ROUTING AND NETWORK DESIGN PROBLEMS: NEW ALGORITHMS AND OPTIMIZATION TECHNIQUES

ZHANG SILONG

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University
Department of Logistics and Maritime Studies

Routing and Network Design Problems: New Algorithms and Optimization Techniques

ZHANG Silong

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
December 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: ZHANG Silong

Abstract

Routing and network design problems are fundamental to optimizing the flow of resources, services, and information across interconnected systems. These problems are critical for ensuring the efficiency, scalability, and reliability of modern network infrastructure but present significant computational challenges due to complex operational constraints, vast solution spaces, and stringent survivability requirements. This thesis addresses three key challenges in this domain by developing innovative algorithms and optimization techniques: an enhanced exact algorithm for tackling multitrip scheduling in vehicle routing, novel optimality cuts and domain reduction techniques to reduce the search space of integer programs (including vehicle routing problems), and an efficient heuristic to address the scalability and survivability in large-scale telecommunications networks under double-link failures.

The first study presents an enhanced exact algorithm for the capacitated multitrip vehicle routing problem with time windows (CMTVRPTW). This problem extends the classical vehicle routing problem by incorporating time windows and allowing vehicles to perform multiple trips. While this problem captures operational requirements of real-world logistics networks, it is significantly more challenging to solve. In this study, we present an enhanced exact algorithm to obtain the optimal integer solution by solving a trip-based formulation, which allows for an efficient solution method to the constraint separation problem. To further

enhance computational efficiency, we reduce the number of variables by eliminating trips using the dominance rule, variable fixing technique, and optimality property of vehicle departure times. Additionally, a dynamic time discovery strategy is introduced to reduce the number of constraints. Computational experiments demonstrate that our enhanced algorithm is highly effective for solving challenging instances of the CMTVRPTW. On 14 benchmark instances previously unsolved or considered difficult, which involve 140–200 customers, our method outperforms the current state-of-the-art exact algorithm within a ten-hour time limit, solving 6 additional instances to optimality and reducing average runtime by over 20%.

The second study concerns the effective reduction of the solution search space for integer programs, particularly in the context of routing problems. Integer programming formulations for combinatorial optimization problems face a significant computational challenge due to the exponential growth of the solution space. To address this challenge, we introduce Lagrangian underestimate cuts (LUCs), a novel class of optimality cuts derived from Lagrangian duals, including infeasible LP duals. We utilize LUCs to strengthen reduced cost cuts and to derive new domain reduction techniques. Additionally, we leverage LUCs to develop new arc fixing techniques for the vehicle routing problem with time windows. These methods demonstrate significant potential for pruning solution spaces.

The third study investigates the survivable traffic grooming problem under double-link failures (STG2) in large-scale telecommunications networks, where each communication demand must be assigned a route for every possible scenario involving zero, one, or two failed fiber links. Ensuring protection against double-link failures is critical for maintaining reliable telecommunications services while minimizing equipment costs. To the best of our knowledge, this problem remains unexplored in the existing literature, with current research primarily focusing on

simpler problem settings involving smaller networks, typically with no more than 300 nodes. To address this gap, we propose a novel hierarchical constructive heuristic for the large-scale STG2 problem. It incorporates various techniques, including demand aggregation, a reuse-then-search routing strategy, and parallel computing, to significantly enhance efficiency. Extensive experiments have been conducted on large-scale STG2 instances provided by an industry partner, encompassing networks with 1,000 to 2,600 nodes. The results demonstrate that within a one-hour time limit and a 16 GB memory limit required by the industry partner, our heuristic improves the objective values of the best-known solutions by 18.6% on average, highlighting its efficiency for large-scale telecommunications networks.

Collectively, the studies presented in this thesis advance the state-of-the-art in routing and network design problems by introducing innovative algorithms and optimization techniques tailored to the computational challenges. By addressing issues such as multitrip scheduling, solution space reduction, and survivability in large-scale networks, these contributions provide effective and scalable solutions to complex real-world problems. These advancements are crucial for improving the efficiency, scalability, and reliability of modern network infrastructure.

Keywords: Vehicle routing; Multitrip; Variable fixing; Trip elimination; Lagrangian dual; Optimality cut; Domain reduction; Arc fixing; Telecommunications; Traffic grooming; Double-link failure; Constructive heuristic; Parallel computing.

Publications and Working Papers

Arising from the Thesis

1. Silong Zhang, Jun Xia, and Zhou Xu, "An Enhanced Exact Algorithm for the Capacitated Multitrip Vehicle Routing Problem with Time Windows Using Trip-Based Formulation", to be submitted.

2. Silong Zhang, Jun Xia, and Zhou Xu, "A Note on Lagrangian Underestimate Cuts and Domain Reduction Techniques for Integer Programs", manuscript submitted to *Computers & Operations Research*.

3. Silong Zhang, Jixuan Feng, Junyan Liu, Yu Liu, Zhou Xu, and Fan Zhang, "An Efficient Hierarchical Constructive Heuristic for Large-Scale Survivable Traffic Grooming Problem under Double-Link Failures", manuscript submitted to *Computers & Operations Research*.

Others

1. Ying Yang, Silong Zhang, Shuaian Wang, "Integrated cruise fleet deployment and itinerary scheduling problem: An enhanced Benders decomposition approach", in *Transportation Research Part B: Methodological* (2025).

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to my chief supervisor, Prof. Zhou Xu, for his invaluable guidance, unwavering support, and constant encouragement throughout my PhD journey. His profound academic insights, rigorous research attitude, and dedication to excellence have deeply inspired me. It has been both an honor and a privilege to learn from him. I am especially thankful for the opportunities he provided, as well as for his mentorship, which went beyond academic guidance to offer profound insights into both life and career development.

I would like to extend my heartfelt thanks to my master's supervisor, Prof. Jun Xia, for inspiring me to pursue research and for laying the foundation of my academic career. His mentorship during my master's studies instilled in me the confidence and passion for research that guided me through my PhD.

My gratitude also goes to the professors at the Department of Logistics and Maritime Studies (LMS) for their inspiring lectures and insightful discussions, particularly Prof. Miao Song, Prof. Yulan Wang, Prof. Pengfei Guo, and Prof. Li Jiang. Their courses broadened my horizons and provided essential tools for my research. Furthermore, I want to thank our department and university for the financial support and office facilities that created a supportive research environment.

I would like to extend my heartfelt gratitude to the board of examiners for their time, effort, and invaluable feedback during the evaluation of my thesis. My sincere thanks go to the external examiners, Prof. Lei Zhao and Prof. Qiaochu He, for their insightful comments and constructive suggestions. I am deeply grateful to Prof. Miao Song, the chair of the board of examiners, for her guidance and support throughout the examination process. I am truly honored to have the opportunity to benefit from your review and advice.

I have been fortunate to work alongside a group of talented and supportive colleagues in my research group. I would like to thank Dr. Shifu Xu, Dr. Shengnan Shu, Mr. Xiaodeng Hao, Ms. Yarui Zhang, Mr. Yu Liu, and Mr. Jixuan Feng. Our discussions, collaborative efforts, and shared experiences in the lab have made this challenging journey much more enjoyable and memorable.

I would also like to thank my fellow classmates in LMS: Mr. Shiyi Jiang, Mr. Bin Tian, Ms. Yu Guo, Ms. Jingwen Xu, Ms. Ying Yang, Ms. Yilu Long, Ms. Fangzhe Wang, and Dr. Fuzhen Liu. Thanks for the conversations and support during stressful times. Your friendship has been a source of comfort and motivation.

I dedicate my heartfelt thanks to my family, especially my parents, for their unconditional love, unwavering support, and endless encouragement throughout my academic journey. I hereby give you my genuine blessing.

Finally, I would like to express my special gratitude to my girlfriend, Ms. Yunyun Zeng. Thank you for your understanding, patience, and companionship. Your presence has brought light and joy to my life, and your encouragement has kept me going even during the most difficult moments of my research. You have been my unwavering source of happiness, strength, and stability. I am deeply grateful for your love and for walking this path by my side.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CMTVRPTW capacitated multitrip vehicle routing problem with time windows

VRPTW vehicle routing problem with time windows

CVRP capacitated vehicle routing problem

LUC Lagrangian underestimate cut

LP linear programming

IP integer programming

ILP integer linear programming

RCF reduced cost fixing

RCC reduced cost cut

STG1 survivable traffic grooming problem under single-link failures

STG2 survivable traffic grooming problem under double-link failures

ESPPRC elementary shortest path problem with resource constraints

TOPTW team orienteering problem with time windows

RSFC relaxed structure feasibility cut

RSSFC relaxed superstructure feasibility cut

OTN optical transport network

RWA routing and wavelength assignment

WDM wavelength division multiplexing

LBAG link-bundled auxiliary graph

PAL protection at the lightpath level

PAC protection at the connection level

FPP full path protection

PPP partial path protection

LBAG link-bundled auxiliary graph

Chapter 1

Introduction

1.1 Background

In the contemporary era, the global economy and daily life rely significantly on the efficiency and reliability of complex network systems. These systems play a fundamental role in ensuring the smooth transfer of resources, services, and information across vast distances, forming the backbone of many essential activities. Among these, logistics networks and telecommunications networks stand out as two critical components that underpin the infrastructure of modern society, facilitating the movement of goods and the flow of information [Jalal et al., 2022, Crainic et al., 2022, Zheng et al., 2019].

Effective routing in logistics networks and the resilient design of telecommunications networks are essential for ensuring efficiency in an increasingly interconnected world. In logistics, the rapid growth of e-commerce has made vehicle routing optimization crucial for minimizing operational costs and ensuring timely deliveries, both of which are essential for maintaining competitiveness within complex supply chains [Konstantakopoulos et al., 2022]. Similarly, in telecommunications, the

strategic design of optical networks is vital for accommodating the exponential increase in data traffic, ensuring the high capacity and reliability needed to support services [Crainic et al., 2022]. Ultimately, optimization in both domains focuses on maximizing the efficient use of limited resources, whether vehicles or bandwidth, thereby sustaining and enhancing economic productivity.

Optimizing these networks involves addressing fundamental problems in routing and network design. In logistics, the routing problem is far more than finding the shortest path. It must satisfy specific constraints, such as vehicle capacity, customer time windows, and multitrip scheduling [Paradiso et al., 2020]. These constraints, combined with the growing scale of logistics networks, make the problem computationally difficult. Similarly, telecommunications networks face significant optimization problems as they grow to encompass thousands of nodes while accommodating ever-increasing traffic demands. An additional critical issue is ensuring network survivability in the face of potential failures [Blanco et al., 2023]. This requires designing a robust network topology capable of maintaining uninterrupted service, even in scenarios where multiple links fail simultaneously.

To address these routing and network design problems, a diverse range of solution methodologies has been proposed in the literature. On the one hand, exact algorithms have been extensively developed to guarantee optimal solutions. Notable approaches in this category include branch-and-cut [Lysgaard et al., 2004], branch-and-price [Desrochers et al., 1992, Yıldız and Karaşan, 2017], column enumeration [Baldacci et al., 2008], and Benders decomposition [Vignac et al., 2016]. These methods are particularly effective for smaller problem instances or when exact solutions are required. On the other hand, due to the NP-hard nature of these problems, heuristic and metaheuristic algorithms are widely employed to obtain high-quality solutions within practical time limits. Popular methods in this category

include the savings heuristic [Clarke and Wright, 1964], genetic algorithms [Holland, 1992], tabu search [Glover, 1986], simulated annealing [Kirkpatrick et al., 1983], and adaptive large neighborhood search [Ropke and Pisinger, 2006]. These approaches are often favored for their ability to handle large-scale instances and their flexibility in adapting to problem-specific constraints.

Despite the variety of algorithms discussed above, solving these routing and network design problems remains a considerable challenge. From a computational complexity perspective, these problems are typically classified as NP-hard. As the scale of the problem increases (e.g., with a growing number of network nodes) or as additional requirements are introduced (e.g., multitrip scheduling in logistics or survivability to double-link failures in telecommunications), the solution space expands significantly, leading to a dramatic rise in computational difficulty. The combinatorial explosion imposes significant challenges, often pushing existing algorithms difficult to obtain optimal or even high-quality solutions within practical computing resources.

1.2 Key Challenges in Routing and Network Design

While the NP-hard nature of routing and network design problems accounts for their inherent computational difficulty, the specific barriers to efficient optimization are multifaceted. This thesis addresses three critical challenges in this domain. First, multitrip scheduling, a practical operational requirement in vehicle routing, can significantly increase the number of decision variables and constraints, making the problems more challenging to solve. Second, the vast search space characteristic of integer programs like vehicle routing often renders standard pruning techniques insufficient, necessitating more effective methods for eliminating non-optimal solutions. Finally, in telecommunications network design, the challenge is compounded by issues of scalability and survivability: the large number of network nodes and

survivability requirements against double-link failures introduce complexities that traditional solution methods struggle to manage.

The first challenge concerns multitrip scheduling in vehicle routing with time windows, formally known in the literature as the capacitated multitrip vehicle routing problem with time windows (CMTVRPTW) [Paradiso et al., 2020]. This problem extends the classical vehicle routing problem with time windows (VRPTW) [Desrochers et al., 1992] by allowing vehicles to perform multiple trips within the planning horizon. Specifically, after serving a sequence of customers and returning to the depot, a vehicle may be replenished and dispatched again. While this extension improves vehicle utilization and aligns with the operational requirements of many real-world logistics systems, it significantly increases computational challenge by introducing a larger number of decision variables and constraints compared to the VRPTW.

This challenge is evident in the mathematical formulations commonly employed to solve the CMTVRPTW, which are typically set-partitioning-like formulations, where columns represent routes, superstructures, journeys, or trips. The route-based formulation [Paradiso et al., 2020] and the superstructure-based formulation [Yang, 2023, 2025] suffer from an exponential growth in the number of constraints required to enforce the fleet size limit. Moreover, the separation problems for identifying these constraints are NP-hard, further complicating the solution process. Alternatively, journey-based formulations [Azi et al., 2010, Şahin and Yaman, 2022], where a journey is defined as a sequence of routes performed by a single vehicle, avoid the exponential growth in constraints but lead to an explosion in the number of decision variables, adding another layer of difficulty. Finally, trip-based formulations [Hernandez et al., 2014, 2016], where a trip is defined as a specific route paired with a specific departure time, offer the benefit of solving the

separation problem in polynomial time. However, these formulations incorporate a larger number of variables compared to route-based formulations, complicating the solution process.

The second challenge concerns the effective reduction of the solution search space for integer programs, particularly in the context of routing problems. This issue is critical due to the combinatorial nature of these problems, where the search space expands exponentially with increasing network size. To address this, valid cuts and domain reduction techniques have been extensively studied in the literature to reduce the search space. However, despite their effectiveness, these approaches are still subject to certain limitations, which constrain their broader applicability.

Valid cuts, such as subset-row cuts [Jepsen et al., 2008] and reduced cost cuts (RCCs) [Yang, 2023], are employed to prune the search space by removing fractional or suboptimal solutions from consideration. However, feasibility cuts (e.g., subset-row cuts) are often problem-specific, requiring detailed structural insights, which limits their applicability to broader problem classes. Similarly, optimality cuts like RCCs depend on feasible linear programming (LP) duals, which further limit their scalability in large-scale or complex problem settings. Consequently, a key challenge is the development of cuts based on more general dual information, such as Lagrangian duals, which are independent of specific problem structures and require only the satisfaction of sign constraints.

Domain reduction techniques provide a complementary approach to reducing the search space. For example, reduced cost fixing (RCF) has been extensively utilized in eliminating decision variables by leveraging feasible LP duals [Hadjar et al., 2006, Baldacci et al., 2008], fixing nonnegative integer variables to zero if their reduced costs exceed the duality gap. Similarly, Yang [2025] introduces

domain reduction techniques that combine the use of Lagrangian and feasible LP duals. While these methods have demonstrated effectiveness, a key challenge lies in leveraging general duals, such as Lagrangian duals, to develop new domain reduction methods that are more flexible and not constrained by the need to compute feasible LP duals. In routing problems such as the VRPTW, existing domain reduction techniques for arc fixing rely on feasible LP duals obtained through column generation [Hadjar et al., 2006, Irnich et al., 2010, Desaulniers et al., 2020]. A challenge is to systematically exploit general dual information to increase the number of arcs that can be eliminated, thereby reducing the network size and the search space while preserving optimality.

The third challenge concerns the scalability and survivability in telecommunications network design. Driven by ever-increasing global telecommunications demands [Cisco, 2024], modern networks are expanding dramatically in size and complexity. For instance, our industry partner requires solutions for network design problems involving thousands of nodes, subject to numerous real-world operational constraints. Even without considering network failures, such design problems are NP-hard [Zhu and Mukherjee, 2002]. This complexity is further exacerbated by the requirement for survivability. As network sizes grow, the probability of simultaneous link failures, particularly double-link failures, has increased notably [Sasithong et al., 2020]. To ensure service reliability, modern telecommunications companies mandate that communication demands remain uninterrupted not only under normal operating conditions but also across all scenarios involving single or double link failures. The number of failure scenarios, which scales quadratically with the number of links, can reach millions for large-scale networks. As a result, the large network sizes and survivability requirements lead to a significant increase in the number of decision variables and constraints. Even for simplified problem settings, such as those that consider only a subset of practical operational constraints and

single-link failures, the network sizes studied in the literature typically involve no more than 300 nodes [Sebbah and Jaumard, 2012, Niu et al., 2024]. Optimizing such networks is essential to minimize facility costs and maintain service reliability, but the combination of large network sizes and survivability requirements presents a substantial computational challenge.

1.3 Thesis Outline

This thesis comprises three studies that address key challenges in routing and network design problems by developing new algorithms and optimization techniques to improve solution efficiency and scalability. The three critical challenges identified in the previous section are addressed in Chapters 2, 3, and 4, respectively. Specifically, the first study (Chapter 2) focuses on the multitrip scheduling challenge in vehicle routing by introducing an enhanced exact algorithm for the CMTVRPTW. The second study (Chapter 3) tackles the challenge of reducing the solution search space for integer programs by proposing novel optimality cuts and domain reduction techniques, including arc fixing methods for the vehicle routing problem. The third study (Chapter 4) addresses the survivability of large-scale telecommunications networks by developing an efficient heuristic for the survivable traffic grooming problem under double-link failures (STG2). These contributions leverage innovative exact algorithms, advanced domain reduction techniques, and efficient heuristic approaches, offering both practical solutions and theoretical advancements for complex routing and network design problems.

The first study (Chapter 2) investigates the CMTVRPTW, which extends the classical VRPTW by allowing vehicles to perform multiple trips within the planning horizon. It captures the practical requirements of many real-world applications, but it is significantly more challenging to solve. The current state-of-the-art exact algorithm employs a three-phase approach: (i) enumerating a set of candidate

routes that could appear in optimal solutions, (ii) constructing superstructures from these routes, and (iii) solving a superstructure-based integer programming (IP) formulation to obtain the optimal solution. Of these phases, (iii) is the most computationally intensive, as it involves solving constraint separation problems that are NP-hard. In this study, we enhance the three-phase exact algorithm for the CMTVRPTW. Based on the candidate routes generated in phase (i), our enhanced algorithm constructs a set of trips in phase (ii) and then solves a trip-based IP formulation in phase (iii). Compared to the route-based and superstructure-based formulations, the trip-based formulation allows for a more efficient algorithm to solve the constraint separation problem, thereby enhancing the efficiency of phase (iii). To further enhance the computational efficiency, we introduce a trip elimination method to reduce the number of variables and propose a dynamic time discovery strategy to reduce the number of constraints. Computational experiments demonstrate that our enhanced algorithm is highly effective for solving challenging instances of the CMTVRPTW. Among 14 benchmark instances previously unsolved or considered difficult in the literature, our method outperforms the state-of-the-art exact algorithm within a ten-hour time limit, solving 6 more instances to optimality and reducing the average runtime by over 20%.

The second study (Chapter 3) examines optimality cuts and domain reduction techniques aimed at reducing the solution search space for integer programs, with a particular focus on arc fixing methods for the vehicle routing problem. Integer programming formulations for combinatorial optimization problems, such as the vehicle routing problem, face significant computational challenges due to the exponential growth of the solution space. To address this, valid cuts and domain reduction techniques are commonly employed to prune the search area. In this study, we introduce Lagrangian underestimate cuts (LUCs), a novel class of optimality cuts, for integer programs. We demonstrate that LUCs can be utilized

to strengthen RCCs and to derive new domain reduction techniques. Moreover, we leverage LUCs to develop new arc fixing techniques for the vehicle routing problem, based on Lagrangian duals and the tailored dual-picking strategy. Computational experiments demonstrate that, compared to the feasible-LP-dual approach studied in the literature, our method achieves significant domain reductions: the number of remaining arcs is reduced to an average of 47.9%, while the number of enumerated routes decreases to an average of 31.3%.

The third study (Chapter 4) investigates a new and challenging network design problem, the STG2, in large-scale telecommunications networks comprising thousands of nodes. In this problem, each communication demand must be assigned a route for every possible scenario involving zero, one, or two failed fiber links. Ensuring protection against double-link failures is critical for maintaining reliable telecommunications services while minimizing equipment costs, a key priority for telecommunications providers. To the best of our knowledge, this problem remains unexplored in the existing literature, with current research primarily focusing on simpler problem settings involving smaller networks, typically with no more than 300 nodes. To address this gap, we propose a novel hierarchical constructive heuristic for the large-scale STG2 problem. The heuristic constructs and assigns routes to communication demands across different scenarios by following a hierarchical sequence. It incorporates various techniques, including demand aggregation, a reuse-then-search routing strategy, and parallel computing, to significantly enhance efficiency. Extensive experiments have been conducted on large-scale STG2 instances provided by an industry partner, encompassing networks with 1,000 to 2,600 nodes. The results demonstrate that within a one-hour time limit and a 16 GB memory limit required by the industry partner, our heuristic improves the objective values of the best-known solutions by 18.6% on average, highlighting its efficiency for large-scale telecommunications networks.

