 instances, the perturbation is sufficiently large to sacrifice the primary minimum-weight objective in favor of a smaller cardinality. Reducing ϵ to 10^{-4} substantially improves the preservation of the primary objective: 1,477 instances attain the minimum cut weight and, among these instances, all 1,477 also attain the minimum cardinality among minimum-weight lb-cuts. The average cardinality is 5.62, with an average runtime of only 0.07 seconds. Further reducing ϵ to 10^{-5} yields essentially the same solution quality, again obtaining 1,477 MWMC-optimal solutions with an average cardinality of 5.62, while the average runtime is 0.06 seconds.

These results indicate that sufficiently small values of ϵ perform well empirically on the tested instances. In particular, whenever [MWMC-ϵ] preserves the minimum-weight objective in our experiments, it also obtains the minimum cardinality among minimum-weight lb-cuts. Nevertheless, the appropriate magnitude of ϵ is inherently instance-dependent: to guarantee lexicographic optimality, ϵ must be small enough that the cumulative cardinality perturbation cannot outweigh the minimum positive difference between distinct cut weights. Since this difference is generally unknown a priori, [MWMC-ϵ] does not provide the same parameter-free lexicographic guarantee as [MWMC-LEX], despite its strong empirical performance for $\epsilon = 10^{-4}$ and 10^{-5}.

The two-stage heuristic [MWMC-HEURISTIC] preserves the minimum-weight objective for all 1,500 instances and is substantially faster than [MWMC-LEX], with an average runtime of 0.07 seconds. However, because it minimizes cardinality only over lb-cut subsets of a single minimum-weight cut obtained in the first stage, it does not necessarily identify the globally minimum-cardinality solution among all minimum-weight lb-cuts. Its average cardinality is 6.72, and only 958 instances are solved to MWMC optimality. Thus, [MWMC-HEURISTIC] offers a reasonable compromise between runtime and solution quality, but it remains noticeably weaker than the exact lexicographic formulation in terms of cardinality minimization.

We also report the performance of the polynomial-time ϵ-MIN heuristic. Although ϵ-MIN requires negligible runtime and produces cuts with a small average cardinality of 2.67, its average cut weight is 0.828218, which is almost twice the optimal average cut weight of 0.436776. Moreover, it fails to obtain a minimum-weight cut for all 1,500 instances. This result should not be interpreted as showing that ϵ-MIN is ineffective in general; rather, it reflects the fact that the test set was deliberately constructed from difficult separation instances for which the polynomial-time heuristic alone is insufficient. Therefore, ϵ-MIN is best viewed as a very fast preliminary separation routine for easy cases, rather than as a substitute for MWMC-LBC separation.

The SPBA-based methods offer a more computationally attractive alternative to the MIP-based MWMC formulations for identifying minimum-weight lb-cuts. As shown in Table 4, the exact SPBA method obtains minimum-weight lb-cuts for all 1,500 instances and achieves a substantially lower average runtime than the MIP-based MWMC methods. However, SPBA still has exponential worst-case complexity, and its maximum runtime among the tested lb-cut instances reaches 43 seconds. This makes the direct use of exact SPBA impractical in a branch-and-cut framework, where separation must be performed repeatedly and potentially for many OD pairs or commodities at each callback.

To address this issue, we impose a short time limit on SPBA. Based on the computational evidence in Fig. 3 and our preliminary tests, we set the time limit to $t = 0.01$ seconds and refer to the resulting variant as SPBA-0.01. This time-limited version reduces the average runtime to 0.001 seconds while still obtaining minimum-weight lb-cuts for 1,476 instances. Nevertheless, since SPBA minimizes only the cut weight, it does not control cut cardinality, and the raw cuts are therefore very dense, with an average cardinality of 170.00. This large cardinality is partly attributable to

the preprocessing step in Algorithm 1, which removes all zero-weight arcs before branching and adds them back to the returned minimum-weight cut. Hence, the cardinality of the raw SPBA cuts should not be interpreted as representative of arbitrary minimum-weight lb-cuts. We therefore apply the cut-min procedure after SPBA-0.01 to remove redundant arcs while preserving the lb-cut property. The resulting method, [SPBA-0.01 + cut-min], reduces the average cardinality from 170.00 to 7.61, while increasing the average runtime only to 0.014 seconds. Although this method does not guarantee MWMC optimality, it provides a practical compromise between computational efficiency and cut compactness, making it suitable for high-frequency separation within large-scale branch-and-cut algorithms.

6.2. Computational Performance on the RSLP

We now present the results on the RSLP.

6.2.1. RSLP Benchmark Instances and Experimental Setting. We note that an RSLP formulation also appears in our previous work, Wang et al. (2026), as the master problem of a decomposition algorithm. In that study, the RSLP serves as a relaxation of a bilevel charging-facility location problem to generate candidate location decisions, whereas station capacities, queuing effects, and EV path-selection decisions are handled in a subsequent subproblem. In contrast, the present study considers the RSLP independently as a test bed for systematically evaluating different lb-cut generation strategies within a branch-and-cut framework, with particular emphasis on the proposed minimum-weight minimum-cardinality lb-cuts.

We conducted extensive computational experiments on four large-scale, realistic transportation networks derived from the road networks of Belgium, California, Quebec, and the United States. These networks consist of hundreds of nodes, thousands of arcs, and thousands of OD pairs, and exhibit diverse geographical distributions and network topologies. The BE581 network, taken from Mahmutoğulları and Yaman (2023), contains 581 municipality nodes and 1,822 highway arcs. The 115 municipalities with populations above 25,000 are selected as origins and destinations, yielding 6,555 OD pairs. The CA339 network, from Arslan et al. (2019), contains 339 road-intersection or population-center nodes and 1,234 highway arcs. It includes 1,167 OD pairs generated from 51 population centers with more than 50,000 inhabitants and pairwise distances of at least 30 km. The QC405 network is derived from Hydro-Québec and contains 405 road-junction or population-center nodes and 1,003 road-segment arcs. It includes 2,842 OD pairs generated from 88 population centers with distances exceeding 100 km. The US420 network, taken from Kınay et al. (2023), contains 420 nodes and 1,276 highway arcs. It includes 741 OD pairs generated from 39 cities with populations above 200,000. For all networks, OD flows are computed using a gravity model. Table 5 summarizes the numbers of nodes, arcs, and OD pairs for the four networks, along with the minimum and maximum OD-pair distances in each network. Figures 5 to 8 provide visual representations of the four transportation networks.

Table 5 Characteristics of the transportation networks

Network	#Nodes	#Arcs	#Pop. Centers	#OD Pairs	Min. OD Dis (km)	Max. OD Dis (km)
BE581[a]	581	1822	115	6555	1.70	380.40
CA339[b]	339	1234	51	1167	30.00	463.52
QC405[c]	405	1003	88	2842	114.10	1570.19
US420[d]	420	1276	39	741	128.40	3155.18

[a] *BE581* network from Mahmutoğulları and Yaman (2023); [b] *CA339* network from Arslan et al. (2019)
[c] *QC405* network from Hydro-Québec; [d] *US420* network from Kınay et al. (2023).

Although our algorithm can handle OD demands with different vehicle ranges and deviation levels, we follow the standard setting in the literature and assume a common vehicle range $r_k = R$ and a common deviation parameter τ for

all demands. The parameter τ determines the maximum allowable route length, i.e., $\lambda_k = (1+\tau)\delta_{o_k d_k}$, where $\delta_{o_k d_k}$ is the shortest-path distance between o_k and d_k. To evaluate the scalability of the proposed algorithm, we consider four deviation levels, namely $\tau \in \{0\%, 10\%, 25\%, 50\%\}$. To facilitate the presentation of computational results, the OD-pair flows in each network are normalized and expressed as percentages of the total flow. Different instances are generated by varying the vehicle range R, the station budget p, and the deviation parameter τ. Table 6 summarizes the experimental settings considered for each network. As a result, each network generates 60 instances, totaling 240 instances.

Table 6 Experimental settings for each network

Network	# Instances	R (km)	τ	p
BE581	60	{200, 250, 300}	{0%, 10%, 25%, 50%}	{10, 20, 30, 40, 50}
CA339	60	{100, 150, 200}		{5, 10, 15, 20, 25}
QC405	60	{200, 300, 400}		{10, 20, 30, 40, 50}
US420	60	{300, 400, 500}		{10, 20, 30, 40, 50}

6.2.2. Algorithmic Configurations. Since our algorithmic enhancements are developed based on the B&C algorithm of Arslan et al. (2019), we further investigate the impact of different MWMC-LBC separation strategies on the computational performance of the RSLP. To this end, we develop four implementations, denoted by ALG1–ALG4, as summarized in Table 7. These implementations are constructed incrementally by gradually incorporating the proposed algorithmic components, including the pre-cut procedure, the ϵ-MIN procedure, separation at fractional solutions, the time-limited SPBA procedure, and the quick-exit strategy. This incremental comparison allows us to assess both the individual and combined effects of these components on the solution performance.