The remainder of this thesis is organized as follows: Chapter 2 presents an enhanced exact algorithm for the CMTVRPTW. Chapter 3 explores optimization techniques for reducing the solution search space of integer programs, with a focus on applications to the vehicle routing problem. Chapter 4 introduces an efficient heuristic for solving large-scale STG2. Finally, Chapter 5 concludes the thesis and outlines potential directions for future research.

Chapter 2

Utilizing Trip-Based Formulation for Solving Capacitated Multitrip Vehicle Routing Problem with Time Windows

2.1 Introduction

This chapter addresses the first challenge outlined in Chapter 1: multitrip scheduling in vehicle routing. This feature gives rise to the CMTVRPTW, which has recently attracted significant attention in studies such as Paradiso et al. [2020], Yang [2023, 2025]. While this problem closely aligns with operational requirements of many logistics networks [Paradiso et al., 2020], the inclusion of multitrip scheduling significantly increases the computational complexity of vehicle routing. To address this, we propose an enhanced exact algorithm to improve the solution efficiency.

The CMTVRPTW is a variant of the capacitated vehicle routing problem

(CVRP) [Dantzig and Ramser, 1959]. The CVRP seeks to determine a set of minimum-cost routes for a fleet of capacitated vehicles to serve a group of customers, ensuring that each customer is visited exactly once. However, in real-world scenarios such as city logistics and last-mile delivery, additional operational constraints often emerge. These include the need to visit customers within specified time windows and the requirement for vehicles to perform multiple trips due to fleet size limitations. These practical considerations result in the CMTVRPTW, a significantly more complex and computationally challenging variant of the CVRP.

The CMTVRPTW can be formally described as follows. Let $\mathcal{N} = \{1, 2, \ldots, |\mathcal{N}|\}$ represent the set of customers, with 0 denoting the depot. Define the set of nodes as $\mathcal{N}_0 = \{0\} \cup \mathcal{N}$, and the set of arcs as $\mathcal{A} = \{(i_1, i_2) : i_1, i_2 \in \mathcal{N}_0, i_1 \neq i_2\}$. Each customer $i \in \mathcal{N}$ is associated with a demand quantity q_i, a service time st_i, and a time window $[a_i, b_i]$. If a vehicle arrives at customer i before time a_i, it must wait until a_i to begin service. If it arrives within $[a_i, b_i]$, the service starts immediately. The planning horizon is defined as $[a_0, b_0]$, representing the earliest departure time and latest return time at the depot. Each arc $(i_1, i_2) \in \mathcal{A}$ is associated with a travel cost $c_{i_1 i_2}$ and a travel time $t_{i_1 i_2}$. As demonstrated in Appendix A.1, all time data (travel times, service times, and endpoints of time windows) can be assumed to be integers without loss of generality.

A fleet of K homogeneous vehicles, each with a capacity of Q, is available to serve the customers. Each vehicle is allowed to perform multiple trips within the planning horizon. The objective of the CMTVRPTW is to minimize the total travel cost while ensuring that each customer is visited exactly once. Figure 2.1 presents an illustrative example in which one vehicle performs two trips, while the other performs a single trip.

Extensive variants of the CMTVRPTW that incorporate additional practical

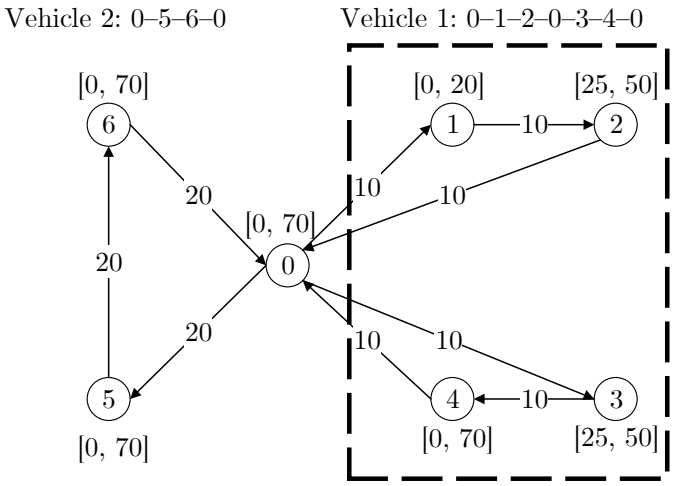


Figure 2.1: A feasible solution to a CMTVRPTW instance is presented for the case where $K = 2$, $Q = 2$, and $q_i = 1$ for all $i \in \mathcal{N}$. The time windows are indicated above the nodes, while the travel times are specified on the arcs. Additionally, the travel cost for all arcs is uniformly set to 1. The total cost of the solution is 9. In this solution, vehicle 1 performs two trips, while vehicle 2 performs a single trip.

constraints have been widely studied in the literature [Hernandez et al., 2014, 2016, Cattaruzza et al., 2016, Cheng et al., 2020]. These problems share several key characteristics with the CMTVRPTW, including a limited fleet size, the allowance for multiple trips, a single visit to each customer, and time window constraints.

Exact solution methods for the CMTVRPTW and its variants typically employ four types of set-partitioning-like formulations, where the columns represent routes, superstructures, trips, and journeys, respectively. Specifically, a *route* is a simple cycle that starts and ends at the depot (e.g., in Figure 2.1, $r_1 = (0, 1, 2, 0)$ and $r_2 = (0, 3, 4, 0)$ represent two routes). A *superstructure* is a set of routes that satisfy the following conditions: they visit the same set of customers, they have the same cost and duration, and the union of their departure time intervals is an interval (e.g., in Figure 2.1, $\{r_3, r_4\}$ is a superstructure where $r_3 = (0, 5, 6, 0)$ and $r_4 = (0, 6, 5, 0)$). A *trip* is defined as a pair of route and the departure time from the depot (e.g., in Figure 2.1, $(r_1, 5)$ represents a trip). A *journey* is a sequence

of routes that can be performed by a vehicle within the planning horizon (e.g., in Figure 2.1, (r_1, r_2) represents a journey).

In the literature, the state-of-the-art exact algorithm [Yang, 2025] for the CMTVRPTW employs a three-phase approach. In the first phase, it solves the LP relaxation of the route-based formulation and enumerates a set of candidate routes that may appear in optimal solutions. In the second phase, it constructs superstructures by combining the enumerated routes that satisfy specific conditions. In the third phase, it solves a superstructure-based IP formulation to obtain the optimal solution. While this algorithm has successfully solved instances with up to 200 customers, the final phase can be computationally expensive, as it involves solving NP-hard constraint separation problems.

In this study, we enhance the three-phase exact algorithm for the CMTVRPTW by leveraging the trip-based formulation, which enables a more efficient algorithm for the constraint separation problem compared to the superstructure-based formulation. The first phase, which enumerates a set of candidate routes, remains the same as in the state-of-the-art algorithm. In the second phase, we construct a set of trips that may appear in optimal solutions based on the enumerated routes. In the third phase, we solve a trip-based IP formulation to obtain an optimal solution to the CMTVRPTW. To further improve computational efficiency, we introduce a trip elimination method to reduce the solution space and a dynamic time discovery strategy to reduce the number of identified time points. Computational results demonstrate that this enhanced algorithm outperforms the state-of-the-art exact algorithm [Yang, 2025] on challenging benchmark instances.

The key contributions of this study are summarized as follows:

- We demonstrate how trip-based formulation can be utilized to enhance the three-phase exact algorithm for the CMTVRPTW, outperforming the state-

of-the-art algorithm on challenging benchmark instances.

- We demonstrate that the optimality of the trip-based formulation is preserved when restricting consideration to trips with integer departure times. This restriction reduces the formulation to a finite number of variables and constraints, enabling the trip-based formulation to be solved directly using IP solvers.

- We propose to eliminate trips by utilizing the dominance rule, variable fixing, and the optimality property of vehicle departure times. The trip elimination method significantly reduces the number of variables without compromising optimality.

- We introduce a dynamic time discovery strategy to reduce the number of identified time points. This strategy accelerates the computation of formulations by significantly decreasing the number of constraints.

- We conduct extensive computational experiments on CMTVRPTW benchmark instances with up to 200 customers. Computational results show that our method is highly effective in solving challenging instances of the CMTVRPTW. Among 14 benchmark instances previously unsolved or considered difficult in the literature, our method outperforms the state-of-the-art exact algorithm within a ten-hour time limit, solving 6 more instances to optimality and reducing the average runtime by over 20%. Furthermore, the computational results reveal that our algorithm eliminates over 93% of the trips, highlighting its effectiveness in reducing the solution space.

The remainder of this chapter is organized as follows: Section 2.2 reviews exact algorithms for solving the CMTVRPTW and its related variants. Section 2.3 presents the mathematical formulations of the problem, while Section 2.4 describes

the proposed solution method in detail. Computational results are analyzed in Section 2.5. Finally, the chapter is summarized in Section 2.6.

2.2 Literature Review

In this section, we review the exact algorithms developed for the CMTVRPTW and its variants. For more comprehensive surveys on multitrip vehicle routing problems, readers are referred to Mingozzi et al. [2013], Cattaruzza et al. [2018], Yang [2023].

2.2.1 Algorithms Leveraging Journey-Based Formulations

Azi et al. [2007] investigate a variant of the CMTVRPTW in which a single vehicle selectively serves customers. This problem accounts for loading times at the depot and imposes a limit on trip duration. To solve this problem, they propose a two-phase exact algorithm. In the first phase, a set of non-dominated routes is enumerated, while the second phase uses dynamic programming to identify the optimal journey. The effectiveness of the method is evaluated on instances adapted from Solomon [1987], with results showing that the performance is sensitive to the trip duration limit.

Azi et al. [2010] extend the work of Azi et al. [2007] by involving multiple homogeneous vehicles. They propose a journey-based set-packing formulation and develop a branch-and-price algorithm to solve it. The pricing problem is an elementary shortest path problem with resource constraints (ESPPRC) on an auxiliary route graph. However, the large number of enumerated routes makes the pricing problem computationally challenging, restricting the method to instances with up to 40 customers.

Şahin and Yaman [2022] explore a variant of the CMTVRPTW that involves multiple depots and heterogeneous vehicles. They propose a branch-and-price

algorithm for a journey-based formulation, capable of solving instances with up to 40 customers, three depots, and two vehicle types. Roboredo et al. [2023] address the journey-based formulation of the CMTVRPTW using VRPSolver [Pessoa et al., 2020], a state-of-the-art solver that implements an advanced branch-cut-and-price algorithm designed for vehicle routing. Their approach successfully solves instances with up to 100 customers.

2.2.2 Algorithms Leveraging Route-Based and Superstructure-Based Formulations

Paradiso et al. [2020] develop a route-based IP formulation for the CMTVRPTW, where the separation of side constraints, ensuring that the selected routes can be performed by the available vehicles within the planning horizon, corresponds to solving a *team orienteering problem with time windows* (TOPTW), which is NP-hard. They tackle the route-based formulation using a price-cut-and-enumerate method. First, the LP relaxation is solved via column generation to obtain a feasible dual solution. Next, candidate routes with reduced costs no greater than the duality gap are enumerated using a variable fixing technique [Baldacci et al., 2008]. Valid cuts are then introduced to reduce the duality gap and further shrink the candidate route set. Finally, the route-based IP formulation, restricted to the candidate routes, is solved using an IP solver, where side constraints are incorporated as lazy constraints. This approach successfully solves instances with up to 50 customers, and has been further adapted to address four variants of the CMTVRPTW. Note that the term "structure" used in Paradiso et al. [2020] is synonymous with the term "route" commonly used in the literature (e.g., Azi et al. [2007]), which refers to a node sequence. Furthermore, according to Azi et al. [2007], all attributes of a "structure" (including the earliest departure time, latest departure time, and minimum duration) are entirely determined by the node sequence.

Yang [2023] introduces a superstructure-based IP formulation for the CMTVRPTW and develops a three-phase exact algorithm to solve it, where the separation of side constraints involves solving the NP-hard TOPTW. The algorithm follows three phases: the first phase enumerates a set of candidate routes that may appear in optimal solutions, the second phase constructs superstructures by combining the enumerated routes that satisfy specific conditions, and the third phase solves the superstructure-based IP formulation to obtain the optimal solution. Compared to the method of Paradiso et al. [2020], it makes several key advancements. First, valid cuts are incorporated into the route-based LP formulation prior to enumeration, yielding tighter dual bounds and reducing out-of-memory errors during enumeration. Second, an auxiliary LP formulation is solved to obtain a dual solution that maximizes the total reduced cost of candidate routes, enabling further route elimination. Third, candidate routes that satisfy specific conditions are combined into superstructures, reducing the number of columns in the formulation. With these advancements, Yang [2023] successfully solves instances with up to 100 customers, outperforming the method proposed by Paradiso et al. [2020].

Yang [2025] enhances the three-phase exact algorithm of Yang [2023] by the "Deluxing" method. A key innovation is a novel variable fixing technique, where the dual solution used to eliminate routes only needs to ensure that a subset of candidate routes has nonnegative reduced costs. This makes obtaining such dual solutions easier than feasible duals. Given that the reduced cost of a variable can vary significantly across different dual solutions, Yang [2025] performs a deep search to obtain multiple duals. Additionally, three techniques are introduced to accelerate the solution process: (i) A variable relaxation technique that relaxes a subset of integer variables into continuous variables; (ii) An efficient method for separating valid cuts; (iii) An effective primal heuristic to improve the upper bound. With a time limit of 3 hours, Yang [2025] successfully solves instances with up to

200 customers.

The route-based and superstructure-based formulations involve fewer variables than the trip-based and journey-based formulations, enabling their solution methods to handle instances with up to 200 customers. However, the number of their constraints to enforce the fleet size limit increases exponentially with the problem size. Additionally, separating these constraints requires solving the NP-hard TOPTW, which can be computationally expensive.

2.2.3 Algorithms Leveraging Trip-Based Formulations

Hernandez et al. [2014] study the CMTVRPTW with limited trip duration and propose a trip-based formulation. The fleet size constraints ensure that, for each unit-length time interval $[w, w+1]$, where $w \in \{a_0, a_0+1, \ldots, b_0-1\}$, the number of vehicles *active* at all time points within $[w, w+1]$ does not exceed the fleet size K. A vehicle is considered *active* if it is traveling between nodes or serving/waiting at a customer. To ensure that the number of vehicles active over the entire interval $[w, w+1]$ is equal to the number of vehicles active at any specific time point within $[w, w+1]$, Hernandez et al. [2014] restrict their formulation to trips with integer departure times. However, whether this restriction preserves optimality is not discussed in Hernandez et al. [2014].

Solving the LP relaxation while incorporating fleet size constraints for all unit-length time intervals can be computationally intensive. To enhance computational efficiency, Hernandez et al. [2014] propose an "equidistant-and-halving" strategy to replace unit-length time intervals with broader intervals, thereby reducing the number of fleet size constraints. Specifically, the planning horizon $[a_0, b_0]$ is partitioned into the interval $[a_0 + \lfloor \frac{b_0-a_0}{\xi} \rfloor * \xi, b_0]$ and the set of intervals $\{[a_0 + k * \xi, a_0 + (k+1) * \xi] : k \in \{0, 1, \ldots, \lfloor \frac{b_0-a_0}{\xi} \rfloor - 1\}\}$, where ξ is initialized as a predefined parameter. This substitution results in a relaxation of the original LP.

If the optimal solution to this relaxed problem is infeasible for the original LP, the value of ξ is halved, and the process is repeated. Using this strategy, their branch-and-price algorithm successfully solves instances with up to 40 customers.

Hernandez et al. [2016] investigate the CMTVRPTW with loading times. They adopt the same trip-based formulation as Hernandez et al. [2014], but without imposing a duration limit on trips. The initial partition of the planning horizon is identical to that in Hernandez et al. [2014], which also results in a relaxation of the original LP. If the optimal solution to this relaxed problem is infeasible for the original LP, they do not reduce the value of ξ. Instead, they aim to find a feasible solution by solving a journey-based formulation, where journeys are constructed by concatenating trips generated during the column generation process. Their branch-and-price algorithm successfully identifies feasible solutions for instances with up to 25 customers.

Huang et al. [2024a] and Huang et al. [2024b] study two variants of the CMTVRPTW. Following the approach introduced by Hernandez et al. [2014], they employ the "equidistant-and-halving" strategy to reduce the number of fleet size constraints. The branch-and-price algorithms proposed by Huang et al. [2024a] and Huang et al. [2024b] successfully solve instances with up to 50 and 100 customers, respectively.

Paradiso et al. [2020] introduce a trip-based formulation for modeling the CMTVRPTW. In this formulation, fleet size constraints are defined based on specific time points rather than time intervals, ensuring that the number of vehicles active at any time point within $[a_0, b_0]$ does not exceed K. Additionally, the departure times of trips are not restricted to integer values. However, this formulation introduces an infinite number of variables and constraints, and the authors do not propose a solution method to address it.

While branch-and-price algorithms have been proposed for solving variant problems of the CMTVRPTW by utilizing trip-based formulations, their scalability is limited to instances with up to 100 customers. The primary challenge lies in the large number of variables and constraints in trip-based formulations. In this study, we propose an enhanced exact algorithm to solve the trip-based formulation of the CMTVRPTW. We demonstrate that optimality is preserved when restricting consideration to trips with integer departure times. Leveraging this property, we obtain a trip-based reformulation with a finite number of variables and constraints, ensuring optimality while enabling the formulation to be solved directly using IP solvers. This eliminates the need for solving pricing problems iteratively at non-root nodes. To further improve computational efficiency, we introduce the trip elimination method to reduce the solution space and a dynamic time discovery strategy to reduce the number of identified time points. Our approach successfully solves CMTVRPTW benchmark instances with up to 200 customers, including 6 instances that have not been solved to optimality in the existing literature.

2.3 Mathematical Formulations

In this section, we first present the route-based formulation in Section 2.3.1 and the superstructure-based formulation in Section 2.3.2, which are employed to solve the CMTVRPTW in the state-of-the-art algorithm. In Section 2.3.3, we present a trip-based formulation from the literature, and obtain a trip-based reformulation with fewer variables and constraints, which is utilized in our enhanced algorithm.

2.3.1 Route-Based Formulation

A route $r = (i_0 = 0, i_1, \ldots, i_{n_r}, i_{n_r+1} = 0)$ represents a simple cycle where a vehicle departs from the depot, visits customers $i_1, \ldots, i_{n_r}$ sequentially, and returns to the depot while satisfying the capacity constraint and time window constraints. The

associated travel cost is $c_r = \sum_{w=0}^{n_r} c_{i_w i_{w+1}}$. Let $\mathcal{R}$ denote the set of all routes. For $i \in \mathcal{N}$ and $r \in \mathcal{R}$, let α_{ir} be a binary parameter equal to 1 if and only if customer i is visited by route r. For a subset of routes $\bar{\mathcal{R}} \subseteq \mathcal{R}$, let $\phi(K, \bar{\mathcal{R}})$ represent the maximum number of routes in $\bar{\mathcal{R}}$ that can be performed by the K available vehicles within the planning horizon. The value of $\phi(K, \bar{\mathcal{R}})$ can be determined by solving the NP-hard TOPTW [Paradiso et al., 2020]. For $r \in \mathcal{R}$, let x_r be the binary variable equal to 1 if and only if route r is selected in the solution. The route-based formulation proposed in Paradiso et al. [2020] is then presented as follows.

$$(RF) \qquad \min \sum_{r \in \mathcal{R}} c_r x_r, \tag{2.1a}$$

$$\text{s.t.} \qquad \sum_{r \in \mathcal{R}} \alpha_{ir} x_r = 1, \quad \forall i \in \mathcal{N}, \tag{2.1b}$$

$$\sum_{r \in \bar{\mathcal{R}}} x_r \leq \phi(K, \bar{\mathcal{R}}), \quad \forall \bar{\mathcal{R}} \subseteq \mathcal{R}, \tag{2.1c}$$

$$x_r \in \{0, 1\}, \quad \forall r \in \mathcal{R}. \tag{2.1d}$$

The objective function (2.1a) minimizes the total travel cost. Constraints (2.1b) guarantee that each customer is visited exactly once. The side constraints (2.1c) enforce the fleet size limit, ensuring that the selected routes can be executed by the fleet of K vehicles within the planning horizon. Finally, constraints (2.1d) define the domain of the decision variables.

As shown in Azi et al. [2007], each route $r \in \mathcal{R}$ is associated with three attributes: e_r, l_r, and d_r. In this context, l_r denotes the latest departure time from the depot that ensures compliance with the time window constraints, while d_r represents the minimum duration required to perform route r (achieved by departing at l_r). The attribute e_r is the earliest non-dominated departure time, with the following properties: Departing at time $t \in [e_r, l_r]$ corresponds to returning

to the depot at time $t + d_r$; Departing at time $t \in [a_0, e_r)$ results in a return time of $e_r + d_r$, which is dominated by departing at e_r. Thus, $e_r + d_r$ is the earliest possible return time at the depot for route r. For example, consider the route $r_1 = (0, 1, 2, 0)$ in Figure 2.1, we have $e_{r_1} = 5$, $l_{r_1} = 10$, and $d_{r_1} = 30$. Moreover, as demonstrated in Appendix A.1, we have $\{e_r, l_r : r \in \mathcal{R}\} \subseteq \{a_0, a_0+1, \ldots, b_0\} \subseteq \mathbb{Z}_+$ and $\{d_r : r \in \mathcal{R}\} \subseteq \mathbb{Z}_{++}$, where $\mathbb{Z}_+$ is the set of nonnegative integers and $\mathbb{Z}_{++}$ is the set of positive integers.

Solving the LP relaxation of formulation (2.1) can be computationally expensive because separating constraints (2.1c) violated by the solution $\boldsymbol{x}$ requires solving the NP-hard TOPTW. This process becomes particularly time-consuming when $\boldsymbol{x}$ is fractional and the set $\{r \in \mathcal{R} : x_r > 0\}$ has a large cardinality. To mitigate this challenge, Paradiso et al. [2020] propose a weaker LP relaxation by replacing constraints (2.1c) with relaxed structure feasibility cuts (RSFCs). Specifically, for $t \in [a_0, b_0]$ and $r \in \mathcal{R}$, let $\beta_{tr} = \mathbb{I}_{[l_r, e_r+d_r)}(t)$, where $\mathbb{I}$ denotes the indicator function. β_{tr} equals 1 if and only if the vehicle performing route r is active at time t for any possible departure time from the depot. The RSFC associated with time point $t \in [a_0, b_0]$ ensures that the lower bound $\sum_{r\in\mathcal{R}} \beta_{tr} x_r$ on the number of vehicles active at time t does not exceed the fleet size K.

$$\text{(RSFC)} \qquad \sum_{r\in\mathcal{R}} \beta_{tr} x_r \leq K, \quad \forall t \in [a_0, b_0]. \tag{2.2}$$

The route-based LP relaxation is then formulated as follows.

$$(LRF) \qquad \min \sum_{r\in\mathcal{R}} c_r x_r, \tag{2.3a}$$

$$\text{s.t.} \qquad (2.1b), (2.2),$$

$$x_r \geq 0, \quad \forall r \in \mathcal{R}. \tag{2.3b}$$

2.3.2 Superstructure-Based Formulation

A superstructure is a set of routes that satisfy the following conditions: they visit the same set of customers, they have the same cost and duration, and the union of their departure time intervals is an interval. For example, routes $r_3 = (0, 5, 6, 0)$ and $r_4 = (0, 6, 5, 0)$ in Figure 2.1 can be combined into a superstructure $\{r_3, r_4\}$, since $\mathcal{N}(r_3) = \mathcal{N}(r_4)$ where $\mathcal{N}(r)$ represents the set of customers visited by route $r \in \mathcal{R}$, $c_{r_3} = c_{r_4} = 3$, $d_{r_3} = d_{r_4} = 60$, and $[e_{r_3}, l_{r_3}] \cup [e_{r_4}, l_{r_4}] = [20, 30] \cup [20, 30] = [20, 30]$ is an interval.

Given a set of routes $\mathcal{R}$, Yang [2023] proposes a method for constructing a set of superstructures $\mathcal{U}$ derived from $\mathcal{R}$. For superstructure $u \in \mathcal{U}$, denote $\tilde{\mathcal{N}}(u)$ as the set of visited customers, $\tilde{c}_u$ is the travel cost, $\tilde{d}_u$ is the duration, $[\tilde{e}_u, \tilde{l}_u]$ is the departure time interval. Consider the example in Figure 2.1, for superstructure $u = \{r_3, r_4\}$, we have $\tilde{\mathcal{N}}(u) = \mathcal{N}(r_3) = \{5, 6\}$, $\tilde{c}_u = c_{r_3} = 3$, $\tilde{d}_u = d_{r_3} = 60$, $[\tilde{e}_u, \tilde{l}_u] = [e_{r_3}, l_{r_3}] \cup [e_{r_4}, l_{r_4}] = [20, 30]$.

For $i \in \mathcal{N}$ and $u \in \mathcal{U}$, let $\tilde{\alpha}_{iu}$ be a binary parameter equal to 1 if and only if customer i is visited by superstructure u. For a subset of superstructures $\bar{\mathcal{U}} \subseteq \mathcal{U}$, let $\tilde{\phi}(K, \bar{\mathcal{U}})$ represent the maximum number of superstructures in $\bar{\mathcal{U}}$ that can be performed by the K available vehicles within the planning horizon. The value of $\tilde{\phi}(K, \bar{\mathcal{U}})$ can be determined by solving the NP-hard TOPTW [Yang, 2023]. For $u \in \mathcal{U}$, let $\tilde{x}_u$ be the binary variable equal to 1 if and only if superstructure u is selected in the solution. The superstructure-based formulation proposed in Yang [2023] is then presented as follows.

$$(UF) \qquad \min \sum_{u \in \mathcal{U}} \tilde{c}_u \tilde{x}_u, \tag{2.4a}$$

$$\text{s.t.} \qquad \sum_{u \in \mathcal{U}} \tilde{\alpha}_{iu} \tilde{x}_u = 1, \quad \forall i \in \mathcal{N}, \tag{2.4b}$$

$$\sum_{u\in\bar{\mathcal{U}}} \tilde{x}_u \le \tilde{\phi}(K,\bar{\mathcal{U}}), \quad \forall \bar{\mathcal{U}} \subseteq \mathcal{U}, \tag{2.4c}$$

$$\tilde{x}_u \in \{0,1\}, \quad \forall u \in \mathcal{U}. \tag{2.4d}$$

The objective function (2.4a) minimizes the total travel cost. Constraints (2.4b) guarantee that each customer is visited exactly once. The side constraints (2.4c) enforce the fleet size limit, ensuring that the selected superstructures can be executed by the fleet of K vehicles within the planning horizon. Finally, constraints (2.4d) define the domain of the decision variables.

Solving the LP relaxation of formulation (2.4) can be computationally expensive because separating constraints (2.4c) violated by the solution $\tilde{\boldsymbol{x}}$ requires solving the NP-hard TOPTW. This process becomes particularly time-consuming when $\tilde{\boldsymbol{x}}$ is fractional and the set $\{u \in \mathcal{U} : \tilde{x}_u > 0\}$ has a large cardinality. To mitigate this challenge, a weaker LP relaxation can be derived by replacing constraints (2.4c) with relaxed superstructure feasibility cuts (RSSFCs). Specifically, for $t \in [a_0, b_0]$ and $u \in \mathcal{U}$, let $\tilde{\beta}_{tu} = \mathbb{I}_{[\tilde{l}_u, \tilde{e}_u + \tilde{d}_u)}(t)$. Then $\sum_{u\in\mathcal{U}} \tilde{\beta}_{tu}\tilde{x}_u$ serves as a lower bound on the number of vehicles active at time t, which cannot exceed the fleet size K.

$$\text{(RSSFC)} \qquad \sum_{u\in\mathcal{U}} \tilde{\beta}_{tu}\tilde{x}_u \le K, \quad \forall t \in [a_0, b_0]. \tag{2.5}$$

The superstructure-based LP relaxation is then formulated as follows.

$$(LUF) \qquad \min \sum_{u\in\mathcal{U}} \tilde{c}_u\tilde{x}_u, \tag{2.6a}$$

$$\text{s.t.} \qquad (2.4b), (2.5),$$

$$\tilde{x}_u \ge 0, \quad \forall u \in \mathcal{U}. \tag{2.6b}$$

Let $\chi(LRF)$ and $\chi(LUF)$ represent the optimal objective values of formulations

LRF and LUF, respectively. Proposition 1 demonstrates that the LP relaxation LRF of the route-based formulation is at least as tight as the LP relaxation LUF of the superstructure-based formulation.

Proposition 1. $\chi(LRF) \geq \chi(LUF)$.

Proof. We establish this proposition by showing that for any optimal solution $\boldsymbol{x}$ to LRF, there exists a feasible solution $\tilde{\boldsymbol{x}}$ to LUF that achieves the same objective value.

By the definition of $\mathcal{U}$, for any route $r \in \mathcal{R}$, there exists a unique superstructure $u \in \mathcal{U}$ such that $\tilde{\mathcal{N}}(u) = \mathcal{N}(r)$, $\tilde{c}_u = c_r$, $\tilde{d}_u = d_r$, and $[\tilde{e}_u, \tilde{l}_u] \supseteq [e_r, l_r]$. For route $r \in \mathcal{R}$, let $u(r)$ denote the associated superstructure. We construct the solution $\tilde{\boldsymbol{x}}$ by setting $\tilde{x}_u = \sum_{r \in \mathcal{R}: u(r)=u} x_r$.

For $r \in \mathcal{R}$, we have $\tilde{\alpha}_{iu(r)} = \alpha_{ir}$ because $\tilde{\mathcal{N}}(u(r)) = \mathcal{N}(r)$. Consequently, for $i \in \mathcal{N}$, it follows that $\sum_{u \in \mathcal{U}} \tilde{\alpha}_{iu} \tilde{x}_u = \sum_{r \in \mathcal{R}} \alpha_{ir} x_r = 1$. This implies that $\tilde{\boldsymbol{x}}$ satisfies constraints (2.4b).

For $r \in \mathcal{R}$ and $t \in [a_0, b_0]$, we have $\tilde{\beta}_{tu(r)} \leq \beta_{tr}$ because $\tilde{d}_{u(r)} = d_r$ and $[\tilde{e}_{u(r)}, \tilde{l}_{u(r)}] \supseteq [e_r, l_r]$. Consequently, for $t \in [a_0, b_0]$, it follows that $\sum_{u \in \mathcal{U}} \tilde{\beta}_{tu} \tilde{x}_u \leq \sum_{r \in \mathcal{R}} \beta_{tr} x_r \leq K$. This implies that $\tilde{\boldsymbol{x}}$ satisfies constraints (2.5), and therefore, $\tilde{\boldsymbol{x}}$ is a feasible solution to LUF.

For $r \in \mathcal{R}$, we have $\tilde{c}_{u(r)} = c_r$. It follows that $\sum_{u \in \mathcal{U}} \tilde{c}_u \tilde{x}_u = \sum_{r \in \mathcal{R}} c_r x_r$, meaning that $\tilde{\boldsymbol{x}}$ has the same objective value with $\boldsymbol{x}$. This, together with that $\tilde{\boldsymbol{x}}$ is a feasible solution to LUF, completes the proof. □

2.3.3 Trip-Based Formulations

A trip $s = (r_s, \tau_s)$ is a route-time pair, where r_s represents the associated route and $\tau_s \in [e_{r_s}, l_{r_s}]$ denotes the departure time from the depot. Let $\mathcal{S} = \{(r, \tau) :$

$r \in \mathcal{R}, \tau \in [e_r, l_r]\}$ represent the set of all trips. For $t \in [a_0, b_0]$ and $s \in \mathcal{S}$, define $\gamma_{ts} = \mathbb{I}_{[\tau_s, \tau_s + d_{r_s})}(t)$, which equals 1 if and only if the vehicle performing trip s is active at time t. For $s \in \mathcal{S}$, let y_s be the binary variable equal to 1 if and only if trip s is selected in the solution. The trip-based formulation introduced in Paradiso et al. [2020] is then presented as follows.

$$
\begin{aligned}
(SF) \qquad & \min \sum_{s \in \mathcal{S}} c_{r_s} y_s, && && \text{(2.7a)} \\
\text{s.t.} \qquad & \sum_{s \in \mathcal{S}} \alpha_{ir_s} y_s = 1, && \forall i \in \mathcal{N}, && \text{(2.7b)} \\
& \sum_{s \in \mathcal{S}} \gamma_{ts} y_s \leq K, && \forall t \in [a_0, b_0], && \text{(2.7c)} \\
& y_s \in \{0, 1\}, && \forall s \in \mathcal{S}. && \text{(2.7d)}
\end{aligned}
$$

The objective function (2.7a) aims to minimize the total travel cost. Constraints (2.7b) ensure that each customer is visited exactly once, while constraints (2.7c) enforce that the number of active vehicles at any time does not exceed the fleet size K. Finally, constraints (2.7d) define the domain of the decision variables.

Proposition 2 implies that verifying whether a trip-based solution $\boldsymbol{y}$ satisfies constraint (2.7c) for all $t \in [a_0, b_0]$ can be reduced to checking whether it satisfies constraint (2.7c) for all $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$, where $\mathcal{S}(\boldsymbol{y}) = \{s \in \mathcal{S} : y_s > 0\}$.

Proposition 2. *Consider a trip-based solution $\boldsymbol{y}$. If $\sum_{s \in \mathcal{S}} \gamma_{ts} y_s \leq K$ holds for all $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$, then $\sum_{s \in \mathcal{S}} \gamma_{ts} y_s \leq K$ holds for all $t \in [a_0, b_0]$.*

Proof. For $s \in \mathcal{S}$, the function $\gamma_{ts} = \mathbb{I}_{[\tau_s, \tau_s + d_{r_s})}(t)$ is a step function that is right-continuous and increases only at $t = \tau_s$. Given the trip-based solution $\boldsymbol{y}$, where $y_s \geq 0$ for all $s \in \mathcal{S}$, it follows that $\sum_{s \in \mathcal{S}} \gamma_{ts} y_s = \sum_{s \in \mathcal{S}(\boldsymbol{y})} \gamma_{ts} y_s$ is also a step function. This step function is right-continuous and can potentially increase

only at $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$. Consequently, $\max\{\sum_{s\in\mathcal{S}} \gamma_{ts} y_s : t \in [a_0, b_0]\} = \max\{\sum_{s\in\mathcal{S}} \gamma_{ts} y_s : t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}\}$. Therefore, if $\sum_{s\in\mathcal{S}} \gamma_{ts} y_s \leq K$ holds for all $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$, then it must also hold for all $t \in [a_0, b_0]$. □

Given a trip-based solution $\boldsymbol{y}$ satisfying constraints (2.7b) and (2.7d), the separation problem for identifying a violated constraint (2.7c) can be solved in polynomial time. By Proposition 2, the separation problem can be solved by evaluating the value of $\sum_{s\in\mathcal{S}} \gamma_{ts} y_s$ for each $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$. Since $\boldsymbol{y}$ satisfies constraints (2.7b) and (2.7d), it follows that $|\mathcal{S}(\boldsymbol{y})| \leq |\mathcal{N}|$, $|\{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}| \leq |\mathcal{N}|$, and $\sum_{s\in\mathcal{S}} \gamma_{ts} y_s = \sum_{s\in\mathcal{S}(\boldsymbol{y})} \gamma_{ts}$. As a result, the separation problem can be solved in $O(|\mathcal{N}|^2)$ time.

In formulation (2.7), the set $\mathcal{S}$ can introduce an infinite number of variables. However, Proposition 3 demonstrates that it suffices to consider only integer departure times. Consequently, optimality is preserved if set $\mathcal{S}$ is replaced by $\mathcal{S}_0 = \{(r, \tau) : r \in \mathcal{R}, \tau \in [e_r, l_r] \cap \mathbb{Z}_+\}$.

Proposition 3. *For any feasible instance with integer time data, there exists an optimal solution where the departure times of all selected trips are integers.*

Proof. Consider an optimal solution $\boldsymbol{y}^1$ to SF. Let $\mathcal{S}(\boldsymbol{y}^1)$ denote the set of selected trips. If $\{\tau_s : s \in \mathcal{S}(\boldsymbol{y}^1)\} \subseteq \mathbb{Z}_+$, the statement above holds. Otherwise, define $\tau_1 = \min\{\tau_s : s \in \mathcal{S}(\boldsymbol{y}^1), \tau_s \notin \mathbb{Z}_+\}$ as the earliest fractional departure time, and let $s_1 = (r_1, \tau_1) \in \mathcal{S}(\boldsymbol{y}^1)$ denote the trip corresponding to this fractional departure time. Further, define $s_2 = (r_2, \tau_2)$, where $r_2 = r_1$ and $\tau_2 = \lfloor \tau_1 \rfloor$. Construct a new solution $\boldsymbol{y}^2$ such that $y^2_{s_1} = 0$, $y^2_{s_2} = 1$, and $y^2_s = y^1_s$ for all $s \in \mathcal{S} \setminus \{s_1, s_2\}$.

We now demonstrate that $\boldsymbol{y}^2$ is also an optimal solution. Since $r_2 = r_1$, it follows that $\boldsymbol{y}^2$ has the same objective value as $\boldsymbol{y}^1$, and $\boldsymbol{y}^2$ satisfies constraints (2.7b) for all $i \in \mathcal{N}$. By definition, we have $\gamma_{ts_1} = \mathbb{I}_{[\tau_1, \tau_1 + d_{r_1})}(t)$ and $\gamma_{ts_2} = \mathbb{I}_{[\tau_2, \tau_2 + d_{r_2})}(t)$.

Since $r_2 = r_1$ and $\tau_2 = \lfloor \tau_1 \rfloor$, it follows that $\gamma_{ts_1} \geq \gamma_{ts_2}$ for all $t \in [a_0, b_0] \setminus [\tau_2, \tau_1)$. As a result, $\boldsymbol{y}^2$ satisfies constraints (2.7c) for all $t \in [a_0, b_0] \setminus [\tau_2, \tau_1)$.

It remains to show that $\boldsymbol{y}^2$ satisfies constraints (2.7c) for all $t \in [\tau_2, \tau_1)$. Since τ_1 is the earliest fractional departure time, $\tau_2 = \lfloor \tau_1 \rfloor$, and $\{d_r : r \in \mathcal{R}\} \subseteq \mathbb{Z}_{++}$, it follows that $\gamma_{ts} \leq \gamma_{\tau_1 s}$ for all $s \in \mathcal{S}(\boldsymbol{y}^1)$ and $t \in [\tau_2, \tau_1)$. Consequently, for $t \in [\tau_2, \tau_1)$, we have $\sum_{s \in \mathcal{S}(\boldsymbol{y}^2)} \gamma_{ts} = 1 + \sum_{s \in \mathcal{S}(\boldsymbol{y}^2) \setminus \{s_2\}} \gamma_{ts} = 1 + \sum_{s \in \mathcal{S}(\boldsymbol{y}^1) \setminus \{s_1\}} \gamma_{ts} \leq 1 + \sum_{s \in \mathcal{S}(\boldsymbol{y}^1) \setminus \{s_1\}} \gamma_{\tau_1 s} = \sum_{s \in \mathcal{S}(\boldsymbol{y}^1)} \gamma_{\tau_1 s} \leq K$. This implies that $\boldsymbol{y}^2$ satisfies constraints (2.7c) for all $t \in [\tau_2, \tau_1)$. Therefore, $\boldsymbol{y}^2$ is also an optimal solution.

By iteratively applying the above procedure, we can construct an optimal solution in which the departure times of all selected trips are integers. □

In formulation (2.7), the planning horizon $[a_0, b_0]$ results in an infinite number of constraints in (2.7c), making it computationally impractical to solve this formulation directly by an IP solver. To handle this issue, we further replace $[a_0, b_0]$ in constraints (2.7c) with the discrete set $\mathcal{H} = \{a_0, a_0 + 1, \ldots, b_0\}$. Consequently, we obtain a trip-based reformulation (2.8) with a finite number of variables and constraints. The optimality of this reformulation is established in Proposition 4.

$$
\begin{aligned}
(SF(\mathcal{S}_0, \mathcal{H})) \qquad & \min \sum_{s \in \mathcal{S}_0} c_{r_s} y_s, && && \text{(2.8a)} \\
\text{s.t.} \qquad & \sum_{s \in \mathcal{S}_0} \alpha_{ir_s} y_s = 1, && \forall i \in \mathcal{N}, && \text{(2.8b)} \\
& \sum_{s \in \mathcal{S}_0} \gamma_{ts} y_s \leq K, && \forall t \in \mathcal{H}, && \text{(2.8c)} \\
& y_s \in \{0, 1\}, && \forall s \in \mathcal{S}_0. && \text{(2.8d)}
\end{aligned}
$$

Proposition 4. *If SF is feasible, then $SF(\mathcal{S}_0, \mathcal{H})$ must also be feasible. Moreover, any optimal solution to $SF(\mathcal{S}_0, \mathcal{H})$ is also optimal to SF.*

Proof. The formulation $SF(\mathcal{S}_0, \mathcal{H})$ is derived from SF by replacing $\mathcal{S}$ with $\mathcal{S}_0 = \{s \in \mathcal{S} : \tau_s \in \mathbb{Z}_+\}$ and substituting $[a_0, b_0]$ in constraints (2.7c) with $\mathcal{H} = \{a_0, a_0 + 1, \ldots, b_0\}$. By Proposition 3, it is sufficient to consider only integer departure times. Consequently, replacing $\mathcal{S}$ with $\mathcal{S}_0$ preserves optimality. Since $\mathcal{H} \subset [a_0, b_0]$, $SF(\mathcal{S}_0, \mathcal{H})$ is a relaxation of SF. Therefore, if SF is feasible, $SF(\mathcal{S}_0, \mathcal{H})$ must also be feasible.

Consider an optimal solution $\boldsymbol{y}$ to $SF(\mathcal{S}_0, \mathcal{H})$, which satisfies constraints (2.7c) for all $t \in \mathcal{H}$. Since $\mathcal{S}(\boldsymbol{y}) \subseteq \mathcal{S}_0$, it follows that $\{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\} \subseteq \mathcal{H}$. As a result, $\boldsymbol{y}$ satisfies constraints (2.7c) for all $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$. By Proposition 2, this implies that $\boldsymbol{y}$ satisfies constraints (2.7c) for all $t \in [a_0, b_0]$. Hence, $\boldsymbol{y}$ is feasible to SF. Since $SF(\mathcal{S}_0, \mathcal{H})$ is a relaxation of SF, and $\boldsymbol{y}$ achieves the same objective value in both $SF(\mathcal{S}_0, \mathcal{H})$ and SF, it follows that $\boldsymbol{y}$ is also optimal to SF. □

Reformulation (2.8) contains a finite number of variables and constraints, allowing it to be solved directly using an IP solver. This eliminates the need for solving pricing problems iteratively at non-root nodes.