Table 7 Algorithmic Configurations for MWMC-LBC Separation in the RSLP

Algorithm	Pre-Cut	Integer		Fractional			
		ϵ-MIN	cut-min	ϵ-MIN	SPBA-t	cut-min	quick-exit
B&C (Arslan et al. 2019)			✓	✓		✓	
ALG1	✓		✓				
ALG2	✓	✓	✓				
ALG3	✓	✓	✓	✓		✓	
ALG4	✓	✓	✓	✓	$t = 0.01$	✓	✓

6.2.3. Performance of Different Algorithmic Configurations. All four configurations, ALG1–ALG4, can solve all 240 RSLP instances to optimality. The summary results are shown in Table 8. Compared with ALG1, ALG2 substantially reduces the average solution time from 20.09 to 9.40 seconds, while slightly improving the root-node gap and generating fewer cuts. This shows that the adaptive ϵ-MIN-based compression strategy substantially improves integer-solution separation. Extending separation to fractional solutions further strengthens the relaxation: ALG3 reduces the average root-node gap to 0.37%, and ALG4 achieves the smallest gap of 0.33%. These results indicate that, for the RSLP benchmark set, the configuration combining pre-generated structural cuts with the proposed integer-solution compression strategy achieves the largest reduction in overall solution time. Fractional separation further strengthens the root relaxation, but the additional cuts and separation effort lead to longer overall runtimes for ALG3 and ALG4 on this benchmark set.

Table 8 Computational Performance of the Four Algorithmic Configurations

Alg.	Avg Time (s)	Avg Root gap	# Avg Pre-Cuts	# Avg lb-Cuts
B&C (Arslan et al. 2019)	137.87	14.81%	0	6571.52
ALG1	20.09	0.63%	5368.10	593.99
ALG2	9.40	0.57%	5368.10	530.10
ALG3	18.56	0.37%	5368.10	1056.84
ALG4	22.71	0.33%	5368.10	1079.45

6.2.4. Head-to-Head Comparison. We select ALG2, achieving the shortest average solution time among the four configurations, for the subsequent head-to-head comparison. We compare it with the following two exact methods. The first benchmark algorithm is the current state-of-the-art (SOTA) B&C approach proposed by Arslan et al. (2019). The second benchmark algorithm is taken from the work of Kınay et al. (2023), who studied a bilevel charging station location problem. To solve the bilevel model on large-scale networks, their approach requires repeatedly solving a challenging location subproblem. For this purpose, they propose a logic-based Benders algorithm (LBBA) that is specifically designed to solve the underlying location problem efficiently. It is worth noting that the location problem considered in their work corresponds to a full-cover problem. Only minor modifications were required to adapt their implementation to solve the RSLP. We are grateful to the authors of both benchmark algorithms for making their implementations available, which enabled us to conduct a fair head-to-head comparison on the same computational platform. For each instance, both benchmark algorithms are run with a maximum time limit of one hour. The detailed results are shown in Tables 14 and 15.

Table 9 summarizes the computational results on the 240 benchmark instances. Overall, ALG2 successfully solves all instances to optimality within five minutes. In contrast, the LBBA algorithm of Kınay et al. (2023) encounters substantial difficulties on the larger networks. Specifically, LBBA fails to find a feasible solution within one hour for 8 instances on the QC405 network and 27 instances on the US420 network. Compared with the branch-and-cut algorithm of Arslan et al. (2019), the ALG2 consistently reduces the computational effort. On the BE581 network, the average solution time is reduced from 84.98 seconds to 2.94 seconds, corresponding to a speedup of nearly 30 times. Similar improvements are observed on the CA339 network, where the average runtime decreases from 34.71 seconds to 0.83 seconds. More importantly, the benefits become increasingly pronounced on the larger QC405 and US420 networks. For example, on QC405 with $R = 200$, the average runtime is reduced from 390.71 seconds to 30.59 seconds, while on US420 with $R = 500$, it decreases from 81.22 seconds to 16.50 seconds.

Table 9 Summary comparison results on the four networks

Network	R (km)	LBBA (Kınay et al. 2023)				B&C (Arslan et al. 2019)		ALG2	
		#Opt	#NFS	Avg Time (s)	Avg Opt Gap (%)	#Opt	Avg Time (s)	#Opt	Avg Time (s)
BE581	200	20/20	0	14.35	0.00	20/20	99.90	20/20	3.45
	250	20/20	0	13.10	0.00	20/20	82.91	20/20	2.63
	300	20/20	0	13.08	0.00	20/20	72.13	20/20	2.72
CA339	100	20/20	0	7.89	0.00	20/20	41.33	20/20	1.13
	150	20/20	0	1.44	0.00	20/20	36.84	20/20	0.95
	200	20/20	0	0.74	0.00	20/20	25.95	20/20	0.43
QC405	200	9/20	8	2091.05	0.64	20/20	390.71	20/20	30.59
	300	20/20	0	114.40	0.00	20/20	338.31	20/20	16.10
	400	20/20	0	13.85	0.00	20/20	274.97	20/20	9.60
US420	300	6/20	10	2634.53	105.23	20/20	119.12	20/20	15.77
	400	5/20	11	2807.00	51.95	20/20	91.01	20/20	12.93
	500	9/20	6	2085.02	39.94	20/20	81.22	20/20	16.50

#NFS is the number of instances for which no feasible solution was found within one hour.
Avg Opt Gap is computed only over instances with a feasible incumbent; NFS instances are excluded.

Table 10 reports the impact of the deviation parameter τ on computational performance. The ALG2 exhibits significantly better scalability than the benchmark methods. For the BE581 and CA339 networks, the ALG2 remains below 4 seconds and approximately 1 second, respectively, even when τ increases to 50%. In contrast, the runtime of the original B&C algorithm grows from 1.88 to 222.39 seconds on BE581 and from 0.99 to 84.32 seconds on CA339. The advantage becomes even more pronounced on the larger QC405 and US420 networks. For example, on QC405 with $\tau = 50\%$, the ALG2 requires only 45.74 seconds, compared with 859.15 seconds for the original B&C and 1123.64 seconds for LBBA. Similarly, on US420 with $\tau = 50\%$, the ALG2 solves the instances in 40.86 seconds on average, whereas the corresponding runtimes of B&C and LBBA are 271.30 and 3363.65 seconds, respectively. These results demonstrate that the proposed enhancements substantially improve the scalability of the branch-and-cut framework.

Table 10 Impact of the deviation parameter τ on average solution times (seconds)

BE581				*CA339*			
τ	LBBA	B&C	ALG2	τ	LBBA	B&C	ALG2
0%	7.29	1.88	1.77	0%	1.29	0.99	0.77
10%	11.38	20.14	2.48	10%	1.41	9.99	0.68
25%	18.02	95.52	3.62	25%	7.07	43.54	1.05
50%	17.36	222.39	3.87	50%	3.65	84.32	0.84
QC405				*US420*			
τ	LBBA	B&C	ALG2	τ	LBBA	B&C	ALG2
0%	286.23	4.80	5.15	0%	1021.22	0.80	0.75
10%	772.89	93.68	7.71	10%	2689.61	17.48	2.76
25%	776.30	381.02	16.46	25%	2960.92	98.88	15.89
50%	1123.64	859.15	45.74	50%	3363.65	271.30	40.86

LBBA: Kınay et al. (2023); B&C: Arslan et al. (2019)

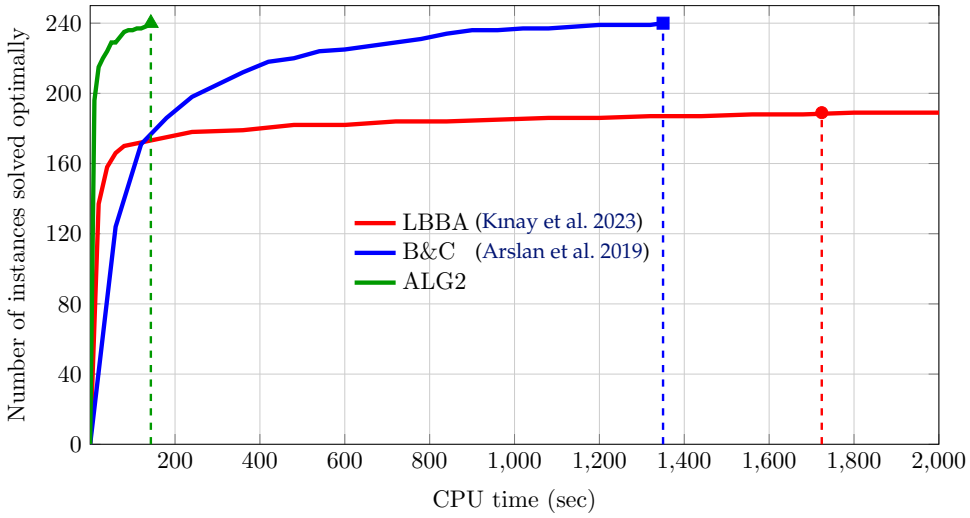

Figure 4 Comparison of cumulative number of RSLP instances solved to optimality within one hour

Fig. 4 further compares the cumulative number of instances solved to optimality within the one-hour time limit. The ALG2 algorithm consistently dominates the two benchmark algorithms throughout the entire time horizon. In particular, it solves all 240 instances within approximately 150 seconds, whereas the original B&C algorithm requires more

than 1,300 seconds to reach the same level of performance. In contrast, LBBA solves only 189 of the 240 instances within one hour, reflecting its difficulty in handling the larger QC405 and US420 networks. These results demonstrate that the proposed separation enhancements substantially improve the efficiency of the original branch-and-cut framework.