Denote LSF as the LP relaxation of $SF(\mathcal{S}_0, \mathcal{H})$. Let $\chi(LRF)$ and $\chi(LSF)$ represent the optimal objective values of formulations LRF and LSF, respectively. Proposition 5 demonstrates that the LP relaxation LSF of the trip-based formulation is at least as tight as the LP relaxation LRF of the route-based formulation. For certain instances, the tighter LP relaxations enable the further elimination of columns and enhance the branch-and-bound algorithm for solving IP formulations.

Proposition 5. $\chi(LSF) \geq \chi(LRF)$.

Proof. We prove this proposition by demonstrating that, for any optimal solution $\boldsymbol{y}$ to LSF, there exists a feasible solution $\boldsymbol{x}$ to LRF with the same objective value. The solution $\boldsymbol{x}$ is contructed by setting $x_r = \sum_{s \in \mathcal{S}_0 : r_s = r} y_s$ for each $r \in \mathcal{R}$.

Since $\sum_{r\in\mathcal{R}} c_r x_r = \sum_{r\in\mathcal{R}} c_r \sum_{s\in\mathcal{S}_0:r_s=r} y_s = \sum_{r\in\mathcal{R}} \sum_{s\in\mathcal{S}_0:r_s=r} c_{r_s} y_s = \sum_{s\in\mathcal{S}_0} c_{r_s} y_s$, it follows that $\boldsymbol{x}$ has the same objective value as $\boldsymbol{y}$.

For $i \in \mathcal{N}$, we have $\sum_{r\in\mathcal{R}} \alpha_{ir} x_r = \sum_{r\in\mathcal{R}} \alpha_{ir} \sum_{s\in\mathcal{S}_0:r_s=r} y_s = \sum_{r\in\mathcal{R}} \sum_{s\in\mathcal{S}_0:r_s=r} \alpha_{ir_s} y_s = \sum_{s\in\mathcal{S}_0} \alpha_{ir_s} y_s = 1$. Thus $\boldsymbol{x}$ satisfies constraints (2.1b).

By Proposition 2, $\sum_{s\in\mathcal{S}_0} \gamma_{ts} y_s \leq K$ holds for all $t \in [a_0, b_0]$. Moreover, for $s \in \mathcal{S}_0$, since $e_{r_s} \leq \tau_s \leq l_{r_s}$, it follows that $\beta_{tr_s} = \mathbb{I}_{[l_{r_s}, e_{r_s}+d_{r_s})}(t) \leq \mathbb{I}_{[\tau_s, \tau_s+d_{r_s})}(t) = \gamma_{ts}$. Consequently, we have $\sum_{r\in\mathcal{R}} \beta_{tr} x_r = \sum_{r\in\mathcal{R}} \beta_{tr} \sum_{s\in\mathcal{S}_0:r_s=r} y_s = \sum_{r\in\mathcal{R}} \sum_{s\in\mathcal{S}_0:r_s=r} \beta_{tr_s} y_s = \sum_{s\in\mathcal{S}_0} \beta_{tr_s} y_s \leq \sum_{s\in\mathcal{S}_0} \gamma_{ts} y_s \leq K$. This implies that $\boldsymbol{x}$ satisfies constraints (2.2).

Therefore, $\boldsymbol{x}$ is a feasible solution to LRF with the same objective value as $\boldsymbol{y}$, completing the proof. □

2.4 Solution Algorithm

In this section, we present an enhanced exact algorithm for the CMTVRPTW, as outlined in Section 2.4.1. The dominance rule for trip elimination is detailed in Section 2.4.2. Section 2.4.3 describes the approach for solving the trip-based LP formulation to obtain dual solutions, which are then used to eliminate trips through the variable fixing technique. Finally, Section 2.4.4 introduces the method for solving the trip-based IP formulation to obtain the optimal solution.

2.4.1 Algorithm Outline

Algorithm 1 requires three inputs: An upper bound, a set of candidate routes (including those selected in optimal solutions), and a set of valid cuts. The upper bound is used to apply the variable fixing technique [Baldacci et al., 2008], the set of candidate routes is used to enumerate the initial set of trips, and the valid cuts strengthen the LP relaxation. These inputs can be obtained using route enumeration methods for the CMTVRPTW, such as that proposed by Yang [2025].

Algorithm 1 begins by enumerating the initial set of trips that includes those selected in optimal solutions (line 1), followed by the elimination of trips using the dominance rule (line 2). Subsequently, the trip-based LP relaxation LSF is solved to apply the variable fixing technique, enabling further elimination of trips (line 3). Finally, the trip-based IP formulation is solved to obtain the optimal solution (line 4).

Algorithm 1 Enhanced Exact Algorithm

Input: An upper bound UB on the optimal objective value; A set of routes $\mathcal{R}_1$ comprising those selected in optimal solutions; A set of valid cuts.
Output: An optimal solution to the CMTVRPTW instance.

1: Enumerate the initial set of trips $\mathcal{S}_1 = \{(r, \tau) : r \in \mathcal{R}_1, \tau \in \mathcal{G}_r^1\}$, where $\mathcal{G}_r^1 = [e_r, l_r] \cap \mathbb{Z}_+$ represents the initial set of departure times for route $r \in \mathcal{R}_1$.
2: Apply the dominance rule to eliminate trips from $\mathcal{S}_1$, and denote the set of remaining trips as $\mathcal{S}_2$.
3: Solve the trip-based LP relaxation LSF restricted to trips in $\mathcal{S}_2$. Apply the variable fixing technique to eliminate trips from $\mathcal{S}_2$, and denote the set of remaining trips as $\mathcal{S}_3$.
4: Solve the trip-based IP formulation, restricted to trips in $\mathcal{S}_3$, to obtain the optimal solution.

Algorithm 1 enhances the state-of-the-art exact algorithm [Yang, 2025] for the CMTVRPTW by leveraging the trip-based formulation. In this algorithm, the initial set of trips is constructed from the routes enumerated by Yang [2025], and the optimal solution is obtained by solving the trip-based IP formulation. Our enhanced algorithm offers several advantages. First, the separation problem for constraints (2.8c) in the trip-based IP formulation is solvable in polynomial time, and the number of constraints (2.8c) is pseudo-polynomial in the input. In contrast, the separation problem for constraints (2.4c) in the superstructure-based formulation requires solving NP-hard TOPTW problems, and the number of these constraints grows exponentially with the problem size. Second, the trip-based LP formulation yields tighter dual bounds, facilitating the elimination of additional routes and accelerating the branch-and-bound algorithm for obtaining the optimal

integer solution. Third, we reduce the number of variables by eliminating trips using the dominance rule, variable fixing technique, and optimality property of vehicle departure times. Finally, a dynamic time discovery strategy is employed during the solution of the LP and IP formulations to reduce the number of identified time points, thereby decreasing the number of constraints.

2.4.2 Trip Elimination by Dominance Rule

Consider two trips $s_1 = (r_1, \tau_1)$ and $s_2 = (r_2, \tau_2)$, where either $c_{r_1} < c_{r_2}$ or $d_{r_1} < d_{r_2}$. We say that s_1 dominates s_2 if, for any feasible solution in which s_2 is selected, another feasible solution with an equal or smaller objective value can be obtained by replacing s_2 with s_1. Proposition 6 provides sufficient conditions for dominance, where $\mathcal{N}(r)$ represents the set of customers visited by route $r \in \mathcal{R}$.

Proposition 6. *A trip $s_1 = (r_1, \tau_1)$ dominates another trip $s_2 = (r_2, \tau_2)$ if the following conditions are satisfied: (i) $\tau_1 = \tau_2$; (ii) $\mathcal{N}(r_1) = \mathcal{N}(r_2)$; (iii) $c_{r_1} \leq c_{r_2}$; (iv) $d_{r_1} \leq d_{r_2}$; (v) At least one of the inequalities in (iii) and (iv) holds strictly.*

Proof. We prove this proposition by demonstrating that if the trip $s_2 = (r_2, \tau_2)$ is selected in a feasible solution $\boldsymbol{y}^2$, then a new feasible solution $\boldsymbol{y}^1$ with an equal or smaller objective value can be obtained by replacing s_2 with the trip $s_1 = (r_1, \tau_1)$. Specifically, the solution $\boldsymbol{y}^1$ is constructed by setting $y^1_{s_1} = 1$, $y^1_{s_2} = 0$, and $y^1_s = y^2_s$ for all $s \in \mathcal{S}_0 \setminus \{s_1, s_2\}$.

Since $c_{r_1} \leq c_{r_2}$, it follows that $\boldsymbol{y}^1$ has an equal or smaller objective value compared to $\boldsymbol{y}^2$.

Furthermore, because $\mathcal{N}(r_1) = \mathcal{N}(r_2)$, we have $\alpha_{ir_1} = \alpha_{ir_2}$ for all $i \in \mathcal{N}$. Thus, $\boldsymbol{y}^1$ satisfies constraints (2.8b).

Since $\tau_1 = \tau_2$ and $d_{r_1} \leq d_{r_2}$, it follows that $\gamma_{ts_1} \leq \gamma_{ts_2}$ for all $t \in [a_0, b_0]$.

Consequently, $\boldsymbol{y}^1$ satisfies constraints (2.8c). Therefore, $\boldsymbol{y}^1$ is a feasible solution. This completes the proof. □

Dominated trips can be eliminated from $\mathcal{S}_1$ without compromising optimality. Let $\mathcal{S}_2$ represent the set of trips remaining after this elimination. Define $\mathcal{R}_2 = \{r \in \mathcal{R}_1 : \exists s \in \mathcal{S}_2 \text{ such that } r_s = r\}$. For each $r \in \mathcal{R}_2$, let $\mathcal{G}_r^2 = \{\tau \in \mathcal{G}_r^1 : (r, \tau) \in \mathcal{S}_2\}$.

2.4.3 Trip Elimination by Variable Fixing

In this section, we present a column-and-cut generation approach for solving LSF to obtain a feasible dual solution, which can be used to eliminate trips through variable fixing techniques. Additionally, as the reduced costs of a trip can vary significantly across different dual solutions, we introduce a dual-picking procedure to generate multiple trip-based dual solutions, further strengthening the trip elimination process.

2.4.3.1 Cut Generation.

The subset-row cuts [Jepsen et al., 2008], strengthened capacity cuts [Baldacci et al., 2008], and elementary cuts [Pecin et al., 2017], used to enhance the route-based formulation, are also valid for the trip-based formulation. This is because a trip is defined as a route-time pair, and the coefficients in these cuts depend solely on the associated route. Specifically, consider one of such cuts valid for the route-based formulation, indexed by j and expressed as $\sum_{r\in\mathcal{R}} \delta_{jr} x_r \leq f_j$. For the trip-based formulation, the corresponding cut $\sum_{s\in\mathcal{S}_0} \delta_{jr_s} y_s \leq f_j$ remains valid. A trip-based solution $\boldsymbol{y}$ violates these cuts if and only if the associated route-based solution $\boldsymbol{x^y}$ violates them, where $x_r^{\boldsymbol{y}} = \sum_{s\in\mathcal{S}_0: r_s = r} y_s$ for each $r \in \mathcal{R}$. These cuts are identified using the methods in Yang [2025].

We identify the time points relevant to constraints (2.8c) using a dynamic time discovery strategy. Given the potentially large number of constraints (2.8c), directly

solving LSF by incorporating all of them can be computationally expensive. To address this challenge, we replace $\mathcal{H}$ in constraints (2.8c) with a subset $\mathcal{T} \subseteq \mathcal{H}$, which is initially empty and then expanded by incrementally adding time points as needed. Specifically, for a given trip-based solution $\boldsymbol{y}$, we verify whether constraints (2.8c) are satisfied for all $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$. If they are satisfied, then by Proposition 2, the constraints (2.8c) are satisfied for all $t \in [a_0, b_0]$. Otherwise, a time point in $\{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$ corresponding to a violated constraint (2.8c) is added to $\mathcal{T}$. This dynamic time discovery strategy significantly reduces the number of time points that need to be identified.

2.4.3.2 Column Generation.

Let $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ denote the dual solution of LSF, where $\boldsymbol{\mu} = \{\mu_t : t \in \mathcal{T}\}$ represents the dual vector associated with constraints (2.8c), and $\boldsymbol{\lambda}$ is the dual vector for the remaining constraints, which depend solely on the routes and are independent of departure times.

For trip $s \in \mathcal{S}$, let $c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu})$ denote its reduced cost w.r.t. dual $(\boldsymbol{\lambda}, \boldsymbol{\mu})$. Note that the value of $c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) + \sum_{t \in \mathcal{T}} \mu_t \gamma_{ts}$ depends solely on the associated route r_s. Thus we can define $\hat{c}_{r_s}(\boldsymbol{\lambda}) = c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) + \sum_{t \in \mathcal{T}} \mu_t \gamma_{ts}$. Consequently, the pricing problem can be represented as follows.

$$\min_{s \in \{(r,\tau): r \in \mathcal{R}_2, \tau \in \mathcal{G}_r^2\}} c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \min_{r \in \mathcal{R}_2} \left(\hat{c}_r(\boldsymbol{\lambda}) + \min_{s \in \{(r,\tau): \tau \in \mathcal{G}_r^2\}} \sum_{t \in \mathcal{T}} (-\mu_t) \gamma_{ts} \right). \tag{2.9}$$

A straightforward approach to solving the pricing problem (2.9) involves evaluating the reduced cost of all trips in $\{(r, \tau) : r \in \mathcal{R}_2, \tau \in \mathcal{G}_r^2\}$. However, this method can be inefficient because, for route $r \in \mathcal{R}_2$, the set $\mathcal{G}_r^2$ may contain a large number of time points. Nevertheless, Proposition 7 shows that it is sufficient to consider only a subset of these time points when solving the pricing problem.

Specifically, the subset is defined as $\mathcal{E}(\mathcal{T}, \mathcal{G}_r^2) = (\mathcal{G}_r^2 \cap \{t - d_r : t \in \mathcal{T}\}) \cup \mathcal{F}(\mathcal{G}_r^2)$, where $\mathcal{F}(\mathcal{G}_r^2) = \{\tau \in \mathcal{G}_r^2 : \tau + 1 \notin \mathcal{G}_r^2\}$.

Proposition 7. *For any route $r \in \mathcal{R}_2$, the inner minimization problem in (2.9) can be equivalently reformulated as*

$$\min_{s \in \{(r,\tau):\tau \in \mathcal{G}_r^2\}} \sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts} = \min_{s \in \{(r,\tau):\tau \in \mathcal{E}(\mathcal{T},\mathcal{G}_r^2)\}} \sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts} = \min_{\tau \in \mathcal{E}(\mathcal{T},\mathcal{G}_r^2)} \sum_{t \in \mathcal{T} \cap [\tau, \tau + d_r)} (-\mu_t). \tag{2.10}$$

Proof. Consider a route $r \in \mathcal{R}_2$ and the set of trips $\{s \in \mathcal{S} : r_s = r\}$. For a given time point $t \in \mathcal{T}$, the function $\gamma_{ts} = \mathbb{I}_{[\tau_s, \tau_s + d_r)}(t)$ is a step function with respect to τ_s. This function is left-continuous and increases only at $\tau_s = t - d_r$. Since $-\mu_t \geq 0$ for all $t \in \mathcal{T}$, it follows that $\sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts}$ is also a step function. This function is left-continuous and can potentially increase only at $\tau_s \in \{t - d_r : t \in \mathcal{T}\}$.

Now, consider the trips $\{s \in \mathcal{S} : r_s = r, \tau_s \in \mathcal{G}_r^2\}$ for the route $r \in \mathcal{R}_2$. Since $\tau_s \in \mathcal{G}_r^2 \subset \mathbb{Z}_+$, $d_r \in \mathbb{Z}_{++}$, and $\mathcal{T} \subseteq \mathcal{H} \subset \mathbb{Z}_+$, it follows that the minimum value of $\sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts}$ can be achieved only at $\tau_s \in (\mathcal{G}_r^2 \cap \{t - d_r : t \in \mathcal{T}\}) \cup \mathcal{F}(\mathcal{G}_r^2)$, where $\mathcal{F}(\mathcal{G}_r^2) = \{\tau \in \mathcal{G}_r^2 : \tau + 1 \notin \mathcal{G}_r^2\}$. Consequently, we have $\min_{s \in \{(r,\tau):\tau \in \mathcal{G}_r^2\}} \sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts} = \min_{s \in \{(r,\tau):\tau \in \mathcal{E}(\mathcal{T},\mathcal{G}_r^2)\}} \sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts}$, where $\mathcal{E}(\mathcal{T}, \mathcal{G}_r^2) = (\mathcal{G}_r^2 \cap \{t - d_r : t \in \mathcal{T}\}) \cup \mathcal{F}(\mathcal{G}_r^2)$.

Moreover, since $\gamma_{ts} = \mathbb{I}_{[\tau_s, \tau_s + d_r)}(t)$, it follows that $\sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts} = \sum_{t \in \mathcal{T} \cap [\tau_s, \tau_s + d_r)} (-\mu_t)$. Therefore, we have $\min_{s \in \{(r,\tau):\tau \in \mathcal{E}(\mathcal{T},\mathcal{G}_r^2)\}} \sum_{t \in \mathcal{T}} (-\mu_t)\gamma_{ts} = \min_{\tau \in \mathcal{E}(\mathcal{T},\mathcal{G}_r^2)} \sum_{t \in \mathcal{T} \cap [\tau, \tau + d_r)} (-\mu_t)$. This completes the proof. □

Consequently, the pricing problem (2.9) can be solved using a two-level loop. In the outer loop, the value of $\hat{c}_r(\boldsymbol{\lambda})$ is computed for each $r \in \mathcal{R}_2$. In the inner loop, the expression $\sum_{t \in \mathcal{T} \cap [\tau, \tau + d_r)} (-\mu_t)$ is evalutated for each $\tau \in \mathcal{E}(\mathcal{T}, \mathcal{G}_r^2)$. This novel algorithm avoids evaluating the reduced costs of trips $\{(r, \tau) : r \in \mathcal{R}_2, \tau \in \mathcal{G}_r^2 \setminus \mathcal{E}(\mathcal{T}, \mathcal{G}_r^2)\}$, thereby significantly accelerating the solution of the pricing problem.

2.4.3.3 Dual Picking.

In addition to solving LSF with a column-and-cut generation approach, we develop a dual-picking procedure to generate multiple feasible dual solutions of LSF, thereby enabling further elimination of trips. Given a subset of trips $\bar{\mathcal{S}} \subseteq \mathcal{S}$, the dual-picking formulation aims to maximize the sum of the dual objective and the average reduced cost of trips in $\bar{\mathcal{S}}$. The formulation is presented as follows, where $L(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is the linear expression representing the objective value of the dual solution $(\boldsymbol{\lambda}, \boldsymbol{\mu})$.

$$(DPF(\bar{\mathcal{S}})) \qquad \max L(\boldsymbol{\lambda}, \boldsymbol{\mu}) + \frac{1}{|\bar{\mathcal{S}}|} \sum_{s \in \bar{\mathcal{S}}} c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}), \tag{2.11a}$$

$$\text{s.t.} \qquad c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) \geq 0, \quad \forall s \in \{(r, \tau) : r \in \mathcal{R}_2, \tau \in \mathcal{G}_r^2\}, \tag{2.11b}$$

$$\text{sign constraints on } \boldsymbol{\lambda}, \tag{2.11c}$$

$$\mu_t \leq 0, \quad \forall t \in \mathcal{T}. \tag{2.11d}$$

The objective function (2.11a) maximizes the sum of the dual objective and the average reduced cost of the trips in $\bar{\mathcal{S}}$. Constraints (2.11b) enforce nonnegativity of the reduced cost for every trip in $\{(r, \tau) : r \in \mathcal{R}_2, \tau \in \mathcal{G}_r^2\}$. Finally, the sign constraints (2.11c) and (2.11d) specify the required nonpositivity or nonnegativity of the decision variables.

Since the number of constraints in (2.11b) can be very large, we solve $DPF(\bar{\mathcal{S}})$ using a cut generation approach. The separation problem of identifying the most violated constraint (2.11b) for a given $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is equivalent to the pricing problem (2.9) for finding the trip with the minimum reduced cost. Consequently, the pricing method for problem (2.9) can be directly applied to the cut generation approach for solving $DPF(\bar{\mathcal{S}})$.

We generate multiple feasible duals by solving $DPF(\bar{\mathcal{S}})$ for C times, each with

a different choice of $\bar{\mathcal{S}}$. Specifically, we first partition the routes in $\mathcal{R}_2$ into C clusters using the k-means algorithm, where the distance between two routes $r_1, r_2 \in \mathcal{R}_2$ is defined as the cardinality of the symmetric difference between $\mathcal{N}(r_1)$ and $\mathcal{N}(r_2)$, i.e., $|(\mathcal{N}(r_1) \setminus \mathcal{N}(r_2)) \cup (\mathcal{N}(r_2) \setminus \mathcal{N}(r_1))|$. For each cluster of routes $\bar{\mathcal{R}} \subseteq \mathcal{R}_2$, the corresponding subset of trips is defined as $\bar{\mathcal{S}} = \{(r, \tau) : r \in \bar{\mathcal{R}}, \tau \in \mathcal{F}(\mathcal{G}_r^2)\}$.

2.4.3.4 Variable Fixing Using Trip-Based Dual Solutions.

This section presents methods for efficiently eliminating trips by leveraging a feasible dual solution $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ to LSF. In particular, if the precondition of Proposition 8 is satisfied for a route $r \in \mathcal{R}_2$, we eliminate r from $\mathcal{R}_2$, which is equivalently to eliminating all trips $s \in \mathcal{S}_2$ where $r_s = r$.

Proposition 8. *Consider a feasible dual solution $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ and a route $r \in \mathcal{R}_2$. If $\hat{c}_r(\boldsymbol{\lambda}) - \sum_{t \in \mathcal{T}} \mu_t \beta_{tr} > UB - L(\boldsymbol{\lambda}, \boldsymbol{\mu})$, then no trip in $\{(r, \tau) : \tau \in \mathcal{G}_r^2\}$ can be selected in any optimal solution to the CMTVRPTW.*

Proof. Consider a route $r \in \mathcal{R}_2$ and a feasible dual solution $(\boldsymbol{\lambda}, \boldsymbol{\mu})$. For any trip $s \in \{(r, \tau) : \tau \in \mathcal{G}_r^2\}$ and time point $t \in \mathcal{T}$, we have $\gamma_{ts} = \mathbb{I}_{[\tau_s, \tau_s + d_r)}(t) \geq \mathbb{I}_{[l_r, e_r + d_r)}(t) = \beta_{tr}$, since $\mathcal{G}_r^2 \subseteq [e_r, l_r]$. Additionally, because $\mu_t \leq 0$ for all $t \in \mathcal{T}$, it follows that $c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \hat{c}_r(\boldsymbol{\lambda}) - \sum_{t \in \mathcal{T}} \mu_t \gamma_{ts} \geq \hat{c}_r(\boldsymbol{\lambda}) - \sum_{t \in \mathcal{T}} \mu_t \beta_{tr}$. Consequently, if $\hat{c}_r(\boldsymbol{\lambda}) - \sum_{t \in \mathcal{T}} \mu_t \beta_{tr} > UB - L(\boldsymbol{\lambda}, \boldsymbol{\mu})$, then the reduced cost satisfies $c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) > UB - L(\boldsymbol{\lambda}, \boldsymbol{\mu})$ for all $s \in \{(r, \tau) : \tau \in \mathcal{G}_r^2\}$. Based on the variable fixing technique, these trips cannot be selected in any optimal solution to the CMTVRPTW, which completes the proof. □

For route $r \in \mathcal{R}_2$, define $\mathcal{D}(\mathcal{T}, \mathcal{G}_r^2) = (\mathcal{T} \cap \mathcal{G}_r^2) \cup \mathcal{E}(\mathcal{T}, \mathcal{G}_r^2)$ and denote $m_r = |\mathcal{D}(\mathcal{T}, \mathcal{G}_r^2)|$. Arrange the elements of $\mathcal{D}(\mathcal{T}, \mathcal{G}_r^2)$ in ascending order as $\theta_1^r < \theta_2^r < \cdots < \theta_{m_r}^r$, and define $\theta_0^r = \min \mathcal{G}_r^2 - 1$, i.e., one unit earlier than the earliest departure time in $\mathcal{G}_r^2$. Then the following Lemma 1 holds, which will be utilized in the proofs

of Proposition 9 and Proposition 10.

Lemma 1. *Consider a time point $t \in \mathcal{T}$, a set of departure times $\mathcal{G}_r \subseteq [e_r, l_r] \cap \mathbb{Z}_+$, and a trip $s_2 = (r, \theta_j^r)$, where $r \in \mathcal{R}_2$ and $\theta_j^r \in \mathcal{D}(\mathcal{T}, \mathcal{G}_r)$ represents the j-th smallest element in $\mathcal{D}(\mathcal{T}, \mathcal{G}_r)$. For all $s_1 \in \{(r, \tau) : \tau \in \mathcal{G}_r \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r - 1\}\}$, it follows that $\gamma_{ts_1} = \gamma_{ts_2}$.*

Proof. Consider a route $r \in \mathcal{R}_2$ and the set of trips $\{s \in \mathcal{S} : r_s = r\}$. For a given time point $t \in \mathcal{T}$, the function $\gamma_{ts} = \mathbb{I}_{[\tau_s, \tau_s + d_r)}(t)$ is a step function with respect to τ_s. This function is left-continuous, increases only at $\tau_s = t - d_r$, and decreases only at $\tau_s = t$. Let $\underline{\tau} = \min \mathcal{G}_r \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r\}$, i.e., the smallest element in the set $\mathcal{G}_r \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r\}$. By the definition of $\mathcal{D}(\mathcal{T}, \mathcal{G}_r)$, it holds that $t - d_r \notin [\underline{\tau}, \theta_j^r)$ and $t \notin [\underline{\tau}, \theta_j^r)$. Consequently, γ_{ts} remains constant for all $\tau_s \in [\underline{\tau}, \theta_j^r]$. This implies that $\gamma_{ts_1} = \gamma_{ts_2}$ for all $s_1 \in \{(r, \tau) : \tau \in \mathcal{G}_r \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r - 1\}\}$, which completes the proof. □

Non-optimal trips can then be eliminated using Proposition 9. Specifically, if the precondition of Proposition 9 is satisfied for trip (r, θ_j^r), we update $\mathcal{G}_r^2 \leftarrow \mathcal{G}_r^2 \setminus \{\theta_{j-1}^r + 1, \ldots, \theta_j^r\}$, which is equivalently to eliminating all trips $\{(r, \tau) : \tau \in \mathcal{G}_r^2 \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r\}\}$.

Proposition 9. *Consider a feasible dual solution $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ and a trip $s = (r, \theta_j^r)$ where $r \in \mathcal{R}_2$ and $\theta_j^r \in \mathcal{D}(\mathcal{T}, \mathcal{G}_r^2)$. If $c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) > UB - L(\boldsymbol{\lambda}, \boldsymbol{\mu})$, then no trip in $\{(r, \tau) : \tau \in \mathcal{G}_r^2 \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r\}\}$ can be selected in any optimal solution to the CMTVRPTW.*

Proof. According to Lemma 1, we have $\gamma_{ts_1} = \gamma_{ts}$ for all $s_1 \in \{(r, \tau) : \tau \in \mathcal{G}_r^2 \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r - 1\}\}$. It follows that $c'_{s_1}(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \hat{c}_r(\boldsymbol{\lambda}) - \sum_{t \in \mathcal{T}} \mu_t \gamma_{ts_1} = \hat{c}_r(\boldsymbol{\lambda}) - \sum_{t \in \mathcal{T}} \mu_t \gamma_{ts} = c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu})$. Hence, if $c'_s(\boldsymbol{\lambda}, \boldsymbol{\mu}) > UB - L(\boldsymbol{\lambda}, \boldsymbol{\mu})$, then $c'_{s_1}(\boldsymbol{\lambda}, \boldsymbol{\mu}) > UB - L(\boldsymbol{\lambda}, \boldsymbol{\mu})$ holds as well. Therefore, by the variable fixing technique, no trip in

$\{(r,\tau) : \tau \in \mathcal{G}_r^2 \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r\}\}$ can be selected in any optimal solution to the CMTVRPTW. This completes the proof. □

Let $\mathcal{S}_3$ denote the set of trips remaining after elimination through variable fixing. Define $\mathcal{R}_3 = \{r \in \mathcal{R}_2 : \exists s \in \mathcal{S}_3 \text{ such that } r_s = r\}$. For each $r \in \mathcal{R}_3$, let $\mathcal{G}_r^3 = \{\tau \in \mathcal{G}_r^2 : (r,\tau) \in \mathcal{S}_3\}$.

2.4.4 Optimal Solution Computation

We obtain the optimal solution by solving $SF(\mathcal{S}_3, \mathcal{H})$, derived from formulation $SF(\mathcal{S}_0, \mathcal{H})$ by replacing $\mathcal{S}_0$ with $\mathcal{S}_3 = \{(r,\tau) : r \in \mathcal{R}_3, \tau \in \mathcal{G}_r^3\}$. However, due to the large number of time points in $\mathcal{H}$, it is computationally expensive to solve $SF(\mathcal{S}_3, \mathcal{H})$ directly. To address this, we employ a dynamic time discovery strategy. Specifically, in each iteration, we solve the relaxation $SF(\mathcal{S}_3, \mathcal{T})$, derived from $SF(\mathcal{S}_3, \mathcal{H})$ by replacing $\mathcal{H}$ with a subset $\mathcal{T}$. If the optimal solution $\boldsymbol{y}$ to $SF(\mathcal{S}_3, \mathcal{T})$ satisfies all constraints (2.8c) corresponding to the time points $\{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$, then it is feasible to $SF(\mathcal{S}_3, \mathcal{H})$ by Proposition 2. In this case, $\boldsymbol{y}$ is also optimal to the CMTVRPTW, since $SF(\mathcal{S}_3, \mathcal{T})$ is a relaxation of $SF(\mathcal{S}_3, \mathcal{H})$. Otherwise, a time point in $\{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$ corresponding to a violated constraint (2.8c) is added to $\mathcal{T}$. The process is then repeated, and $SF(\mathcal{S}_3, \mathcal{T})$ is solved again with the updated $\mathcal{T}$. Furthermore, as detailed in Appendix A.2, this algorithm can be enhanced by additionally incorporating side constraints adapted from the route-based formulation.

To reduce the number of iterations, $\mathcal{T}$ is first initialized with the set of time points corresponding to constraints (2.8c) separated during the solution of LSF, and then expanded using a warm-start procedure. Specifically, a trip-based IP formulation is solved, derived from $SF(\mathcal{S}_3, \mathcal{T})$ by replacing $\mathcal{S}_3$ with a small number of trips having relatively smaller reduced costs. During this process, time points

corresponding to constraints (2.8c) violated by identified integer solutions are added to $\mathcal{T}$.

Based on the optimality property of vehicle departure times established in Proposition 10, it is sufficient to incorporate only a subset of trips $\{(r,\tau) : r \in \mathcal{R}_3, \tau \in \mathcal{D}(\mathcal{T},\mathcal{G}_r^3)\}$, when solving $SF(\mathcal{S}_3,\mathcal{T})$ using an IP solver. By eliminating the trips $\{(r,\tau) : r \in \mathcal{R}_3, \tau \in \mathcal{G}_r^3 \setminus \mathcal{D}(\mathcal{T},\mathcal{G}_r^3)\}$, the efficiency of solving the trip-based IP formulation is significantly improved.

Proposition 10. *Consider a trip $s_1 = (r,\tau_1) \in \mathcal{S}_3$ where $r \in \mathcal{R}_3$ and $\tau_1 \in \mathcal{G}_r^3 \setminus \mathcal{D}(\mathcal{T},\mathcal{G}_r^3)$. If s_1 is selected in an optimal solution to $SF(\mathcal{S}_3,\mathcal{T})$, then there exists another optimal solution obtained by replacing s_1 with trip $s_2 = (r,\tau_2)$ where $\tau_2 = \min\{\tau \in \mathcal{D}(\mathcal{T},\mathcal{G}_r^3) : \tau > \tau_1\}$.*

Proof. Consider a trip $s_1 = (r,\tau_1) \in \mathcal{S}_3$, where $r \in \mathcal{R}_3$ and $\tau_1 \in \mathcal{G}_r^3 \setminus \mathcal{D}(\mathcal{T},\mathcal{G}_r^3)$. Assume that s_1 is selected in an optimal solution $\boldsymbol{y}^1$ to $SF(\mathcal{S}_3,\mathcal{T})$. Construct a new solution $\boldsymbol{y}^2$ by replacing s_1 in $\boldsymbol{y}^1$ with trip $s_2 = (r,\tau_2)$, where $\tau_2 = \min\{\tau \in \mathcal{D}(\mathcal{T},\mathcal{G}_r^3) : \tau > \tau_1\}$. Specifically, $\boldsymbol{y}^2$ is obtained by setting $y^2_{s_1} = 0$, $y^2_{s_2} = 1$, and $y^2_s = y^1_s$ for all $s \in \mathcal{S}_3 \setminus \{s_1, s_2\}$.

Since $r_{s_1} = r_{s_2} = r$, $\boldsymbol{y}^2$ has the same objective value as $\boldsymbol{y}^1$ and still satisfies constraints (2.8b).

Assume that τ_2 is the j-th smallest element in $\mathcal{D}(\mathcal{T},\mathcal{G}_r^3)$, i.e., $\tau_2 = \theta_j^r$. By the definition of τ_1 and τ_2, it follows that $\tau_1 \in \mathcal{G}_r^3 \cap \{\theta_{j-1}^r + 1, \ldots, \theta_j^r - 1\}$. Consequently, Lemma 1 implies that $\gamma_{ts_1} = \gamma_{ts_2}$ for all $t \in \mathcal{T}$. Hence, $\boldsymbol{y}^2$ satisfies constraints (2.8c) for all $t \in \mathcal{T}$, and is feasible to $SF(\mathcal{S}_3,\mathcal{T})$.

Therefore, $\boldsymbol{y}^2$ is feasible to $SF(\mathcal{S}_3,\mathcal{T})$ and has the same objective value as $\boldsymbol{y}^1$. Since $\boldsymbol{y}^1$ is optimal to $SF(\mathcal{S}_3,\mathcal{T})$, we conclude that $\boldsymbol{y}^2$ is also optimal to $SF(\mathcal{S}_3,\mathcal{T})$. This completes the proof. □

2.5 Computational Results

This section presents the computational results of our algorithm for solving the CMTVRPTW, comparing its performance with the state-of-the-art method proposed by Yang [2025]. Both our algorithm and the program of Yang [2025] are executed on the same desktop, equipped with an Intel® Core™ i9-13900K CPU @ 4.30GHz and 128 GB of RAM. Both methods are implemented in C++ and utilize Gurobi 9.1.1 as the LP and IP solver. To solve the IP formulations, both programs utilize 32 threads for parallel computation. The input data (UB, $\mathcal{R}_1$, and a set of valid cuts) for our algorithm are generated using the program of Yang [2025]. Following the convention in the literature, the CMTVRPTW test instances are adapted from VRPTW instances by modifying the fleet sizes and vehicle capacities.

2.5.1 Results on Benchmark Instances

We consider 90 large-scale CMTVRPTW benchmark instances introduced by Yang [2025], derived from 30 VRPTW instances (C2, R2, and RC2) with 200 customers from the G02 group [Gehring and Homberger, 2001]. For each VRPTW instance, three CMTVRPTW instances are generated by including the first 140, the first 170, and all 200 customers. The number of vehicles is set to 12, 16, and 20 for instances with 140, 170, and 200 customers, respectively. The vehicle capacity is fixed at 100 for these instances.

Among the 90 large-scale benchmark instances, computational results for 10 of them are not reported for one of the following reasons: (i) Yang [2025] fails to enumerate a set of candidate routes and is unable to solve the instance; (ii) Yang [2025] solves the instance directly at the root node; or (iii) Yang [2025] proves optimality for the instance through trial route enumeration. In case (i), our algorithm cannot be executed due to the lack of input. In cases (ii) and (iii), these

instances are solved to optimality prior to generating input data for our algorithm. The remaining 80 instances are divided into two categories: 14 challenging instances that were not solved to optimality within 3 hours by the benchmark method in Yang [2025], and 66 easier instances that were solved to optimality.

Table 2.1 presents the computational results for 14 challenging instances, comparing the benchmark method proposed by Yang [2025] with our algorithm. In the table, "$|\mathcal{N}|$" is the number of customers, and "Obj" represents the objective value of the incumbent solution. "G_{10h}" and "G_{3h}" denote the optimality gaps, expressed as percentages, after 10 hours and 3 hours of computation, respectively. "CT_{IP}" is the time required to solve the IP formulation, and "CT_{tot}" is the total computing time for an instance. For our algorithm, "CT_{tot}" also includes the time consumed by the program of Yang [2025] for generating the input data of our algorithm.

Table 2.1 compares the two methods in terms of objective values, optimality gaps, and computing times. Compared to the benchmark method, our algorithm achieves a smaller objective value for one instance and equal objective values for the remaining instances. In terms of optimality gaps, our algorithm performs better by proving optimality for 10 instances within a ten-hour time limit, compared to 4 instances for the benchmark method. Among the 14 challenging instances, our algorithm achieves smaller optimality gaps for 8 instances and equal optimality gaps for 4 instances. Additionally, within a 3-hour time limit, our algorithm solves one more instance to optimality than the benchmark method. With respect to computing times, our algorithm demonstrates an advantage by requiring shorter total computing times ("CT_{tot}") for 9 instances, while taking longer for only one instance. On average, our algorithm consumes on average 79.4% of the total computing time required by the benchmark for these challenging instances. This

improvement primarily results from that our algorithm solves the IP formulation more efficiently, consuming on average 72.2% of the time ("CT_{IP}") required by the benchmark. Moreover, for instances in which both algorithms attain the same objective value, our algorithm finds the incumbent solution in 1.3 hours on average, slightly faster than the benchmark method, which requires 1.5 hours on average. These results demonstrate that our algorithm solves challenging instances more efficiently than the benchmark method.

Table 2.1: Computational results on challenging instances

Instance	$\|\mathcal{N}\|$	Yang [2025]					Our Algorithm				
		Obj	G_{10h}	G_{3h}	CT_{IP}	CT_{tot}	*Obj*	G_{10h}	G_{3h}	CT_{IP}	CT_{tot}
RC2_2_02	140	3573.6	0.00	0.00	2.0	2.8	3573.6	0.00	0.00	0.7	**1.6**
R2_2_01	170	4631.2	0.10	0.37	9.3	10.0	4631.2	**0.00**	0.28	3.7	**4.3**
R2_2_05	170	4476.5	0.19	0.44	9.3	10.0	4476.5	**0.00**	0.34	5.0	**5.7**
R2_2_06	170	4296.6	0.21	0.43	9.2	10.0	4296.6	**0.00**	0.48	5.7	**6.6**
R2_2_09	170	4378.1	0.10	0.40	9.4	10.0	4378.1	**0.00**	0.35	7.7	**8.3**
RC2_2_01	170	4404.3	0.60	0.60	8.1	10.0	4404.3	**0.43**	0.81	7.9	10.0
RC2_2_03	170	4158.1	0.00	0.26	2.9	5.3	4158.1	0.00	0.35	2.1	**4.6**
RC2_2_08	170	4231.2	**0.27**	0.37	7.3	10.0	4231.2	0.34	0.49	7.2	10.0
R2_2_01	200	5295.8	0.42	0.47	8.5	10.0	5295.8	**0.32**	0.67	8.5	10.0
R2_2_05	200	5119.5	0.08	0.32	8.9	10.0	5119.5	**0.00**	0.43	8.0	**9.1**
R2_2_06	200	4950.1	0.05	0.24	8.9	10.0	4950.1	**0.00**	0.00	1.8	**3.0**
R2_2_09	200	5034.4	**0.48**	0.48	7.9	10.0	**5033.6**	0.55	0.64	7.7	10.0
RC2_2_06	200	4841.5	0.00	0.31	3.3	5.9	4841.5	0.00	0.33	2.2	**4.8**
RC2_2_07	200	4784.0	0.00	0.17	0.3	**3.2**	4784.0	0.00	0.18	0.6	3.7

Notes. The unit for computing time is hours. The time limit is set as 10 hours. The optimality gaps G_{10h} and G_{3h} are reported as percentages.

Table 2.2 presents key statistics on routes, departure times, and time points generated by our algorithm for the challenging instances. In this table, the notation "$|\mathcal{R}_1|$" represents the number of input routes, which are obtained using the program of Yang [2025]. "$|\mathcal{R}_3|$" denotes the number of routes incorporated for solving the trip-based IP formulation. "Δ_R" is the percentage reduction in the number of routes, i.e., $\Delta_R = \frac{|\mathcal{R}_1|-|\mathcal{R}_3|}{|\mathcal{R}_1|} \times 100$. "$|\mathcal{R}_3^m|$" represents the number of routes in $\mathcal{R}_3$ that have more than one departure time, i.e., $\mathcal{R}_3^m = \{r \in \mathcal{R}_3 : e_r < l_r\}$. The notation "Avg $|\mathcal{G}_r^1|$" denotes the average number of departure times initialized for each route in $\mathcal{R}_3^m$, while "Avg $|\mathcal{D}(\mathcal{T}, \mathcal{G}_r^3)|$" represents the average number of departure times

considered for each route in $\mathcal{R}_3^m$ during the solution of the trip-based IP formulation. Additionally, the terms "$|\mathcal{T}_{LP}|$", "$|\mathcal{T}_w|$", and "$|\mathcal{T}_{IP}|$" represent the number of time points identified during the solution phases of the LP formulation, the small-sized IP formulation used for warm-start, and the final IP formulation for computing the optimal integer solution, respectively. Specifically, when solving the final IP formulation, the set of time points is given by $\mathcal{T} = \mathcal{T}_{LP} \cup \mathcal{T}_w \cup \mathcal{T}_{IP}$.