As shown in Table 11, the proposed enhancements dramatically reduce the number of dynamically generated lb-cuts throughout the branch-and-cut process. Compared with the original B&C algorithm of Arslan et al. (2019), the average number of generated cuts decreases by more than 90% on the BE581 and QC405 networks and by over 75% on the CA339 and US420 networks. These results are consistent with the intended role of the pre-generated structural cuts and the proposed integer-solution compression strategy, namely, to reduce the burden of dynamic separation. Since ALG2 combines several algorithmic enhancements relative to the original B&C, however, the observed reduction in solution time cannot be attributed to any single component in isolation. Overall, the results show that the combined strategy substantially reduces dynamic cut generation and achieves robust computational performance across all network sizes while solving every benchmark instance to proven optimality.

Table 11 Reduction in dynamically generated lb-cuts for the RSLP

Network	#Avg lb-cuts (B&C)	#Avg lb-cuts (ALG2)	Reduction
BE581	6145.28	14.90	99.76%
CA339	4076.25	70.98	98.26%
QC405	10541.87	845.33	91.98%
US420	5522.67	1189.20	78.47%

B&C: Arslan et al. (2019).

6.3. Computational Performance on the NDSR

This section presents the results on the NDSR.

6.3.1. NDSR Benchmark Instances and Experimental Setting. We use the benchmark instances originally generated by Balakrishnan et al. (2017) and later adopted by Gudapati et al. (2022). The instances are randomly generated directed networks with prescribed numbers of nodes, arcs, and commodities. Given these sizes, nodes are randomly located on a rectangular grid, a directed spanning tree is first constructed to ensure connectivity, and additional arcs are then added until the desired network density is reached. Arc fixed costs depend on both Euclidean distances and random components, while arc weights are generated to be negatively correlated with arc costs in order to create challenging service-constrained routing decisions. Each instance considers two service metrics, and the service limit of each commodity is determined from the length of a randomly selected q-th shortest origin–destination path, where q is randomly selected from a prescribed interval.

The benchmark set contains 90 instances, consisting of five instances for each of 18 scenarios. Scenarios 1–7 use the default cost and service-parameter settings and vary the network size from 30 nodes, 120 arcs, and 90 commodities to 50 nodes, 250 arcs, and 150 commodities. Scenarios 8–15 fix the network size at 50 nodes, 200 arcs, and 150 commodities, but vary the cost structure, the weight–cost correlation, and the tightness of the service requirements. Scenarios 16–18 return to the default parameter setting but consider larger networks, up to 80 nodes, 320 arcs, and 240 commodities. Each scenario in Table 13 is denoted by $|N|/|A|/|K|/\beta\gamma Q_{\text{avg}}\Delta Q$, where $|N|$, $|A|$, and $|K|$ denote the numbers of nodes, arcs, and commodities, respectively; the letters L, M, and H represent the low, medium, and high settings of the corresponding parameters, with β controlling the correlation between arc weights and activation costs, γ specifying the ratio of flow costs to activation costs, and Q_{avg} and ΔQ determining the average level and variability of the service requirements, respectively.

6.3.2. Algorithmic Configurations. To evaluate the effectiveness of the proposed pre-generated lb-cuts, we build on the path-based formulation of Gudapati et al. (2022), which currently represents the strongest benchmark for the NDSR and significantly improves upon the exact method of Balakrishnan et al. (2017). Gudapati et al. (2022) report 80 instances solved to optimality in their original experiments. When we reran their publicly available implementation on our computational platform, 79 out of the 90 benchmark instances were solved to optimality within the one-hour time limit. We observed that their implementation explicitly restricts Gurobi to a single CPU thread, whereas our implementation uses Gurobi's default thread setting. This difference in solver configuration contributes to the runtime difference between the two implementations.

In our experiments, we implement the same path-based formulation in Java but do not impose such a restriction on the number of CPU threads; instead, Gurobi is run with its default settings. To isolate the computational impact of the pre-generated lb-cuts, we compare two configurations using exactly the same Java implementation, computational environment, and solver settings. The first solves the path-based formulation without pre-generated lb-cuts. The second additionally incorporates the pre-generated S-CUT, T-CUT, and MIN-CUT inequalities for each commodity. Thus, the only algorithmic difference between the two configurations is the explicit inclusion of these redundant valid inequalities, allowing us to directly assess their computational impact.

6.3.3. Effectiveness of Pre-Generated lb-Cuts. Table 12 evaluates the computational impact of explicitly adding the pre-generated lb-cuts to the path-based NDSR formulation. For reference, the publicly available implementation of Gudapati et al. (2022), which restricts Gurobi to a single thread, solves 79 out of 90 instances to optimality on our computational platform. In contrast, our Java implementation uses the default Gurobi thread setting. Therefore, to isolate the effect of the pre-generated lb-cuts, we focus on the comparison between the last two configurations in Table 12, which use exactly the same implementation, computational environment, and solver settings.

Both configurations solve 85 out of 90 instances to optimality. Among these 85 commonly solved instances, the average solution time decreases from 162.85 seconds for the Path-Based formulation to 136.92 seconds after adding the pre-generated lb-cuts, corresponding to a 15.92% reduction. The average optimality gap over all 90 instances also decreases from 0.0842% to 0.0690%. On average, only 117.52 distinct lb-cuts are pre-generated for each instance after removing duplicates. These results show that, although the lb-cut inequalities are already implied by the linking and path-selection constraints and therefore do not strengthen the theoretical LP relaxation, explicitly adding a small number of structural lb-cuts at negligible preprocessing cost still provides a meaningful computational benefit. In particular, the controlled comparison indicates that the pre-generated lb-cuts reduce the average solution time by approximately 16%, while also providing a modest improvement in the final optimality gaps of the unsolved instances.

Table 12 Computational impact of pre-generated lb-cuts for the NDSR

Algorithm	Threads	#Opt	Avg. time (s)	Avg. Opt time (s)	Avg. opt. gap	Avg. #Pre lb-Cuts
Gudapati et al. (2022)	1	79/90	714.08	312.25	0.2614%	0
Path-Based	Default	85/90	353.81	162.85	0.0842%	0
Path-Based + Pre lb-cuts	Default	85/90	329.32	136.92	0.0690%	117.52

6.3.4. Summary Comparison. Table 13 presents a detailed comparison of the path-based formulation with and without the pre-generated lb-cuts on the 90 NDSR benchmark instances.

Both configurations solve 85 out of the 90 instances to proven optimality within the one-hour time limit. Among these 85 commonly solved instances, the average solution time decreases from 162.85 seconds without lb-cuts to 136.92 seconds with the pre-generated lb-cuts, corresponding to a 15.92% reduction. The improvement is observed across

many scenarios and is particularly noticeable in Scenarios 5–7 and 17. For example, the average solution time in Scenario 7 decreases from 292.08 to 183.34 seconds, while that in Scenario 17 decreases from 528.53 to 377.77 seconds. On the other hand, modest increases in solution time are observed in a few scenarios, such as Scenarios 13, 14, and 16, indicating that the computational benefit is not uniform across all instances. For the five instances that are not solved to proven optimality, the pre-generated lb-cuts also generally improve the final optimality gaps. Consequently, the average optimality gap over all 90 instances decreases from 0.08% to 0.07%. These results provide computational evidence that, although the pre-generated lb-cut inequalities are redundant from a formulation perspective and do not change the feasible region or LP relaxation, their explicit inclusion can nevertheless improve the overall efficiency of the MIP search.