Table 2.2: Route elimination, departure time elimination, and time discovery on challenging instances

Instance	$\|\mathcal{N}\|$	Route Elimination				Departure Time Elimination		Time Discovery		
		$\|\mathcal{R}_1\|$	$\|\mathcal{R}_3\|$	$\Delta_R(\%)$	$\|\mathcal{R}_3^m\|$	Avg $\|\mathcal{G}_r^1\|$	Avg $\|\mathcal{D}(\mathcal{T}, \mathcal{G}_r^3)\|$	$\|\mathcal{T}_{LP}\|$	$\|\mathcal{T}_w\|$	$\|\mathcal{T}_{IP}\|$
RC2_2_02	140	111559	108836	2.4	12239	552.0	2.3	25	2	0
R2_2_01	170	176521	174607	1.1	1632	400.0	1.1	26	3	0
R2_2_05	170	219185	217076	1.0	6230	700.0	1.4	33	1	0
R2_2_06	170	251669	246944	1.9	38677	961.7	1.6	26	2	0
R2_2_09	170	172614	170095	1.5	8448	805.6	1.7	36	1	0
RC2_2_01	170	481162	459439	4.5	23010	235.3	1.7	32	6	0
RC2_2_03	170	316467	252668	20.2	98619	1518.3	2.9	20	2	0
RC2_2_08	170	371024	351357	5.3	151941	1406.5	1.4	8	2	0
R2_2_01	200	399382	397926	0.4	2164	356.4	1.2	32	2	0
R2_2_05	200	207115	204206	1.4	5513	645.4	1.5	26	7	0
R2_2_06	200	124971	122807	1.7	14765	1197.3	1.3	9	2	0
R2_2_09	200	746309	717067	3.9	32458	731.7	1.9	25	14	0
RC2_2_06	200	195561	192912	1.4	50971	492.5	2.9	34	1	0
RC2_2_07	200	218397	73702	66.3	22319	786.0	3.1	26	1	0

Table 2.2 presents the effectiveness of our algorithm by demonstrating that the number of trips and side constraints (2.8c) incorporated into the IP formulation can be significantly reduced. Notably, eliminating routes and departure times is equivalent to eliminating trips, as a trip is defined by a route and its departure time from the depot. The table shows that the number of routes can be consistently reduced across all instances, with the percentage reduction ranging from 0.4% to 66.3%. This validates the effectiveness of the variable fixing technique described in Proposition 8. The average number of departure times ("Avg $|\mathcal{G}_r^1|$") for routes in $\mathcal{R}_3^m$ can surpass 1,500. However, by applying the dominance rule (Proposition 6), the variable fixing technique (Proposition 9), and the optimality property of vehicle departure times (Proposition 10), the average number of departure times for these

routes is reduced significantly to a range of 1.1 to 3.1 ("Avg $|\mathcal{D}(\mathcal{T}, \mathcal{G}_r^3)|$"). The reduction in departure times eliminates 93.9% of the trips. Furthermore, although the number of time points in $\mathcal{H}$ exceeds 25,000 for all these instances, only 29 time points ($\mathcal{T}$) on average need to be identified, which significantly reduces the number of trips and side constraints (2.8c) incorporated into the IP formulation. Notably, all these time points are identified during the solution phases of the trip-based LP formulation ("$|\mathcal{T}_{LP}|$") and the small-sized IP formulation ("$|\mathcal{T}_w|$"), demonstrating the effectiveness of dynamic time discovery and the warm-start strategy for initializing $\mathcal{T}$ prior to solving the final IP formulation.

Appendix A.3 presents the computational results for 66 easy instances, showing that both methods can solve these instances to optimality within a few hours, and our algorithm delivers performance comparable to the benchmark method on these instances.

2.5.2 Results on Extended Instances

We further compare the benchmark method with our algorithm on extended instances, which are derived from the 140-customer benchmark instances to mitigate frequent occurrences of time or memory limit violations. These extended instances are derived by reducing either the fleet sizes (K) or the vehicle capacities (Q). As smaller fleet sizes and reduced capacities impose stricter constraints on the delivery system, allowing each vehicle to perform multiple trips becomes increasingly essential to meet customer demands. For each benchmark instance, four extended instances are generated by either reducing K from 12 to 10 or 11, or by reducing Q from 100 to 60 or 80. Out of the 120 extended instances, the benchmark method fails to provide upper bounds for 3 instances due to out-of-memory errors encountered while solving the TOPTW for separating the side constraints (2.1c). For the remaining 117 instances, both the benchmark method and our algorithm

successfully generate feasible solutions.

Table 2.3 presents computational results for 28 instances, where either the benchmark method or our algorithm requires more than one hour of computing time. For these instances, our algorithm outperforms the benchmark in terms of objective values, optimality gaps, and computing times. Specifically, our algorithm achieves smaller objective values for 6 instances and equal objective values for the remaining instances. Additionally, our algorithm achieves smaller optimality gaps for 15 instances, while the benchmark has smaller optimality gaps for only 3 instances. Furthermore, our algorithm proves optimality for 23 instances, compared to only 11 instances for the benchmark. Notably, there is just one instance where the benchmark proves optimality while our algorithm does not. Furthermore, our algorithm demonstrates superior computational efficiency, with shorter total computing times ("CT_{tot}") for 20 instances and equal computing times for 4 instances. Across all 28 instances, our algorithm achieves an average total computing time that is only 63.6% of that required by the benchmark method. Moreover, for instances in which both algorithms attain the same objective value, our algorithm finds the incumbent solution in 0.4 hours on average, slightly faster than the benchmark method, which requires 0.5 hours on average.

Figure 2.2 presents the average computing times ("CT_{tot}") for instances with varying fleet sizes and vehicle capacities. The benchmark group consists of instances where $K = 12$ and $Q = 100$, along with four extended groups: (i) $K = 11, Q = 100$; (ii) $K = 10, Q = 100$; (iii) $K = 12, Q = 80$; and (iv) $K = 12, Q = 60$. Each group contains 30 instances, with computing times averaged across the group. Figure 2.2(a) shows that the average CT_{tot} increases as the fleet size decreases. Specifically, when K decreases from 12 to 10, the average CT_{tot} of our algorithm and the benchmark method increases by factors of 1.2 and 0.8, respectively. Similarly,

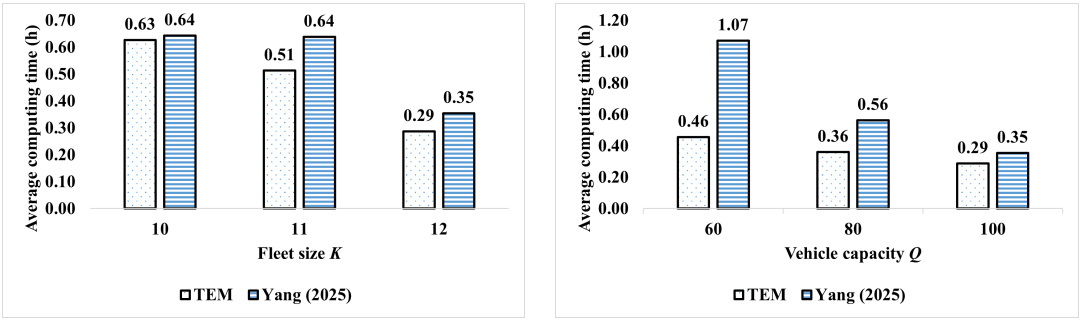
Table 2.3: Computational results on instances with varying fleet sizes and vehicle capacities

Instance	K	Q	Yang [2025]			Our Algorithm		
			Obj	$G_{3h}(\%)$	CT_{tot}	Obj	$G_{3h}(\%)$	CT_{tot}
C2_2_06	12	60	5031.9	0.16	3.0	**5027.9**	**0.00**	**0.3**
C2_2_07	12	60	5025.8	0.08	3.0	5025.8	**0.00**	**0.3**
C2_2_08	12	60	5004.0	0.21	3.0	5004.0	**0.00**	**0.4**
C2_2_08	12	80	3948.7	0.00	1.3	3948.7	0.00	**0.1**
C2_2_09	12	60	4986.8	0.09	3.0	4986.8	**0.00**	**0.6**
C2_2_10	12	60	4985.3	0.09	3.0	4985.3	**0.00**	**1.4**
R2_2_03	10	100	3626.1	0.24	3.0	3626.1	**0.00**	**2.1**
R2_2_03	11	100	3613.5	0.15	3.0	3613.5	**0.00**	**1.3**
R2_2_03	12	60	5331.7	0.02	3.0	5331.7	**0.00**	**0.7**
R2_2_05	12	80	4503.7	0.00	1.2	4503.7	0.00	**1.1**
R2_2_06	12	60	5412.6	0.03	3.0	5412.6	**0.00**	**0.8**
R2_2_06	12	80	4316.4	0.00	2.2	4316.4	0.00	**1.4**
R2_2_07	10	100	3585.9	**0.52**	3.0	3585.9	0.70	3.0
R2_2_07	11	100	3573.8	0.45	3.0	**3573.2**	**0.29**	3.0
R2_2_09	11	100	3805.7	0.00	**0.7**	3805.7	0.00	1.3
R2_2_09	12	80	4409.2	0.00	1.0	4409.2	0.00	**0.6**
RC2_2_02	10	100	3604.9	**0.61**	3.0	3604.9	0.79	3.0
RC2_2_02	11	100	3584.4	0.19	3.0	**3583.7**	**0.00**	**1.7**
RC2_2_05	12	80	4259.1	0.00	1.4	4259.1	0.00	**0.6**
RC2_2_06	10	100	3657.9	0.00	1.7	3657.9	0.00	**1.0**
RC2_2_06	11	100	3639.9	**0.00**	**2.8**	3639.9	0.01	3.0
RC2_2_06	12	80	4266.7	0.33	3.0	**4264.5**	**0.00**	**0.5**
RC2_2_07	10	100	3599.9	0.33	3.0	**3598.8**	**0.01**	3.0
RC2_2_07	11	100	3584.3	0.26	3.0	**3580.3**	**0.00**	**0.5**
RC2_2_07	12	60	5309.3	0.04	3.0	5309.3	**0.00**	**0.3**
RC2_2_07	12	80	4221.8	0.00	1.1	4221.8	0.00	**0.6**
RC2_2_08	10	100	3551.7	0.00	**1.0**	3551.7	0.00	1.3
RC2_2_08	11	100	3543.5	0.00	**0.6**	3543.5	0.00	1.3

Notes. The unit for computing time is hours. The time limit is set as 3 hours.

Figure 2.2(b) shows that the average CT_{tot} rises as the vehicle capacity decreases. Specifically, as Q decreases from 100 to 60, the average CT_{tot} of our algorithm and the benchmark method increases by factors of 0.6 and 2.1, respectively. These results indicate that the instances become increasingly challenging to solve as fleet size or vehicle capacity decreases.

Figure 2.2(b) indicates that as Q decreases from 100 to 60, the average CT_{tot} of the benchmark method increases more sharply compared to that of our algorithm. An intuitive explanation is that as vehicle capacity decreases, the number of customers visited per route tends to decrease, while the number of required routes increases. This results in larger TOPTW instances for separating constraints (2.1c), making the benchmark method more time-consuming.



(a) Average CT_{tot} for $Q = 100$ and $K = 10, 11, 12$.

(b) Average CT_{tot} for $K = 12$ and $Q = 60, 80, 100$.

Figure 2.2: Average computing times for instances with varying fleet sizes and vehicle capacities.

2.6 Summary

In this study, we propose an enhanced exact algorithm for solving the CMTVRPTW. Our algorithm constructs the initial set of trips from routes enumerated by Yang [2025] and obtains the optimal solution by solving the trip-based IP formulation. This approach improves upon the state-of-the-art exact algorithm [Yang, 2025], as the trip-based formulation enables a more efficient algorithm for the constraint separation problem compared to the superstructure-based formulation. To further enhance computational efficiency, we reduce the number of variables by eliminating trips using the dominance rule, variable fixing technique, and optimality property of vehicle departure times. Additionally, a dynamic time discovery strategy is introduced to reduce the number of identified time points, thereby decreasing the

number of constraints in the formulation.

Computational experiments on benchmark instances with up to 200 customers, as well as on extended instances, highlight the effectiveness of our algorithm. On 14 benchmark instances previously unsolved or considered difficult, our algorithm outperforms the state-of-the-art method proposed by Yang [2025] in terms of objective values, optimality gaps, and computational efficiency. Specifically, our algorithm consistently achieves smaller or equal incumbent objective values, proves optimality for 6 more previously unsolved instances, and reduces average computing time by over 20%. Moreover, our algorithm demonstrates superior performance on extended instances.

Chapter 3

Lagrangian Underestimate Cuts with Applications to Vehicle Routing

3.1 Introduction

This chapter addresses the second challenge outlined in Chapter 1: effective reduction of the solution search space for integer programs, with a particular focus on vehicle routing. The combinatorial nature of routing and network design problems causes the solution space to expand significantly as network size increases. To reduce the search space, we propose a novel class of optimality cuts, referred to as LUCs, which can be utilized to strengthen existing cuts and develop new domain reduction methods. These optimization techniques are broadly applicable to general integer programs. Particularly, we demonstrate their effectiveness by utilizing them to develop new arc fixing techniques for vehicle routing, which significantly reduce the underlying network size while preserving optimality.

IP formulations for routing and network design problems, such as the set-partitioning formulation of the vehicle routing problem, are difficult to solve

due to the combinatorial explosion of the solution space. Although advanced branch-and-price and column enumeration methods have led to significant progress, the computational complexity of large-scale instances remains a critical challenge. Consequently, developing effective strategies to reduce the search space is important for improving the efficiency and scalability of solution approaches. By leveraging Lagrangian duals, this work introduces novel optimality cuts and domain reduction techniques for IPs.

Deriving and incorporating valid cuts is a fundamental technique widely used in the literature to reduce the search space for solving integer programs, systematically eliminating fractional or suboptimal solutions from consideration. For example, subset-row cuts [Jepsen et al., 2008] have been demonstrated to effectively tighten the LP relaxations of vehicle routing problems. More recently, Yang [2023] introduces RCCs to eliminate non-optimal solutions by exploiting feasible LP duals, thereby emphasizing the effectiveness of dual-based approaches in deriving optimality cuts. In this study, we further demonstrate that optimality cuts can also be derived from Lagrangian duals (where sign constraints on dual variables are satisfied), and that these cuts can also be utilized to lift RCCs.

Domain reduction techniques provide a complementary approach to reducing the search space for IPs. For example, RCF has been extensively utilized in eliminating decision variables by leveraging feasible LP duals [Hadjar et al., 2006, Baldacci et al., 2008, Bianchessi et al., 2024]. Its underlying principle is as follows: given a feasible LP dual and an upper bound on the optimal objective value, a nonnegative integer variable can be fixed to zero if its reduced cost exceeds the duality gap (i.e., the difference between the upper bound and the dual objective value). This approach preserves optimality while significantly reducing computational effort.

RCF can be utilized in various ways to reduce domains. For example, in

the context of vehicle routing, RCF enables techniques such as route enumeration [Baldacci et al., 2008], arc fixing [Hadjar et al., 2006], and resource window reduction [Bianchessi et al., 2024]. Specifically, Baldacci et al. [2008] propose enumerating all routes with reduced costs no greater than the duality gap, and then solving the set-partitioning formulation restricted to the enumerated routes directly by an IP solver, thereby eliminating the need to solve pricing problems at non-root nodes. Hadjar et al. [2006] and Irnich et al. [2010] suggest eliminating arcs whose reduced costs exceed the duality gap, which not only reduces the branch-and-bound tree but also accelerates the solution of pricing algorithms. More recently, Bianchessi et al. [2024] introduce a novel technique for reducing resource windows (e.g., time windows and capacity windows). Consider the VRPTW as an example: if the reduced costs of all routes capable of visiting a customer within a specific time interval exceed the duality gap, that time interval can be eliminated from the customer's time window without compromising optimality.

The domain reduction techniques mentioned above rely exclusively on feasible LP duals. While these techniques have demonstrated effectiveness in accelerating solution processes, further improvements can be achieved by leveraging Lagrangian duals, including those that are infeasible for the LP relaxation. Yang [2025] proposes variable fixing techniques that jointly utilize a Lagrangian dual and a feasible LP dual, where the feasible LP dual partitions the variables into two sets, and the Lagrangian dual is required to ensure nonnegative reduced costs for all variables in one of those sets. In this study, we derive new domain reduction techniques by jointly utilizing Lagrangian duals and subset sum inequalities, which frequently appear in integer programs. Compared to Yang [2025], our method can leverage a broader class of Lagrangian duals and does not require the computation of feasible LP duals.

The main contributions of this study are summarized as follows:

- We introduce LUCs for general integer programs, which guarantee that the sum of the reduced costs of all variables does not exceed the duality gap. The LUCs form a novel class of optimality cuts derived from Lagrangian duals, including infeasible LP duals.
- We demonstrate that LUCs can be utilized to strengthen RCCs, thereby improving their effectiveness in reducing the search space of integer programming formulations. This is because LUCs associated with feasible LP duals can be interpreted as knapsack constraints, enabling the derivation and lifting of (extended) cover inequalities such as RCCs.
- We employ LUCs in combination with subset sum inequalities to derive new domain reduction techniques. Compared to the approach proposed by Yang [2025], our method can leverage a broader class of Lagrangian duals and does not require the computation of feasible LP duals.
- We leverage LUCs to develop new arc fixing techniques for the VRPTW, incorporating Lagrangian duals with the tailored dual-picking strategy to maximize the number of eliminated arcs. Computational experiments demonstrate that, compared to the feasible-LP-dual approach studied in the literature, our method achieves significant domain reductions: the number of remaining arcs is reduced to an average of 47.9%, while the number of enumerated routes decreases to an average of 31.3%. For some instances, the route count drops from millions to around ten. These results highlight the effectiveness of the proposed techniques in enhancing arc fixing and column enumeration for the VRPTW.

In the remainder of this chapter, we introduce LUCs in Section 3.2. We then

utilize LUCs to strengthen RCCs in Section 3.3, to derive new domain reduction techniques in Section 3.4, and to develop new arc fixing methods for the VRPTW in Section 3.5. The chapter is summarized in Section 3.6.

3.2 Lagrangian Underestimate Cuts

In this section, we introduce the LUCs for general IP formulations. Let $\mathbb{Z}_+$ denote the set of nonnegative integers. Define $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$ as the sets of constraint indices, $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3$, and $\mathcal{J}$ as the set of variable indices. Consider the following IP formulation, where $b_i, c_j, f_{ij}, i \in \mathcal{I}, j \in \mathcal{J}$ are parameters, and $\boldsymbol{x}$ represents the vector of nonnegative integer decision variables.

$$\min \sum_{j \in \mathcal{J}} c_j x_j, \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} f_{ij} x_j = b_i, \quad \forall i \in \mathcal{I}_1, \tag{3.1b}$$

$$\sum_{j \in \mathcal{J}} f_{ij} x_j \leq b_i, \quad \forall i \in \mathcal{I}_2, \tag{3.1c}$$

$$\sum_{j \in \mathcal{J}} f_{ij} x_j \geq b_i, \quad \forall i \in \mathcal{I}_3, \tag{3.1d}$$

$$x_j \in \mathbb{Z}_+, \quad \forall j \in \mathcal{J}. \tag{3.1e}$$

Let UB be an upper bound on the optimal objective value. Denote $\mathcal{X}$ as the set of primal feasible solutions and $\mathcal{X}^*$ as the set of primal optimal solutions. Define $\Pi = \{\boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{I}|} : \pi_i \leq 0, \forall i \in \mathcal{I}_2; \pi_i \geq 0, \forall i \in \mathcal{I}_3\}$ as the set of Lagrangian dual solutions, where sign constraints on dual variables are satisfied. For any $\boldsymbol{\pi} \in \Pi$, the dual objective value is given by $\boldsymbol{\pi}^T \boldsymbol{b}$. For variable x_j where $j \in \mathcal{J}$, the reduced cost w.r.t. dual $\boldsymbol{\pi}$ is $c'_j(\boldsymbol{\pi}) = c_j - \sum_{i \in \mathcal{I}} \pi_i f_{ij}$.

Proposition 11 (LUC)**.** *For any $\boldsymbol{\pi} \in \Pi$, the following cut holds for all $\boldsymbol{x} \in \mathcal{X}^*$.*

$$\boldsymbol{\pi}^T \boldsymbol{b} + \sum_{j \in \mathcal{J}} c'_j(\boldsymbol{\pi}) x_j \leq UB. \tag{3.2}$$

Proof. By dualizing constraints (3.1b)–(3.1d), we obtain the Lagrangian function where $\boldsymbol{F} = \{f_{ij} : i \in \mathcal{I}, j \in \mathcal{J}\}$ is the coefficient matrix:

$$L(\boldsymbol{\pi}, \boldsymbol{x}) = \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{\pi}^T (\boldsymbol{b} - \boldsymbol{F}\boldsymbol{x}) = \boldsymbol{\pi}^T \boldsymbol{b} + (\boldsymbol{c}^T - \boldsymbol{\pi}^T \boldsymbol{F})\boldsymbol{x} = \boldsymbol{\pi}^T \boldsymbol{b} + \sum_{j \in \mathcal{J}} c'_j(\boldsymbol{\pi}) x_j.$$

For any $\boldsymbol{\pi} \in \Pi$ and $\boldsymbol{x} \in \mathcal{X}^*$, we have $\boldsymbol{\pi}^T(\boldsymbol{b} - \boldsymbol{F}\boldsymbol{x}) \leq 0$, which implies that

$$\boldsymbol{\pi}^T \boldsymbol{b} + \sum_{j \in \mathcal{J}} c'_j(\boldsymbol{\pi}) x_j \leq \boldsymbol{c}^T \boldsymbol{x}. \tag{3.3}$$

Moreover, $\boldsymbol{c}^T \boldsymbol{x} \leq UB$ holds for $\boldsymbol{x} \in \mathcal{X}^*$, which, together with (3.3), completes the proof. □

Expression (3.2) is an optimality cut satisfied by all optimal solutions. We call it a Lagrangian underestimate cut, since it is derived from the Lagrangian dual $\boldsymbol{\pi} \in \Pi$, and $L(\boldsymbol{\pi}, \boldsymbol{x}) = \boldsymbol{\pi}^T \boldsymbol{b} + \sum_{j \in \mathcal{J}} c'_j(\boldsymbol{\pi}) x_j$ serves as an underestimate on $\boldsymbol{c}^T \boldsymbol{x}$ for any $\boldsymbol{x} \in \mathcal{X}$.