Table 13 Summary comparison results on the NDSR benchmark instances

Scenario	$\|N\|/\|A\|/\|K\|$/Parameter	Path-Based			Path-Based + Pre lb-cuts		
		#Opt	Avg Gap	Time (s)	#Opt	Avg Gap	Time (s)
1	30/120/90/MMMM	5/5	0.00%	3.76	5/5	0.00%	2.74
2	40/160/120/MMMM	5/5	0.00%	6.62	5/5	0.00%	5.33
3	50/150/150/MMMM	5/5	0.00%	4.43	5/5	0.00%	4.14
4	50/200/100/MMMM	5/5	0.00%	22.88	5/5	0.00%	18.08
5	50/200/150/MMMM	5/5	0.00%	133.20	5/5	0.00%	100.47
6	50/200/200/MMMM	5/5	0.00%	102.76	5/5	0.00%	68.03
7	50/250/150/MMMM	5/5	0.00%	292.08	5/5	0.00%	183.34
8	50/200/150/LMMM	5/5	0.00%	14.89	5/5	0.00%	12.17
9	50/200/150/HMMM	3/5	0.69%	1555.55	3/5	0.63%	1552.61
10	50/200/150/MLMM	5/5	0.00%	394.34	5/5	0.00%	342.62
11	50/200/150/MHMM	5/5	0.00%	85.00	5/5	0.00%	74.23
12	50/200/150/MMLM	5/5	0.00%	2.34	5/5	0.00%	2.27
13	50/200/150/MMHM	4/5	0.40%	847.73	4/5	0.32%	863.87
14	50/200/150/MMML	5/5	0.00%	358.13	5/5	0.00%	362.51
15	50/200/150/MMMH	5/5	0.00%	103.80	5/5	0.00%	89.02
16	60/240/180/MMMM	5/5	0.00%	185.17	5/5	0.00%	207.11
17	70/280/210/MMMM	5/5	0.00%	528.53	5/5	0.00%	377.77
18	80/320/240/MMMM	3/5	0.43%	1727.27	3/5	0.28%	1661.43
Overall		85/90	0.08%	162.85*	85/90	0.07%	136.92*

*: Avg. time over the 85 commonly solved-to-optimality instances.

Overall, the results show that the proposed pre-generation strategy provides a simple and inexpensive enhancement to the path-based NDSR formulation. Rather than changing the strength of the formulation, the explicit redundant inequalities appear to facilitate the subsequent MIP solution process, leading to an average solution-time reduction of 15.92% on the commonly solved instances and modestly smaller optimality gaps on the unsolved instances. The detailed instance-level results are reported in Tables 16 and 17.

7. Conclusion

This paper introduced the minimum-weight minimum-cardinality length-bounded cut (MWMC-LBC) problem, which seeks a minimum-cardinality lb-cut among all minimum-weight solutions. To the best of our knowledge, this is the first study that explicitly incorporates cardinality minimization into the length-bounded cut problem and treats it as a standalone optimization problem.

To solve the MWMC-LBC problem, we proposed several exact and heuristic approaches, including a two-stage exact formulation, a single-stage ϵ-based formulation, a heuristic cardinality-reduction procedure, and a shortest-path branching algorithm for minimum-weight lb-cut separation. Extensive computational experiments on 1,500 collected

lb-cut instances demonstrated that cardinality minimization can significantly improve the quality of generated cuts, while the proposed methods provide an effective balance between solution quality and computational efficiency.

The proposed lb-cut generation framework was further evaluated within two state-of-the-art exact optimization approaches. For the RSLP, we developed an enhanced branch-and-cut algorithm that combines pre-generated structural lb-cuts with a fast integer-solution cut-compression procedure. Computational results on 240 benchmark instances from four realistic transportation networks showed that these enhancements substantially reduce the number of dynamically generated lb-cuts and improve computational performance, achieving average solution-time reductions of approximately 93.2% compared with the existing state-of-the-art approach of Arslan et al. (2019). For the NDSR, we showed that lb-cut inequalities are already implicitly enforced by the linking and path-selection constraints of the path-based formulation of Gudapati et al. (2022). Nevertheless, a small number of structural lb-cuts can be pre-generated at negligible computational cost and explicitly added to the formulation as redundant valid inequalities. In a controlled comparison using the same Java implementation and default Gurobi settings, both configurations solve 85 out of 90 instances to proven optimality. Among these 85 commonly solved instances, adding the pre-generated lb-cuts reduces the average solution time from 162.85 to 136.92 seconds, corresponding to a 15.92% reduction, while also modestly reducing the final optimality gaps of the unsolved instances.

Overall, the results demonstrate that lb-cuts can play different but computationally valuable roles in exact algorithms for network optimization problems. In the RSLP, they serve as essential inequalities that can be dynamically generated and improved through cardinality-aware cut selection, whereas in the path-based NDSR formulation, they serve as redundant valid inequalities whose explicit pre-generation can meaningfully improve computational efficiency. Beyond the RSLP and NDSR, the proposed MWMC-LBC methods are applicable to a broad class of optimization problems in which length-bounded cuts arise as separation subproblems or can be exploited as structural valid inequalities.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants RGPIN-2022-04553 and ALLRP 577172-2022. The authors gratefully acknowledge Mahmutoğulları and Yaman (2023) for providing the Belgium network; Hydro-Québec for providing the Québec road network; Kınay et al. (2023) for providing the U.S. highway network and the source code of their LBBA algorithm; Balakrishnan et al. (2017) for providing the NDSR benchmark instances; and Gudapati et al. (2022) for providing the source code of their path-based algorithm. These benchmark instances, transportation networks, and source codes facilitated the computational comparisons reported in this study.

References

Arslan O, Jabali O, Laporte G (2018) Exact solution of the evasive flow capturing problem. *Operations Research* 66(6):1625–1640.

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

Arslan O, Karaşan OE (2016) A Benders decomposition approach for the charging station location problem with plug-in hybrid electric vehicles. *Transportation Research Part B: Methodological* 93:670–695.

Arslan O, Karaşan OE, Mahjoub AR, Yaman H (2019) A branch-and-cut algorithm for the alternative fuel refueling station location problem with routing. *Transportation Science* 53(4):1107–1125.

Baier G, Erlebach T, Hall A, Köhler E, Kolman P, Pangrác O, Schilling H, Skutella M (2010) Length-bounded cuts and flows. *ACM Transactions on Algorithms (TALG)* 7(1):1–27.

Balakrishnan A, Li G, Mirchandani P (2017) Optimal network design with end-to-end service requirements. *Operations Research* 65(3):729–750.

Capar I, Kuby M, Leon VJ, Tsai YJ (2013) An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations. *European Journal of Operational Research* 227(1):142–151.

Chung SH, Kwon C (2015) Multi-period planning for electric car charging station locations: A case of Korean expressways. *European Journal of Operational Research* 242(2):677–687.

Codato G, Fischetti M (2006) Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research* 54(4):756–766.

Exoo G (1983) On line disjoint paths of bounded length. *Discrete Mathematics* 44(3):317–318.

Fischetti M, Salvagnin D, Zanette A (2010) A note on the selection of Benders' cuts. *Mathematical Programming* 124(1):175–182.

Ford Jr LR, Fulkerson DR (1956) Maximal flow through a network. *Canadian Journal of Mathematics* 8:399–404.

Golovach PA, Thilikos DM (2011) Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization* 8(1):72–86.

Göpfert P, Bock S (2019) A branch and cut approach to recharging and refueling infrastructure planning. *European Journal of Operational Research* 279(3):808–823.

Gudapati NV, Malaguti E, Monaci M (2022) Network design with service requirements: Scaling-up the size of solvable problems. *INFORMS Journal on Computing* 34(5):2571–2582.

Gudapati NV, Malaguti E, Monaci M, Paronuzzi P (2025) A computational study on integer programming formulations for hop-constrained survivable network design. *Discrete Applied Mathematics* 362:71–81.

Hamacher H (1982) Determining minimal cuts with a minimal number of arcs. *Networks* 12(4):493–504.

He J, Yang H, Tang TQ, Huang HJ (2018) An optimal charging station location model with the consideration of electric vehicle's driving range. *Transportation Research Part C: Emerging Technologies* 86:641–654.

Itai A, Perl Y, Shiloach Y (1982) The complexity of finding maximum disjoint paths with length constraints. *Networks* 12(3):277–286.

Kınay ÖB, Gzara F, Alumur SA (2021) Full cover charging station location problem with routing. *Transportation Research Part B: Methodological* 144:1–22.

Kınay ÖB, Gzara F, Alumur SA (2023) Charging station location and sizing for electric vehicles under congestion. *Transportation Science* 57(6):1433–1451.

Kuby M, Lim S (2005) The flow-refueling location problem for alternative-fuel vehicles. *Socio-Economic Planning Sciences* 39(2):125–145.

Lovász L, Neumann-Lara V, Plummer M (1978) Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica* 9(4):269–276.

Mahjoub AR, McCormick ST (2010) Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. *Mathematical Programming* 124(1):271–284.

Mahmutoğulları Ö, Yaman H (2023) Robust alternative fuel refueling station location problem with routing under decision-dependent flow uncertainty. *European Journal of Operational Research* 306(1):173–188.

Mahmutoğulları Ö, Yaman H (2024) Mathematical formulations for the multi-period alternative fuel refueling station location problem with routing under decision-dependent flow dynamics. *Transportation Research Part B: Methodological* 186:102999.

MirHassani SA, Ebrazi R (2013) A flexible reformulation of the refueling station location problem. *Transportation Science* 47(4):617–628.

Park H, Lee C (2024) An exact algorithm for maximum electric vehicle flow coverage problem with heterogeneous chargers, nonlinear charging time and route deviations. *European Journal of Operational Research* 315(3):926–951.

Plaza CV, Arslan O, Laporte G, Ribeiro GM, Bahiense L, Clímaco G (2026) The charger location problem with routing and driver working hours for long-haul electric heavy-duty trucks. *Transportation Research Part C: Emerging Technologies* 186:105598.

Wang W, Dems A, Adulyasak Y, Arslan O, Cordeau JF (2026) Electric vehicle fast-charging facility location with endogenous queuing in path selection. Available at SSRN 6466686.

Xie F, Liu C, Li S, Lin Z, Huang Y (2018) Long-term strategic planning of inter-city fast charging infrastructure for battery electric vehicles. *Transportation Research Part E: Logistics and Transportation Review* 109:261–276.

Xu M, Meng Q (2020) Optimal deployment of charging stations considering path deviation and nonlinear elastic demand. *Transportation Research Part B: Methodological* 135:120–142.

Yıldız B, Arslan O, Karaşan OE (2016) A branch and price approach for routing and refueling station location model. *European Journal of Operational Research* 248(3):815–826.

Zhang A, Kang JE, Kwon C (2017) Incorporating demand dynamics in multi-period capacitated fast-charging location planning for electric vehicles. *Transportation Research Part B: Methodological* 103:5–29.

Zhang H, Moura SJ, Hu Z, Song Y (2016) PEV fast-charging station siting and sizing on coupled transportation and power networks. *IEEE Transactions on Smart Grid* 9(4):2595–2605.