3.3 Utilizing LUCs to Strengthen Reduced Cost Cuts

In this section, we derive and lift the RCCs introduced in Yang [2023] by interpreting the LUCs associated with feasible LP duals as knapsack constraints. Let $\mathbb{Z}_{++}$ denote the set of positive integers, and denote $\hat{\Pi} = \{\boldsymbol{\pi} \in \Pi : c'_j(\boldsymbol{\pi}) \geq 0, \forall j \in \mathcal{J}\}$ as the

set of feasible LP dual solutions. For a positive integer $m \in \mathbb{Z}_{++}$ and a feasible LP dual $\boldsymbol{\pi} \in \hat{\Pi}$, define $\mathcal{J}_m(\boldsymbol{\pi}) = \{j \in \mathcal{J} : c'_j(\boldsymbol{\pi}) > \frac{UB - \boldsymbol{\pi}^T \boldsymbol{b}}{m}\}$ as the set of indices corresponding to variables with reduced costs greater than $\frac{UB - \boldsymbol{\pi}^T \boldsymbol{b}}{m}$. For any $\boldsymbol{x} \in \mathcal{X}^*$ and $\boldsymbol{\pi} \in \hat{\Pi}$, the following LUC holds:

$$\sum_{j \in \mathcal{J}} c'_j(\boldsymbol{\pi}) x_j \leq UB - \boldsymbol{\pi}^T \boldsymbol{b}. \tag{3.4}$$

Since $c'_j(\boldsymbol{\pi}) \geq 0$ for all $j \in \mathcal{J}$, LUC (3.4) is indeed a knapsack constraint. This, together with $c'_j(\boldsymbol{\pi}) > \frac{UB - \boldsymbol{\pi}^T \boldsymbol{b}}{m}$ for all $j \in \mathcal{J}_m(\boldsymbol{\pi})$, implies the following RCC:

$$\sum_{j \in \mathcal{J}_m(\boldsymbol{\pi})} x_j \leq m - 1. \tag{3.5}$$

Remark 1. *Yang [2023] establishes RCCs by constructing a feasible dual solution for an auxiliary optimization problem and utilizing the LP duality theory. In contrast, this study derives RCCs directly by treating LUCs as knapsack constraints, providing a simpler and more intuitive method for understanding RCCs.*

If $x_j, j \in \mathcal{J}$ are binary variables, i.e., $\boldsymbol{x} \in \{0,1\}^{|\mathcal{J}|}$, then the RCC (3.5) is a cover inequality [Bektas and Oğuz, 2007] when $|\mathcal{J}_m(\boldsymbol{\pi})| = m$ and $m \in \mathbb{Z}_{++} \setminus \{1\}$. Moreover, when $|\mathcal{J}_m(\boldsymbol{\pi})| > m$ and $m \in \mathbb{Z}_{++} \setminus \{1\}$, the RCC (3.5) becomes an extended cover inequality [Büsing et al., 2025], which can be further strengthened through lifting procedures. For example, the lifting procedure described in Section 9.3.2 of Wolsey [2020] can be adapted by treating the LUC (3.4) as a knapsack constraint, as outlined in Algorithm 2. In this algorithm, for $\mathcal{J}_m(\boldsymbol{\pi}) = \{1, 2, \ldots, q\}$ with $q > m$, we reorder the variables $x_1, x_2, \ldots, x_q$ so that $c'_1(\boldsymbol{\pi}) \leq c'_2(\boldsymbol{\pi}) \leq \cdots \leq c'_q(\boldsymbol{\pi})$. In line 1, the coefficients of the first m variables are initialized to 1. The coefficients of the remaining variables, $x_{m+1}, \ldots, x_q$, are then determined sequentially (lines 2–4). The algorithm outputs the valid cut $\sum_{j \in \mathcal{J}_m(\boldsymbol{\pi})} \alpha_j x_j \leq m-1$.

It can be shown that $\alpha_j \geq 1$ for all $j \in \mathcal{J}_m(\boldsymbol{\pi})$, which implies that the resulting cut can be tighter than the original RCC.

Algorithm 2 Lifting the RCC

Input: $\mathcal{J}_m(\boldsymbol{\pi}) = \{1, 2, \ldots, q\}$, where $m \in \mathbb{Z}_{++} \setminus \{1\}$, $\boldsymbol{\pi} \in \hat{\Pi}$, $q > m$, and $c'_1(\boldsymbol{\pi}) \leq c'_2(\boldsymbol{\pi}) \leq \cdots \leq c'_q(\boldsymbol{\pi})$

Output: A valid cut $\sum_{j \in \mathcal{J}_m(\boldsymbol{\pi})} \alpha_j x_j \leq m - 1$, where $\alpha_j \geq 1$ for all $j \in \mathcal{J}_m(\boldsymbol{\pi})$

1: Set $\alpha_1 = \alpha_2 = \cdots = \alpha_m = 1$
2: **for** $p = m + 1 \to q$ **do**
3: Solve $\zeta_p = \max\{\sum_{j=1}^{p-1} \alpha_j x_j : \sum_{j=1}^{p-1} c'_j(\boldsymbol{\pi}) x_j \leq UB - \boldsymbol{\pi}^T \boldsymbol{b} - c'_p(\boldsymbol{\pi}), \boldsymbol{x} \in \{0,1\}^{p-1}\}$
4: Set $\alpha_p = m - 1 - \zeta_p$

For instance, consider the set $\mathcal{J} = \{1, 2, 3, 4\}$, with $UB - \boldsymbol{\pi}^T \boldsymbol{b} = 2.9$, $c'_1(\boldsymbol{\pi}) = c'_2(\boldsymbol{\pi}) = c'_3(\boldsymbol{\pi}) = 1$, and $c'_4(\boldsymbol{\pi}) = 2.9$. For $m = 3$, the corresponding RCC is given by $x_1 + x_2 + x_3 + x_4 \leq 2$. However, by applying Algorithm 2, the RCC can be lifted to $x_1 + x_2 + x_3 + 2x_4 \leq 2$. By lifting the coefficients of RCCs, we can obtain tighter LP relaxations and reduce the solution space, potentially accelerating the solution process of IP formulations.

3.4 Utilizing LUCs and Subset Sum Inequalities for Domain Reduction

In this section, we derive new domain reduction techniques for decision variables by jointly utilizing LUCs and subset sum inequalities. For a positive integer $k \in \mathbb{Z}_{++}$ and a subset $\mathcal{S} \subseteq \mathcal{J}$, the corresponding subset sum inequality imposes that the sum of variables $x_j, j \in \mathcal{S}$ must not exceed k, i.e.,

$$\sum_{j \in \mathcal{S}} x_j \leq k. \tag{3.6}$$

Subset sum inequalities can arise from feasibility conditions (e.g., limiting the number of available vehicles to K in vehicle routing problems) or optimality

conditions such as RCCs. Given $\boldsymbol{\pi} \in \Pi$ such that $c'_j(\boldsymbol{\pi}) \geq 0$ for all $j \in \mathcal{J} \setminus \mathcal{S}$, the inequality (3.6) enables the computation of a lower bound on $\boldsymbol{\pi}^T\boldsymbol{b} + \sum_{j\in\mathcal{J}} c'_j(\boldsymbol{\pi})x_j$, i.e., the left-hand-side of the LUC. Combined with the optimality conditions implied by LUCs, this can be leveraged to develop domain reduction techniques as established in Proposition 12.

Proposition 12. *Assume that inequality (3.6) holds. Let $\boldsymbol{x} \in \mathcal{X}^*$ and $\boldsymbol{\pi} \in \Pi$ where $c'_j(\boldsymbol{\pi}) \geq 0$ for all $j \in \mathcal{J} \setminus \mathcal{S}$. Then, the following statements hold:*

(a) If $h \in \mathcal{S}$ and $c'_h(\boldsymbol{\pi}) > (UB - \boldsymbol{\pi}^T\boldsymbol{b})/k$, then $x_h \leq k - 1$.

(b) If $h \in \mathcal{S}$, $x_h \in \{0, 1\}$, and $c'_h(\boldsymbol{\pi}) + (k-1)\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S} \setminus \{h\}\}\} > UB - \boldsymbol{\pi}^T\boldsymbol{b}$, then $x_h = 0$.

(c) If $h \in \mathcal{S}$, $c'_h(\boldsymbol{\pi}) \geq 0$, and $c'_h(\boldsymbol{\pi}) + (k-1)\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S} \setminus \{h\}\}\} > UB - \boldsymbol{\pi}^T\boldsymbol{b}$, then $x_h = 0$.

(d) If $h \in \mathcal{J} \setminus \mathcal{S}$ and $c'_h(\boldsymbol{\pi}) + k\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S}\}\} > UB - \boldsymbol{\pi}^T\boldsymbol{b}$, then $x_h = 0$.

Proof. Consider an optimal primal solution $\boldsymbol{x} \in \mathcal{X}^*$ and a Lagrangian dual solution $\boldsymbol{\pi} \in \Pi$ where $c'_j(\boldsymbol{\pi}) \geq 0$ for all $j \in \mathcal{J} \setminus \mathcal{S}$. It follows that $c'_j(\boldsymbol{\pi})x_j \geq 0$ for all $j \in \mathcal{J} \setminus \mathcal{S}$. We next prove the statements by contradiction as follows.

(a) By contradiction, suppose that $x_h > k - 1$. Since x_h is an integer and inequality (3.6) holds, we have $x_h = k$ and $x_j = 0$ for all $j \in \mathcal{S} \setminus \{h\}$. It follows that $\boldsymbol{\pi}^T\boldsymbol{b} + \sum_{j\in\mathcal{J}} c'_j(\boldsymbol{\pi})x_j \geq \boldsymbol{\pi}^T\boldsymbol{b} + c'_h(\boldsymbol{\pi})x_h > UB$.

(b) By contradiction, suppose that $x_h \neq 0$. Since x_h is binary and inequality (3.6) holds, we have $x_h = 1$ and $\sum_{j\in\mathcal{S}\setminus\{h\}} x_j \leq k - 1$. It follows that $\sum_{j\in\mathcal{S}\setminus\{h\}} c'_j(\boldsymbol{\pi})x_j \geq (k-1)\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S} \setminus \{h\}\}$. Therefore, we

have $\boldsymbol{\pi}^T\boldsymbol{b} + \sum_{j\in\mathcal{J}} c'_j(\boldsymbol{\pi})x_j \geq \boldsymbol{\pi}^T\boldsymbol{b} + c'_h(\boldsymbol{\pi}) + (k-1)\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S}\setminus\{h\}\}\} > UB$.

(c) By contradiction, suppose that $x_h \neq 0$. Since $x_h \in \mathbb{Z}_+$ and inequality (3.6) holds, we have $x_h \geq 1$ and $\sum_{j\in\mathcal{S}\setminus\{h\}} x_j \leq k-1$. It follows that $\sum_{j\in\mathcal{S}\setminus\{h\}} c'_j(\boldsymbol{\pi})x_j \geq (k-1)\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S}\setminus\{h\}\}$. Moreover, $c'_h(\boldsymbol{\pi}) \geq 0$ and $x_h \geq 1$ imply that $c'_h(\boldsymbol{\pi})x_h \geq c'_h(\boldsymbol{\pi})$. Therefore, we have $\boldsymbol{\pi}^T\boldsymbol{b} + \sum_{j\in\mathcal{J}} c'_j(\boldsymbol{\pi})x_j \geq \boldsymbol{\pi}^T\boldsymbol{b} + c'_h(\boldsymbol{\pi}) + (k-1)\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S}\setminus\{h\}\}\} > UB$.

(d) By contradiction, suppose that $x_h \neq 0$. Since $x_h \in \mathbb{Z}_+$, we have $x_h \geq 1$. This, together with $h \in \mathcal{J}\setminus\mathcal{S}$, implies that $c'_h(\boldsymbol{\pi})x_h \geq c'_h(\boldsymbol{\pi})$. Moreover, since inequality (3.6) holds, it follows that $\sum_{j\in\mathcal{S}} c'_j(\boldsymbol{\pi})x_j \geq k\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S}\}\}$. Therefore, we have $\boldsymbol{\pi}^T\boldsymbol{b} + \sum_{j\in\mathcal{J}} c'_j(\boldsymbol{\pi})x_j \geq \boldsymbol{\pi}^T\boldsymbol{b} + c'_h(\boldsymbol{\pi}) + k\min\{0, \min\{c'_l(\boldsymbol{\pi}) : l \in \mathcal{S}\}\} > UB$.

However, by Proposition 11, we know that $\boldsymbol{\pi}^T\boldsymbol{b} + \sum_{j\in\mathcal{J}} c'_j(\boldsymbol{\pi})x_j \leq UB$. This leads to contradictions for all the above statements, thereby proving that all these assumptions are false and completing the proof. □

Since $\sum_{j\in\mathcal{J}_2(\hat{\boldsymbol{\pi}})} x_j \leq 1$ holds for $\boldsymbol{x} \in \mathcal{X}^*$ and $\hat{\boldsymbol{\pi}} \in \hat{\Pi}$, the following Proposition 13 is a special case of Proposition 12(a) for $\mathcal{S} = \mathcal{J}_2(\hat{\boldsymbol{\pi}})$ and $k = 1$. Moreover, when $\mathcal{S} = \mathcal{J}_2(\hat{\boldsymbol{\pi}})$ and $h \in \mathcal{S}$, x_h must be binary, it follows that Proposition 13 is also a special case of Proposition 12(b) for $\mathcal{S} = \mathcal{J}_2(\hat{\boldsymbol{\pi}})$ and $k = 1$.

Proposition 13 (Yang 2025). *Let $\boldsymbol{x} \in \mathcal{X}^*$, $\boldsymbol{\pi} \in \Pi$, and $\hat{\boldsymbol{\pi}} \in \hat{\Pi}$. If $c'_j(\boldsymbol{\pi}) \geq 0$ for all $j \in \mathcal{J}\setminus\mathcal{J}_2(\hat{\boldsymbol{\pi}})$, $h \in \mathcal{J}_2(\hat{\boldsymbol{\pi}})$, and $c'_h(\boldsymbol{\pi}) > UB - \boldsymbol{\pi}^T\boldsymbol{b}$, then $x_h = 0$.*

In comparison to Proposition 13, a broader range of Lagrangian dual solutions can be leveraged by Proposition 12 to reduce variable domains. For instance, these

dual solutions can be obtained by setting $\mathcal{S} = \mathcal{J}_m(\hat{\boldsymbol{\pi}})$ and $k = m - 1$, where $\hat{\boldsymbol{\pi}} \in \hat{\Pi}$ and $m \in \mathbb{Z}_{++} \setminus \{1\}$. In the context of vehicle routing, several additional approaches can be employed: (i) Let $\mathcal{S}$ correspond to the set of all routes and set k to the number of available vehicles K. In this case, $\mathcal{J} \setminus \mathcal{S} = \emptyset$ and all Lagrangian duals are applicable to Proposition 12; (ii) Let $\mathcal{S}$ correspond to the set of all routes visiting a specific customer and set $k = 1$; (iii) Let $\mathcal{S}$ correspond to the set of all routes traversing a specific arc and set $k = 1$. Furthermore, even when a variable has not been fixed to zero using existing variable fixing techniques, it is still possible to reduce its domain by Proposition 12(a) for cases where $k > 1$.

3.5 Utilizing LUCs for Arc Fixing in VRPTW

In this section, we leverage LUCs to develop novel arc fixing techniques for the VRPTW that reduce the underlying network size while preserving optimality. We introduce a dual-picking method designed to maximize the number of eliminated arcs by exploiting Lagrangian dual solutions, including infeasible LP duals. In contrast, existing literature [Hadjar et al., 2006, Irnich et al., 2010, Desaulniers et al., 2020] has relied on feasible LP duals obtained via column generation.

The VRPTW aims to determine at most K routes that minimize the total travel cost, ensuring each customer is visited exactly once [Desrochers et al., 1992]. A route is a simple cycle that starts and ends at the depot, visiting a subset of customers while satisfying both capacity constraints and time window constraints [Ben Ticha et al., 2017]. Let $\mathcal{R}$ denote the set of all routes, whose cardinality may increase exponentially in the problem size. For each route $r \in \mathcal{R}$, the parameter c_r represents the travel cost, and the binary variable y_r equals 1 if and only if route r

is selected in the solution. The VRPTW can be formulated as follows:

$$
\begin{aligned}
(P) \qquad & \min \sum_{r \in \mathcal{R}} c_r y_r, && && \text{(3.7a)}\\
\text{s.t.} \qquad & \sum_{r \in \mathcal{R}} f_{ir} y_r = b_i, && \forall i \in \mathcal{I}_1, && \text{(3.7b)}\\
& \sum_{r \in \mathcal{R}} f_{ir} y_r \leq b_i, && \forall i \in \mathcal{I}_2, && \text{(3.7c)}\\
& \sum_{r \in \mathcal{R}} f_{ir} y_r \geq b_i, && \forall i \in \mathcal{I}_3, && \text{(3.7d)}\\
& y_r \in \{0,1\}, && \forall r \in \mathcal{R}. && \text{(3.7e)}
\end{aligned}
$$

The objective function (3.7a) minimizes the total travel cost. Constraints (3.7b) guarantee that each customer is visited exactly once. The less-than-or-equal-to constraints (3.7c) include the fleet size constraints (i.e., $\sum_{r \in \mathcal{R}} y_r \leq K$) and classes of valid inequalities such as subset-row cuts [Jepsen et al., 2008]. The greater-than-or-equal-to constraints (3.7d) encompass inequalities such as rounded capacity cuts [Dalmeijer and Spliet, 2018]. Finally, constraints (3.7e) specify the domain of decision variables.

Let $\mathcal{A}$ denote the set of arcs. For arc $a \in \mathcal{A}$ and route $r \in \mathcal{R}$, the binary parameter g_{ar} equals 1 if and only if arc a is traversed by route r. Define $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}^{|\mathcal{I}|} : \lambda_i \leq 0, \forall i \in \mathcal{I}_2; \lambda_i \geq 0, \forall i \in \mathcal{I}_3\}$ as the set of Lagrangian dual solutions to P. For a given dual $\boldsymbol{\lambda} \in \Lambda$, the reduced cost of the variable y_r is $\hat{c}_r(\boldsymbol{\lambda}) = c_r - \sum_{i \in \mathcal{I}} \lambda_i f_{ir}$.

3.5.1 Arc Fixing Using LP Feasible Duals

The arc fixing technique explored in the existing literature operates as Algorithm 3. The first step is to obtain a feasible LP dual $\boldsymbol{\lambda}$, where $\hat{c}_r(\boldsymbol{\lambda}) \geq 0$ for all $r \in \mathcal{R}$, by solving the LP relaxation using column generation. The second step is to calculate $\min\{\hat{c}_r(\boldsymbol{\lambda}) : r \in \mathcal{R}, g_{ar} = 1\}$. According to RCF, if $\min\{\hat{c}_r(\boldsymbol{\lambda}) : r \in \mathcal{R}, g_{ar} = 1\} >$

$UB - \boldsymbol{\lambda}^T \boldsymbol{b}$, then no route in $\{r \in \mathcal{R} : g_{ar} = 1\}$ can be part of an optimal solution. It follows that arc a cannot be selected in any optimal solution and can be eliminated from the underlying network without compromising optimality.

Algorithm 3 Arc Fixing Using LP Feasible Duals

1: Solve the LP relaxation of (3.7) to obtain a feasible LP dual $\boldsymbol{\lambda}$
2: Solve $\min\{\hat{c}_r(\boldsymbol{\lambda}) : r \in \mathcal{R}, g_{ar} = 1\}$
3: **if** $\min\{\hat{c}_r(\boldsymbol{\lambda}) : r \in \mathcal{R}, g_{ar} = 1\} > UB - \boldsymbol{\lambda}^T \boldsymbol{b}$ **then**
4: └ Eliminate arc a

3.5.2 LUC-Based Arc Fixing

Arc fixing techniques studied in the literature rely on the feasible LP dual obtained by solving the LP relaxation of (3.7) using column generation. However, the reduced costs of a variable can differ significantly across various dual solutions. By leveraging LUCs, we are going to propose a strategy for selecting alternative dual solutions to maximize the number of arcs that can be eliminated. To derive the LUCs involving arc $a \in \mathcal{A}$, we incorporate a redundant constraint (3.8b) linking $\boldsymbol{y}$ and z_a, where z_a is a binary variable equal to 1 if and only if arc a is selected in the solution.

$$
\begin{aligned}
(P(a)) \qquad & \min \sum_{r \in \mathcal{R}} c_r y_r, && \text{(3.8a)} \\
\text{s.t.} \qquad & \text{(3.7b)–(3.7e)}, \\
& \sum_{r \in \mathcal{R}} g_{ar} y_r - z_a = 0, && \text{(3.8b)} \\
& z_a \in \{0, 1\}. && \text{(3.8c)}
\end{aligned}
$$

Introducing the linking constraint (3.8b) does not alter the feasibility or optimality of formulation (3.8). Specifically, there is a one-to-one correspondence between solutions: if $\boldsymbol{y}$ is a feasible (or optimal) solution to P, then $(\boldsymbol{y}, z_a(\boldsymbol{y}))$ is a feasible (or optimal) solution to $P(a)$, where $z_a(\boldsymbol{y}) = \sum_{r \in \mathcal{R}} g_{ar} y_r$. The converse

also holds.