Appendix A: Network Illustrations

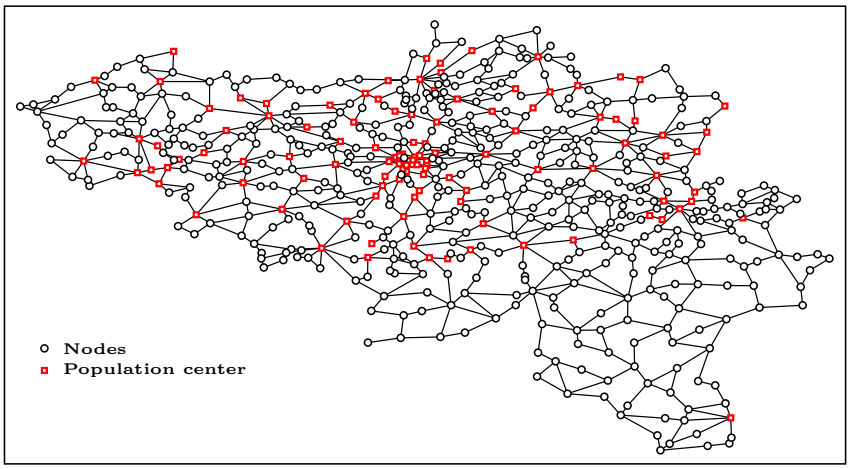

Figure 5 Belgium Network

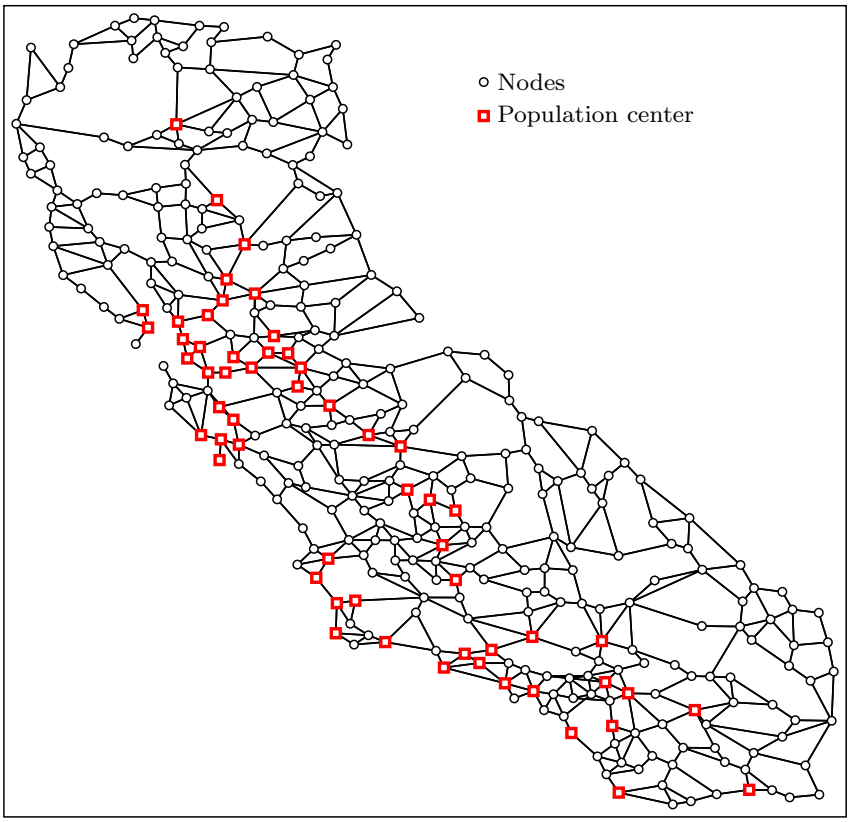

Figure 6 California Network

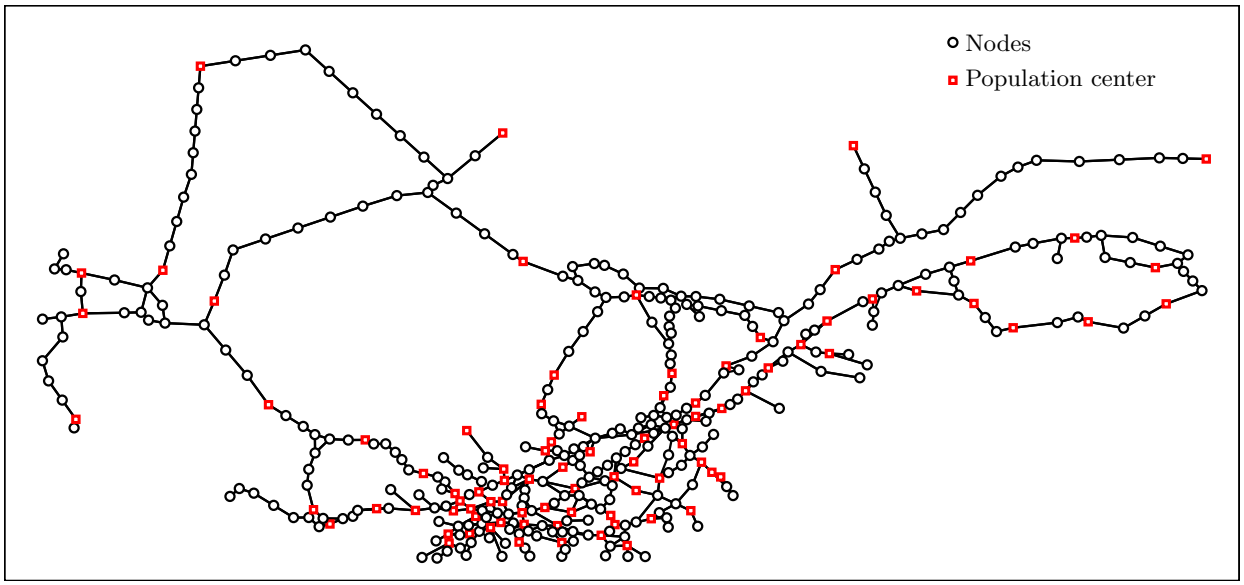

Figure 7 **Québec Network**

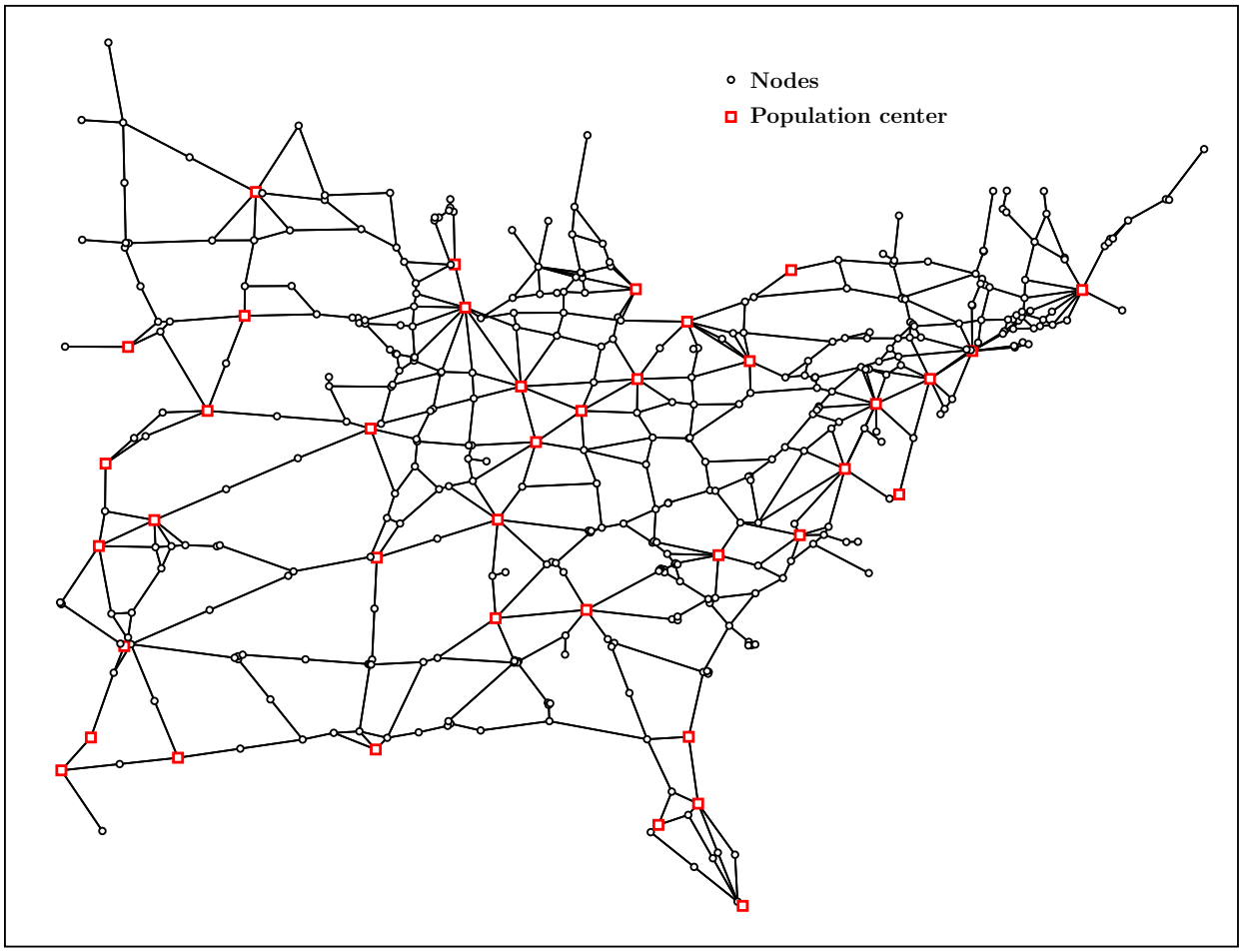

Figure 8 **United States Network**

Appendix B: Detailed Computational Results

Table 14 Detailed computational results for RSLP on the BE581 and CA339 networks

BE581								*CA339*							
R (km)	p	τ	Obj (%)	B&C		ALG2		R (km)	p	τ	Obj (%)	B&C		ALG2	
				#lb-cut	Time (s)	#lb-cut	Time (s)					#lb-cut	Time (s)	#lb-cut	Time (s)
200	10	0	69.05	12032	1.32	0	0.51	100	5	0	67.08	4958	1.14	86	0.14
		0.1	72.88	9989	30.07	180	7.29			0.1	79.57	5255	14.51	59	0.55
		0.25	76.84	7840	134.96	99	8.18			0.25	84.11	6034	73.61	344	4.57
		0.5	80.80	5946	272.66	65	5.60			0.5	90.46	6173	125.63	211	2.62
	20	0	85.30	9665	2.07	0	1.50		10	0	87.98	4646	0.98	16	0.32
		0.1	88.01	7349	23.93	3	1.07			0.1	93.47	5254	14.13	537	1.74
		0.25	89.76	6141	111.46	20	2.72			0.25	96.06	4752	58.60	299	2.73
		0.5	91.99	4880	262.57	4	3.68			0.5	98.82	4186	99.57	105	2.00
	30	0	92.85	9257	1.94	0	2.29		15	0	95.01	5395	1.33	87	0.52
		0.1	93.95	6910	24.40	120	3.04			0.1	98.88	4115	11.78	92	0.56
		0.25	94.92	5549	105.85	4	3.36			0.25	99.57	4146	49.44	27	0.36
		0.5	96.29	4009	247.22	4	4.02			0.5	100.00	3851	88.36	17	0.19
	40	0	96.63	7980	2.31	0	3.16		20	0	98.41	5551	1.26	266	1.52
		0.1	97.17	6472	28.15	121	5.57			0.1	99.82	4187	11.90	118	0.89
		0.25	97.86	5638	129.15	2	3.07			0.25	100.00	4317	49.71	17	0.20
		0.5	98.70	3483	244.98	0	2.77			0.5	100.00	3965	89.70	2	0.19
	50	0	98.75	6991	2.32	12	1.97		25	0	99.79	5375	1.45	554	1.05
		0.1	98.99	5673	24.11	81	2.42			0.1	99.99	3961	11.17	8	0.62
		0.25	99.31	4490	104.07	8	3.74			0.25	100.00	3600	38.85	19	0.67
		0.5	99.72	3301	244.52	0	3.09			0.5	100.00	3750	83.37	103	1.13
250	10	0	70.54	11092	1.31	0	0.44	150	5	0	79.94	4455	0.90	0	0.09
		0.1	74.27	8657	18.82	0	1.13			0.1	85.91	4265	11.65	15	0.45
		0.25	77.29	6678	91.57	1	2.35			0.25	90.28	4481	57.93	79	1.09
		0.5	80.90	5040	217.98	0	2.86			0.5	94.51	4047	109.99	36	2.11
	20	0	86.12	8772	1.84	0	1.33		10	0	92.98	4186	0.90	78	3.47
		0.1	88.18	6879	18.19	0	1.20			0.1	97.40	4312	10.62	285	1.00
		0.25	89.77	5460	91.92	0	3.54			0.25	98.50	3867	47.40	68	1.54
		0.5	91.99	4151	219.39	0	2.81			0.5	99.80	3720	90.85	30	2.00
	30	0	93.23	8295	1.71	0	1.46		15	0	98.35	4235	1.08	89	1.82
		0.1	94.01	6432	18.61	44	2.40			0.1	99.79	3793	9.58	12	0.31
		0.25	94.92	4934	90.16	0	3.10			0.25	99.96	3785	44.05	179	2.88
		0.5	96.29	3439	215.39	0	4.66			0.5	100.00	3558	87.71	17	0.24
	40	0	96.73	7465	2.38	0	4.80		20	0	99.89	3878	0.99	1	0.67
		0.1	97.19	5890	18.66	66	3.10			0.1	99.98	3733	9.35	1	0.40
		0.25	97.86	4890	99.18	0	3.14			0.25	100.00	3534	41.37	10	0.17
		0.5	98.70	3118	219.12	0	3.50			0.5	100.00	3533	87.04	5	0.19
	50	0	98.82	6629	2.88	0	0.96		25	0	100.00	3823	0.76	0	0.07
		0.1	98.99	5306	19.06	35	1.99			0.1	100.00	3539	8.95	13	0.13
		0.25	99.31	4080	90.45	6	4.60			0.25	100.00	3523	38.64	9	0.17
		0.5	99.72	2941	219.62	0	3.25			0.5	100.00	3470	77.13	5	0.20
300	10	0	71.06	10746	1.35	0	0.56	200	5	0	85.18	3882	0.86	0	0.08
		0.1	74.49	8226	15.23	0	1.10			0.1	90.53	3905	8.17	189	2.78
		0.25	77.32	6292	76.41	1	2.78			0.25	95.11	3679	32.16	14	0.17
		0.5	80.91	4654	191.12	1	4.11			0.5	97.52	3715	70.68	55	0.78
	20	0	86.43	8566	1.62	0	1.29		10	0	95.64	3914	0.78	0	0.62
		0.1	88.20	6610	14.88	0	0.88			0.1	98.63	3984	8.04	0	0.12
		0.25	89.77	5037	76.63	0	2.52			0.25	99.24	3848	35.85	6	0.38
		0.5	91.99	3787	197.39	0	3.90			0.5	99.90	3688	70.81	16	0.35
	30	0	93.34	8056	1.62	0	1.02		15	0	99.21	3964	0.93	6	0.44
		0.1	94.01	6198	15.60	5	1.67			0.1	99.86	3648	7.08	4	0.31
		0.25	94.92	4608	76.48	0	2.98			0.25	99.97	3535	31.03	37	0.58
		0.5	96.31	3329	195.83	0	5.80			0.5	100.00	3480	62.62	6	0.20
	40	0	96.79	7175	2.05	0	3.36		20	0	99.96	3739	0.81	3	0.67
		0.1	97.21	5704	16.37	5	2.20			0.1	100.00	3499	6.58	0	0.23
		0.25	97.86	4490	77.63	3	4.33			0.25	100.00	3477	28.23	3	0.14
		0.5	98.72	3053	191.72	0	3.39			0.5	100.00	3470	62.19	9	0.16
	50	0	98.84	6528	1.53	0	1.91		25	0	100.00	3570	0.67	0	0.07
		0.1	99.01	5127	16.04	2	2.07			0.1	100.00	3494	6.31	0	0.11
		0.25	99.30	3911	76.81	2	3.97			0.25	100.00	3450	26.16	3	0.15
		0.5	99.72	2877	196.36	0	4.63			0.5	100.00	3426	59.13	9	0.19

B&C: Arslan et al. (2019)

Table 15 Detailed computational results for RSLP on the QC405 and US420 networks

QC405								*US420*							
R (km)	p	τ	Obj (%)	B&C		ALG2		R (km)	p	τ	Obj (%)	B&C		ALG2	
				#lb-cut	Time (s)	#lb-cut	Time (s)					#lb-cut	Time (s)	#lb-cut	Time (s)
200	10	0	29.91	13211	6.60	126	13.04	300	10	0	28.02	3116	0.55	34	0.24
		0.1	45.05	13204	115.31	927	14.96			0.1	31.12	3558	10.31	127	2.36
		0.25	55.22	15343	516.76	2374	49.62			0.25	32.27	4690	51.92	264	2.00
		0.5	61.98	16130	1106.51	3843	122.99			0.5	35.29	4951	136.99	410	2.63
	20	0	59.94	13374	9.69	920	15.99		20	0	41.42	3424	0.63	112	0.30
		0.1	79.83	12538	127.87	3107	26.42			0.1	47.68	4175	25.80	237	0.67
		0.25	92.87	13855	498.99	1259	17.76			0.25	50.76	5739	68.13	487	2.54
		0.5	97.19	17335	1350.79	2932	63.26			0.5	54.07	10330	354.41	1645	12.75
	30	0	83.46	12250	8.85	916	22.91		30	0	52.04	3555	0.70	295	0.51
		0.1	97.15	12398	106.17	3658	25.68			0.1	58.18	5007	16.88	712	2.00
		0.25	99.76	11293	392.66	2593	33.71			0.25	64.21	8723	127.29	2297	16.26
		0.5	99.97	12997	978.42	5585	139.61			0.5	69.96	13271	491.73	3713	63.76
	40	0	97.52	12896	6.64	1667	8.32		40	0	61.57	3960	0.70	572	0.45
		0.1	99.96	10896	94.92	956	4.27			0.1	68.93	5334	17.42	970	2.39
		0.25	100.00	10441	372.92	861	7.78			0.25	76.59	9912	153.31	4005	30.84
		0.5	100.00	11720	866.39	786	13.33			0.5	86.35	14824	553.45	5009	86.18
	50	0	99.97	12340	4.19	708	1.35		50	0	69.94	4030	0.71	624	0.67
		0.1	100.00	10995	98.76	789	2.34			0.1	78.91	5328	16.52	1086	4.45
		0.25	100.00	10683	356.75	946	12.33			0.25	88.04	8466	122.22	3937	38.22
		0.5	100.00	10585	794.95	925	16.23			0.5	95.45	7547	232.67	2809	46.08
300	10	0	52.18	11325	4.70	4	7.86	400	10	0	35.53	3213	0.64	36	0.67
		0.1	67.88	11367	109.10	247	7.55			0.1	40.55	3760	13.59	77	0.80
		0.25	79.16	11813	477.58	917	26.46			0.25	42.34	5915	121.96	238	2.27
		0.5	86.52	13939	1182.70	2619	101.34			0.5	44.65	7128	291.49	517	5.39
	20	0	87.51	10916	3.95	39	1.47		20	0	47.98	3585	0.84	237	1.62
		0.1	97.93	10599	100.32	743	8.17			0.1	55.26	6331	31.79	754	2.67
		0.25	99.77	9416	378.07	460	12.51			0.25	62.19	8424	153.57	1935	19.20
		0.5	100.00	10407	864.90	1272	67.99			0.5	69.07	11224	482.49	2820	43.50
	30	0	99.72	10317	3.91	5	0.71		30	0	60.03	3865	1.04	421	1.13
		0.1	100.00	10000	97.51	253	1.81			0.1	70.25	5393	22.03	1609	7.15
		0.25	100.00	9302	372.05	289	8.82			0.25	79.99	8052	130.81	3337	42.14
		0.5	100.00	9462	741.69	531	19.28			0.5	89.36	6212	206.21	3804	76.67
	40	0	100.00	10131	3.45	87	0.65		40	0	71.86	3942	0.92	528	0.58
		0.1	100.00	9539	91.30	214	1.29			0.1	83.76	5121	18.17	975	3.67
		0.25	100.00	9469	387.43	488	11.57			0.25	93.63	4782	60.87	1406	15.78
		0.5	100.00	9324	789.14	585	22.78			0.5	98.40	4475	131.13	1412	16.24
	50	0	100.00	10411	3.62	0	0.16		50	0	82.01	4039	0.88	743	0.57
		0.1	100.00	9030	81.70	525	3.28			0.1	93.16	4209	12.27	1081	3.48
		0.25	100.00	8869	345.82	170	6.35			0.25	98.92	3900	45.03	544	6.77
		0.5	100.00	8610	727.31	450	11.98			0.5	100.00	3219	94.38	562	8.39
400	10	0	75.05	9656	3.28	0	0.63	500	10	0	38.89	3313	0.71	37	0.65
		0.1	84.98	9413	87.60	51	7.19			0.1	43.93	4036	15.57	36	0.36
		0.25	94.21	9745	388.70	1232	44.51			0.25	46.37	6572	124.90	247	2.57
		0.5	98.91	9323	816.54	1441	78.42			0.5	51.76	8063	325.24	1139	18.09
	20	0	98.43	10321	4.00	56	2.01		20	0	54.82	3736	0.88	286	1.79
		0.1	99.92	8711	78.98	257	4.97			0.1	64.05	4770	20.07	471	4.22
		0.25	100.00	8558	335.46	134	7.75			0.25	70.50	7387	155.65	2262	35.11
		0.5	100.00	8477	706.72	277	12.56			0.5	80.00	9996	421.58	5133	143.54
	30	0	99.99	9183	3.49	253	1.83		30	0	71.43	3740	0.90	457	0.76
		0.1	100.00	8743	73.82	148	2.18			0.1	82.01	4887	19.13	657	2.80
		0.25	100.00	8635	330.64	92	3.41			0.25	90.41	5602	81.58	1264	17.03
		0.5	100.00	8618	690.36	272	7.10			0.5	95.97	4774	157.42	2775	72.80
	40	0	100.00	9177	2.90	1	0.23		40	0	83.75	3988	0.98	571	0.81
		0.1	100.00	8331	74.11	242	3.26			0.1	94.17	4254	13.90	684	3.25
		0.25	100.00	7955	300.15	44	2.46			0.25	98.85	3725	48.39	661	5.93
		0.5	100.00	7795	628.46	143	4.72			0.5	99.92	2870	96.10	665	9.44
	50	0	100.00	8605	2.78	0	0.16		50	0	92.78	4171	0.96	577	0.51
		0.1	100.00	7890	67.67	189	2.27			0.1	99.55	3238	8.69	256	1.11
		0.25	100.00	7304	261.33	30	1.82			0.25	100.00	2869	37.63	222	1.72
		0.5	100.00	7342	642.32	82	4.56			0.5	100.00	2640	94.22	571	7.51

B&C: Arslan et al. (2019)

Table 16 Detailed computational results for NDSR on Scenarios 1 to 9

Scenario	Instance	Path-Based			Path-Based + Pre lb-cuts		
		Cost	Time (s)	Opt Gap	Cost	Time (s)	Opt Gap
1	N30A120C90_8405	4182	4.30	0.00%	4182	3.80	0.00%
	N30A120C90_1943	3431	3.29	0.00%	3431	2.59	0.00%
	N30A120C90_7984	4206	3.32	0.00%	4206	2.72	0.00%
	N30A120C90_5138	3700	3.28	0.00%	3700	2.01	0.00%
	N30A120C90_6703	3720	4.60	0.00%	3720	2.59	0.00%
2	N40A160C120_3861	6040	16.96	0.00%	6040	13.79	0.00%
	N40A160C120_1205	5037	2.18	0.00%	5037	2.23	0.00%
	N40A160C120_1448	4963	5.32	0.00%	4963	3.02	0.00%
	N40A160C120_2415	5451	5.74	0.00%	5451	5.07	0.00%
	N40A160C120_2845	5618	2.91	0.00%	5618	2.53	0.00%
3	N50A150C150_8958	7462	1.11	0.00%	7462	0.82	0.00%
	N50A150C150_1116	7404	3.70	0.00%	7404	4.35	0.00%
	N50A150C150_3118	6530	3.13	0.00%	6530	2.46	0.00%
	N50A150C150_7176	6483	5.45	0.00%	6483	4.28	0.00%
	N50A150C150_5958	6978	8.77	0.00%	6978	8.79	0.00%
4	N50A200C100_8226	5811	33.54	0.00%	5811	27.18	0.00%
	N50A200C100_1287	5915	39.56	0.00%	5915	18.03	0.00%
	N50A200C100_3697	6109	2.95	0.00%	6109	3.51	0.00%
	N50A200C100_7554	5674	31.50	0.00%	5674	35.26	0.00%
	N50A200C100_7859	5995	6.86	0.00%	5995	6.43	0.00%
5	N50A200C150_6077	6964	410.64	0.00%	6964	294.34	0.00%
	N50A200C150_1788	7037	202.51	0.00%	7037	173.07	0.00%
	N50A200C150_3801	6846	21.18	0.00%	6846	10.41	0.00%
	N50A200C150_5508	6529	20.23	0.00%	6529	13.87	0.00%
	N50A200C150_5890	6402	11.46	0.00%	6402	10.67	0.00%
6	N50A200C200_8273	8303	459.33	0.00%	8303	291.05	0.00%
	N50A200C200_1025	7656	12.27	0.00%	7656	12.86	0.00%
	N50A200C200_4325	7964	11.28	0.00%	7964	13.45	0.00%
	N50A200C200_4546	8829	16.34	0.00%	8829	10.98	0.00%
	N50A200C200_7098	7854	14.56	0.00%	7854	11.82	0.00%
7	N50A250C150_9563	6982	92.01	0.00%	6982	87.90	0.00%
	N50A250C150_1306	6294	42.54	0.00%	6294	34.67	0.00%
	N50A250C150_5249	6732	105.66	0.00%	6732	91.99	0.00%
	N50A250C150_5278	6723	1205.16	0.00%	6723	691.81	0.00%
	N50A250C150_7421	6170	15.05	0.00%	6170	10.30	0.00%
8	N50A200C150_6077	6998	46.62	0.00%	6998	29.80	0.00%
	N50A200C150_1788	7531	10.86	0.00%	7531	8.99	0.00%
	N50A200C150_3801	6955	3.99	0.00%	6955	4.66	0.00%
	N50A200C150_5508	6976	7.54	0.00%	6976	11.37	0.00%
	N50A200C150_5890	6615	5.45	0.00%	6615	6.02	0.00%
9	N50A200C150_6077	7098	3600.03	2.18%	7098	3600.17	1.90%
	N50A200C150_1788	7284	3600.13	1.26%	7284	3600.11	1.26%
	N50A200C150_5508	6773	219.83	0.00%	6773	273.03	0.00%
	N50A200C150_3801	6759	209.73	0.00%	6759	131.72	0.00%
	N50A200C150_5890	6534	148.01	0.00%	6534	158.02	0.00%

Table 17 Detailed computational results for NDSR on Scenarios 10 to 18

Scenario	Instance	Path-Based			Path-Based + Pre lb-cuts		
		Cost	Time (s)	Opt Gap	Cost	Time (s)	Opt Gap
10	N50A200C150_6077	5816	1259.31	0.00%	5816	1003.95	0.00%
	N50A200C150_1788	5922	243.34	0.00%	5922	323.70	0.00%
	N50A200C150_3801	5770	119.53	0.00%	5770	58.10	0.00%
	N50A200C150_5508	5805	288.58	0.00%	5805	287.11	0.00%
	N50A200C150_5890	5539	60.94	0.00%	5539	40.21	0.00%
11	N50A200C150_1788	8540	269.61	0.00%	8540	226.73	0.00%
	N50A200C150_3801	8104	8.11	0.00%	8104	6.65	0.00%
	N50A200C150_5508	8020	15.62	0.00%	8020	9.45	0.00%
	N50A200C150_5890	7799	9.17	0.00%	7799	7.92	0.00%
	N50A200C150_6077	8210	122.51	0.00%	8210	120.39	0.00%
12	N50A200C150_6077	7727	4.88	0.00%	7727	4.99	0.00%
	N50A200C150_1788	8289	1.09	0.00%	8289	1.08	0.00%
	N50A200C150_3801	7458	2.81	0.00%	7458	2.14	0.00%
	N50A200C150_5508	7543	1.04	0.00%	7543	1.51	0.00%
	N50A200C150_5890	7132	1.87	0.00%	7132	1.63	0.00%
13	N50A200C150_6077	6830*	3600.09	1.98%	6846*	3600.08	1.62%
	N50A200C150_1788	7117	265.92	0.00%	7117	326.25	0.00%
	N50A200C150_3801	6556	106.73	0.00%	6556	72.77	0.00%
	N50A200C150_5508	6496	88.95	0.00%	6496	109.83	0.00%
	N50A200C150_5890	6391	176.98	0.00%	6391	210.43	0.00%
14	N50A200C150_1788	7356	219.26	0.00%	7356	198.06	0.00%
	N50A200C150_3801	6947	215.48	0.00%	6947	276.80	0.00%
	N50A200C150_5508	6601	141.74	0.00%	6601	222.04	0.00%
	N50A200C150_5890	6513	28.53	0.00%	6513	13.97	0.00%
	N50A200C150_6077	7077	1185.64	0.00%	7077	1101.70	0.00%
15	N50A200C150_6077	7081	276.98	0.00%	7081	247.16	0.00%
	N50A200C150_1788	7566	166.42	0.00%	7566	154.52	0.00%
	N50A200C150_3801	6786	19.21	0.00%	6786	12.06	0.00%
	N50A200C150_5508	6779	30.10	0.00%	6779	17.05	0.00%
	N50A200C150_5890	6654	26.28	0.00%	6654	14.33	0.00%
16	N60A240C180_7479	8394	234.11	0.00%	8394	216.36	0.00%
	N60A240C180_1064	9031	186.32	0.00%	9031	230.70	0.00%
	N60A240C180_2161	8022	198.56	0.00%	8022	213.77	0.00%
	N60A240C180_2751	8027	280.04	0.00%	8027	361.77	0.00%
	N60A240C180_4866	8217	26.81	0.00%	8217	12.95	0.00%
17	N70A280C210_3668	11182	716.63	0.00%	11182	493.52	0.00%
	N70A280C210_3704	9979	57.12	0.00%	9979	83.87	0.00%
	N70A280C210_5973	10663	533.13	0.00%	10663	234.69	0.00%
	N70A280C210_9452	10749	673.63	0.00%	10749	569.56	0.00%
	N70A280C210_9630	10481	662.13	0.00%	10481	507.18	0.00%
18	N80A320C240_4874	11744	717.53	0.00%	11744	551.87	0.00%
	N80A320C240_5469	11731	219.49	0.00%	11731	163.48	0.00%
	N80A320C240_6371	11697	499.06	0.00%	11697	391.68	0.00%
	N80A320C240_7362	11292	3600.13	1.26%	11292	3600.05	0.75%
	N80A320C240_8411	12127*	3600.15	0.90%	12132*	3600.04	0.67%

*: For unsolved instances, a smaller optimality gap does not necessarily imply a better incumbent objective.