Define $\Pi = \{(\boldsymbol{\lambda}, \mu_a) : \boldsymbol{\lambda} \in \Lambda, \mu_a \in \mathbb{R}\}$ as the set of Lagrangian dual solutions to $P(a)$, where μ_a is the dual variable associated with linking constraint (3.8b). For dual $(\boldsymbol{\lambda}, \mu_a) \in \Pi$, the reduced cost of y_r is $c'_r(\boldsymbol{\lambda}, \mu_a) = \hat{c}_r(\boldsymbol{\lambda}) - g_{ar}\mu_a$, and the reduced cost of variable z_a is $\bar{c}_a(\boldsymbol{\lambda}, \mu_a) = \mu_a$. Then, for any $(\boldsymbol{\lambda}, \mu_a) \in \Pi$ and any optimal solution $(\boldsymbol{y}, z_a)$ to formulation (3.8), the corresponding LUC (3.9) holds in accordance with Proposition 11.

$$\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a z_a + \sum_{r \in \mathcal{R}} c'_r(\boldsymbol{\lambda}, \mu_a) y_r \leq UB. \tag{3.9}$$

For any dual $(\boldsymbol{\lambda}, \mu_a) \in \Pi$, if $\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a + \sum_{r \in \mathcal{R}} c'_r(\boldsymbol{\lambda}, \mu_a) y_r > UB$ holds for all feasible solutions to $P(a)$, then any feasible solution $(\boldsymbol{y}, z_a)$ with $z_a = 1$ must violate LUC (3.9) and cannot be optimal. Consequently, arc a can be eliminated without compromising optimality. Motivated by this observation, we formulate the following dual-picking problem $DP(a)$, which maximizes $\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a$ subject to a lower bound on $c'_r(\boldsymbol{\lambda}, \mu_a)$ for $r \in \mathcal{R}$. Since $c'_r(\boldsymbol{\lambda}, \mu_a)$ is linear and Π is a polyhedron, $DP(a)$ is an LP.

$$DP(a) \qquad \max \boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a, \tag{3.10a}$$

$$\text{s.t.} \qquad c'_r(\boldsymbol{\lambda}, \mu_a) \geq 0, \quad \forall r \in \mathcal{R}, \tag{3.10b}$$

$$(\boldsymbol{\lambda}, \mu_a) \in \Pi. \tag{3.10c}$$

The objective function (3.10a) maximizes the sum of the dual objective (i.e., $\boldsymbol{\lambda}^T \boldsymbol{b}$) and the reduced cost of arc a (i.e., μ_a). Constraints (3.10b) require that the reduced costs of all routes are nonnegative. Constraints (3.10c) specify the domain of decision variables. The solutions of $DP(a)$ can be leveraged to fix

arc a as established in Proposition 14, where $\delta = \min\{c'_r(\boldsymbol{\lambda}, \mu_a) : r \in \mathcal{R}\}$ and $\delta^- = \min\{0, \delta\}$.

Proposition 14. *Let* $(\boldsymbol{\lambda}, \mu_a) \in \Pi$. *If* $\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a + K\delta^- > UB$, *then* $z_a = 0$ *holds for any optimal solution* $(\boldsymbol{y}, z_a)$.

Proof. Since z_a is a binary variable and at most K routes can be selected (i.e., $\sum_{r\in\mathcal{R}} y_r \leq K$), the inequality $z_a + \sum_{r\in\mathcal{R}} y_r \leq K + 1$ holds trivially. Thus, Proposition 14 follows directly from Proposition 12(b) by setting $\mathcal{J} = \mathcal{S} = \{a\} \cup \mathcal{R}$, $h = a$, and $k = K + 1$. □

Notably, compared to arc fixing techniques in the literature, Proposition 14 leverages a broader class of duals, including infeasible LP duals. This property permits the early termination of the solution process for $DP(a)$ once the eliminability of the arc is established. Since the number of constraints (3.10b) can grow exponentially with the problem size, we solve $DP(a)$ using a cut generation approach (Algorithm 4), where the constraints (3.10b) are added incrementally. At each iteration of Algorithm 4, we solve a relaxed version of the dual-picking problem that includes only a subset of the constraints (3.10b). Consider an optimal solution $(\boldsymbol{\lambda}, \mu_a)$ to the relaxed problem. If $\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a \leq UB$, we terminate the cut generation process without eliminating arc a. This early termination is justified because $\min\{c'_r(\boldsymbol{\lambda}, \mu_a) : r \in \mathcal{R}\} \leq 0$ and the value of $\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a$ is non-increasing throughout the process, making the elimination condition still unsatisfied in all subsequent iterations. Conversely, if $\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a > UB$, we solve the subproblem $\delta = \min\{c'_r(\boldsymbol{\lambda}, \mu_a) : r \in \mathcal{R}\}$, which corresponds to an ESPPRC [Lozano et al., 2016]. If $\boldsymbol{\lambda}^T \boldsymbol{b} + \mu_a + K\delta > UB$, we terminate the cut generation process and eliminate arc a according to Proposition 14. Otherwise, we add the constraint $c'_r(\boldsymbol{\lambda}, \mu_a) \geq 0$ to the relaxed problem and repeat the iteration, where r is the route with the minimum reduced cost.

Algorithm 4 Cut Generation for $DP(a)$

```
Let R' ← ∅
Let Stop ← False
while Stop = False do
    Solve the relaxed problem of (3.10) where R is replaced with R', and obtain the optimal solution (λ, μ_a)
    if λ^T b + μ_a ≤ UB then
        Set Stop ← True
    else
        Solve the subproblem δ = min{c'_r(λ, μ_a) : r ∈ R}. Denote r' as the route with the minimum reduced cost
        if λ^T b + μ_a + Kδ > UB then
            Eliminate arc a according to Proposition 14
            Set Stop ← True
        else
            Let R' ← R' ∪ {r'}
```

3.5.3 Optimal Value of μ_a

Given a Lagrangian dual $\boldsymbol{\lambda} \in \Lambda$, we seek to determine the value of μ_a that maximizes $\mu_a + K\delta^-$. Let $\beta = \min\{\hat{c}_r(\boldsymbol{\lambda}) : r \in \mathcal{R}, g_{ar} = 1\}$, $\gamma = \min\{\hat{c}_r(\boldsymbol{\lambda}) : r \in \mathcal{R}, g_{ar} = 0\}$, and $\gamma^- = \min\{0, \gamma\}$. The values of β and γ can be computed by running additional labeling algorithms for the ESPPRC [Lozano et al., 2016]. Proposition 15 demonstrates that $\mu_a = \beta - \gamma^-$ is an optimal value for maximizing $\mu_a + K\delta^-$.

Proposition 15. *For a given $\boldsymbol{\lambda} \in \Lambda$, the maximization problem $\max\{\mu_a + K\delta^- : \mu_a \in \mathbb{R}\}$ yields an optimal value of $\beta + (K-1)\gamma^-$, which can be attained at $\mu_a = \beta - \gamma^-$.*

Proof. By the definition of δ^-, we have $\delta^- = \min\{0, \beta - \mu_a, \gamma\}$. It follows that $\mu_a + K\delta^- = \mu_a + K\min\{0, \beta - \mu_a, \gamma\} = \min\{\mu_a, \mu_a + K(\beta - \mu_a), \mu_a + K\gamma\}$.

If $\gamma \geq 0$, then $\min\{\mu_a, \mu_a + K(\beta - \mu_a), \mu_a + K\gamma\} = \min\{\mu_a, \mu_a + K(\beta - \mu_a)\}$. It follows that $\max\{\mu_a + K\delta^- : \mu_a \in \mathbb{R}\} = \max_{\mu_a \in \mathbb{R}} \min\{\mu_a, \mu_a + K(\beta - \mu_a)\} = \beta$,

and the maximum value can be attained at $\mu_a = \beta$.

If $\gamma < 0$, then $\min\{\mu_a, \mu_a+K(\beta-\mu_a), \mu_a+K\gamma\} = \min\{\mu_a+K(\beta-\mu_a), \mu_a+K\gamma\}$. It follows that $\max\{\mu_a+K\delta^- : \mu_a \in \mathbb{R}\} = \max_{\mu_a\in\mathbb{R}} \min\{\mu_a+K(\beta-\mu_a), \mu_a+K\gamma\} = \beta + (K-1)\gamma$, and the maximum value can be attained at $\mu_a = \beta - \gamma$.

By the definition $\gamma^- = \min\{0, \gamma\}$, we can unify these results. Given a Lagrangian dual $\boldsymbol{\lambda} \in \Lambda$, the maximization problem $\max\{\mu_a + K\delta^- : \mu_a \in \mathbb{R}\}$ yields an optimal value of $\beta + (K-1)\gamma^-$, and an optimal solution for this maximization problem is $\mu_a = \beta - \gamma^-$. This completes the proof. □

Given a Lagrangian dual $\boldsymbol{\lambda} \in \Lambda$, Corollary 1 provides a sufficient condition for eliminating arc a. However, applying this corollary introduces an additional computational burden, as it requires calculating the values of β and γ.

Corollary 1. *Let $\boldsymbol{\lambda} \in \Lambda$. If $\boldsymbol{\lambda}^T\boldsymbol{b} + \beta + (K-1)\gamma^- > UB$, then $z_a = 0$ holds for any optimal solution $(\boldsymbol{y}, z_a)$.*

Proof. Given $\boldsymbol{\lambda} \in \Lambda$, we construct a dual solution $(\boldsymbol{\lambda}, \mu_a) \in \Pi$ by setting $\mu_a = \beta - \gamma^-$. According to Proposition 15, we have $\boldsymbol{\lambda}^T\boldsymbol{b} + \mu_a + K\delta^- = \boldsymbol{\lambda}^T\boldsymbol{b} + \beta + (K-1)\gamma^- > UB$. Consequently, Proposition 14 implies that any optimal solution must satisfy $z_a = 0$. This completes the proof. □

3.5.4 Computational Experiments on Arc Fixing

In this study, we apply arc fixing techniques to the VRPTW through the following procedure. First, we solve the LP relaxation via column-and-cut generation and perform arc fixing using LP feasible duals (Section 3.5.1). Next, we employ the LUC-based dual picking method to further eliminate arcs (Section 3.5.2). Subsequently, we enumerate a set of routes that are possibly selected in optimal solutions according to RCF. Finally, we solve the restricted problem over these enumerated routes

using an IP solver. The technique introduced in Section 3.5.3 is not applied, since it introduces an additional computational burden that can render the arc fixing process more time-consuming.

Following Desaulniers et al. [2020] and Bianchessi et al. [2024], we set the upper bound as $UB = opt + 1$, where opt is the optimal objective value. Subset-row cuts and rounded capacity cuts are incorporated into the cut generation process, with the number of subset-row cuts limited to 32 to improve the efficiency of the labeling algorithm for solving the ESPPRC. Computational results on 36 Solomon instances with 100 customers are shown in Table 3.1. In this table, $|\mathcal{A}|$ is the number of arcs remaining after applying the time window reduction technique proposed by Desrochers et al. [1992]. The column "FeasPick" corresponds to the algorithm presented in the preceding paragraph. The column "FeasOnly" refers to the modified algorithm that exclusively applies arc fixing using feasible LP duals, while excluding the LUC-based dual picking method. "Feas" and "Pick" represent the number of arcs eliminated through feasible LP duals and the LUC-based dual picking method, respectively. Additionally, $|\mathcal{A}_T|$ and $|\mathcal{A}_F|$ indicate the number of arcs that remain uneliminated, while $|\mathcal{R}_T|$ and $|\mathcal{R}_F|$ denote the number of routes enumerated. The column $|\mathcal{A}_T|/|\mathcal{A}_F|$ represents the ratio between $|\mathcal{A}_T|$ and $|\mathcal{A}_F|$, while $|\mathcal{R}_T|/|\mathcal{R}_F|$ denotes the ratio between $|\mathcal{R}_T|$ and $|\mathcal{R}_F|$.

Table 3.1 demonstrates that the LUC-based dual picking method is consistently effective for arc fixing across all these instances. The average ratio $|\mathcal{A}_T|/|\mathcal{A}_F|$ across these instances is 47.9%, indicating that compared with the modified algorithm that exclusively applies arc fixing using feasible LP duals, incorporating the LUC-based dual picking method significantly reduces the number of remaining arcs. This reduction has the potential to accelerate branch-and-price methods that rely on branching over arc variables.

Table 3.1: Computational results on arc fixing and route enumeration for VRPTW.

Instance	$\|\mathcal{A}\|$	FeasPick				FeasOnly		Ratios (%)	
		Feas	Pick	$\|\mathcal{A}_T\|$	$\|\mathcal{R}_T\|$	$\|\mathcal{A}_F\|$	$\|\mathcal{R}_F\|$	$\|\mathcal{A}_T\|/\|\mathcal{A}_F\|$	$\|\mathcal{R}_T\|/\|\mathcal{R}_F\|$
C101	4512	4282	117	113	18	230	778	49.1	2.3
C102	6578	6254	209	115	17	324	1459	35.5	1.2
C103	8366	7977	265	124	18	389	771	31.9	2.3
C104	9523	8997	383	143	17	526	876	27.2	1.9
C105	5050	4793	147	110	15	257	1098	42.8	1.4
C106	5421	5146	159	116	20	275	1514	42.2	1.3
C107	5617	5361	140	116	23	256	2638	45.3	0.9
C108	6342	6025	194	123	16	317	719	38.8	2.2
C109	7454	7133	196	125	17	321	487	38.9	3.5
R101	3241	2950	53	238	882	291	1468	81.8	60.1
R102	5602	5292	120	190	148	310	1236	61.3	12.0
R103	7342	6961	234	147	68	381	1005	38.6	6.8
R104	8904	6907	788	1209	548762	1997	687515	60.5	79.8
R105	4256	3759	294	203	323	497	5325	40.8	6.1
R106	6400	5649	399	352	1636	751	18365	46.9	8.9
R107	7920	6633	658	629	30567	1287	71335	48.9	42.8
R108	9209	6863	775	1571	2585305	2346	2733203	67.0	94.6
R109	6027	4950	591	486	35207	1077	54565	45.1	64.5
R110	7959	6444	717	798	91597	1515	122540	52.7	74.7
R111	7953	6410	695	848	125197	1543	202426	55.0	61.8
R112	10012	7251	896	1865	3452635	2761	3477032	67.5	99.3
RC101	3639	2897	370	372	3816	742	15242	50.1	25.0
RC102	5632	3370	839	1423	418462	2262	513365	62.9	81.5
RC103	7334	5430	760	1144	175815	1904	249286	60.1	70.5
RC104	8888	6630	1018	1240	190077	2258	303740	54.9	62.6
RC105	5244	3430	615	1199	168211	1814	225529	66.1	74.6
RC106	5383	2136	862	2385	2055430	3247	2109527	73.5	97.4
RC107	7538	6378	694	466	1912	1160	12999	40.2	14.7
RC108	9311	6875	1130	1306	316907	2436	436506	53.6	72.6
C201	5221	4981	137	103	11	240	27410542	42.9	0.0
C202	7350	7059	188	103	9	279	6464179	36.9	0.0
C203	8866	8519	244	103	7	326	794849	31.6	0.0
C205	5698	5423	172	103	8	277	6549462	37.2	0.0
C206	6212	5888	221	103	11	308	2747823	33.4	0.0
C207	6578	6240	235	103	10	336	967719	30.7	0.0
C208	6665	6296	263	106	13	329	1656669	32.2	0.0

LUC-based arc fixing further reduces the number of enumerated routes. Across these instances, the average ratio $|\mathcal{R}_T|/|\mathcal{R}_F|$ is 31.3%. Route reduction is more effective for C-type instances than for R-type and RC-type instances, possibly because C-type instances tend to yield tighter dual bounds on average. For several instances, the number of enumerated routes is reduced from millions to approximately ten, highlighting the effectiveness of LUC-based arc fixing in enhancing the column enumeration method. Specifically, the LUC-based method solves 35 of the 36 instances to optimality, one more than the feasible-LP-dual method. In addition, route reduction shortens the average solution time for the final IP formulation restricted to the enumerated routes, reducing it from 519 seconds to 407 seconds.

3.6 Summary

This study introduces LUCs, a novel class of optimality cuts derived from Lagrangian duals, including infeasible LP duals. We utilize LUCs to strengthen reduced cost cuts and to derive new domain reduction techniques. Additionally, we leverage LUCs to develop new arc fixing techniques for the VRPTW, based on Lagrangian duals and the tailored dual-picking strategy. Computational experiments demonstrate that compared to the feasible-LP-dual approach studied in the literature, LUC-based arc fixing reduces the average number of remaining arcs to 47.9%, while the average number of enumerated routes decreases to 31.3%. For several instances, the reduction in enumerated routes reaches orders of magnitude, underscoring the substantial effectiveness of the proposed methods.

Chapter 4

Solving Large-Scale Survivable Traffic Grooming in Telecommunications Network Design under Double-Link Failures

4.1 Introduction

This chapter addresses the third challenge outlined in Chapter 1: the scalability and survivability in telecommunications network design. As modern telecommunications networks expand in size and complexity, ensuring survivability against link failures has become essential for maintaining reliable and uninterrupted services. However, the large-scale network sizes and stringent survivability requirements introduce a vast number of decision variables and operational constraints, significantly increasing the complexity of optimizing network design and signal traffic routing. To tackle this issue, we propose a heuristic algorithm capable of efficiently optimizing the

survivable design of large-scale telecommunications networks, accommodating over 2,000 nodes, 10,000 demands, and 9 million failure scenarios.

Traffic grooming optimizes telecommunications networks by consolidating low-rate traffic flows into high-rate *lightpaths*, reducing facility costs and increasing network throughput [Wu et al., 2020]. In this context, each demand must be assigned a route connecting its two terminal nodes, subject to various side constraints. This is crucial given the ever-increasing global telecommunications demands [Cisco, 2024]. To ensure service reliability, telecommunications companies often require protection against double-link failures, maintaining communication routes for demands even when two fiber links fail. This motivates our study on the STG2.

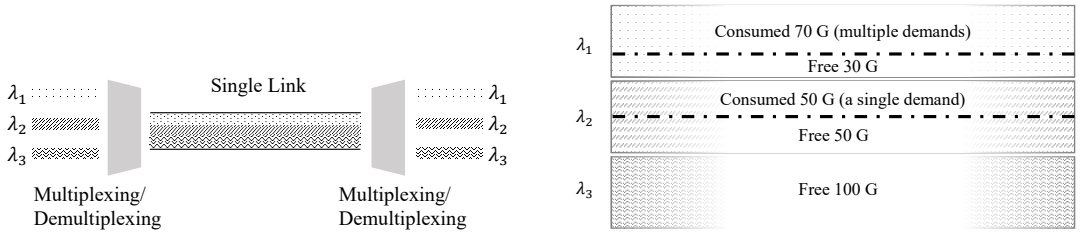
To the best of our knowledge, the existing literature on traffic grooming does not address double-link failures with partial path protection. Moreover, the telecommunications network sizes examined in existing studies on simpler problem settings are limited to no more than 300 nodes, which are significantly smaller than emerging real-world telecommunications networks that encompass thousands of nodes. In this study, we aim to tackle the STG2 in large-scale networks, considering various practical side constraints. These constraints facilitate fast switching between working routes under the scenario without link failures and backup routes under the scenarios with link failures. Solving the STG2 problem is challenging due to the complexity introduced by the large-scale network, numerous failure scenarios, and practical constraints involving interdependency among scenarios. Additionally, practitioners often impose time and memory limits on the solution method to ensure efficiency. To tackle these challenges, we develop a novel hierarchical constructive heuristic for the STG2 in this study.

4.1.1 Background of Survivable Traffic Grooming

The traffic grooming problem is based on an *optical transport network* (OTN), a telecommunications network that transmits high-speed, high-capacity, and reliable traffic flows using optical fibers. It serves as a critical backbone for internet connections across extensive geographical areas. An OTN consists of interconnected nodes, such as data centers and switching centers, and links made of optical fibers that enable high-speed transmission over long distances. The network manages a series of demands, each defined by a specific traffic volume and two terminal nodes. These demands are strategically routed to minimize facility costs and ensure service reliability.

Traffic grooming enables higher utilization of network resources, compared to the widely studied *routing and wavelength assignment* (RWA) problem. Both problems are based on OTNs utilizing *wavelength division multiplexing* (WDM), which divides the optical signal on a single link into multiple wavelengths (see Figure 4.1(a)), each capable of transmitting traffic flows independently. In the RWA, a wavelength on a link is assigned to at most one demand. In the traffic grooming problem, the network throughput can be further increased, as multiple demands can share the same wavelength on a link, provided that their combined traffic does not exceed the bandwidth capacity (see Figure 4.1(b)). As a result, it can be seen that the RWA is a special case of the traffic grooming problem.

Compared to RWA, the traffic grooming problem requires a significantly larger number of decisions, as it must determine how different communication demands share the lightpaths. A lightpath, defined by a wavelength and a sequence of end-to-end connected links, serves as a communication channel between two endpoints equipped with signal processing devices. Multiple low-speed demands can be multiplexed onto this lightpath and transmitted between the endpoints



(a) Utilizing WDM, the optical signal on a link is divided into 3 wavelengths, capable of transmitting traffic flows independently.

(b) Each wavelength has a bandwidth capacity of 100 Gbps. Multiple demands can share a wavelength, provided the total traffic does not exceed this capacity.

Figure 4.1: Wavelength and bandwidth capacity of a single link in a WDM network supporting traffic grooming.

using the designated wavelength and links. It aims to minimize the total number of established lightpaths, which is proportional to the number of required devices, constituting a major portion of network costs [Wu et al., 2015].

The STG2 enhances network reliability by ensuring survivability against link failures, such as accidental fiber cuts that disrupt routes. A double-link failure occurs when one link fails and another fails before the first is repaired. As OTN sizes continue to expand dramatically, the probability of double-link failures has increased [Sasithong et al., 2020]. To improve reliability, STG2 considers the working scenario without link failures, as well as all other scenarios involving a single link failure or two link failures, as required by many telecommunications companies today. The network reverts to its working scenario once the failed links are repaired, thereby ensuring protection against future failures. In practice, triple-link failure scenarios are excluded from consideration due to both their low probability of occurrence and the prohibitively high costs associated with over-conservative protection schemes.

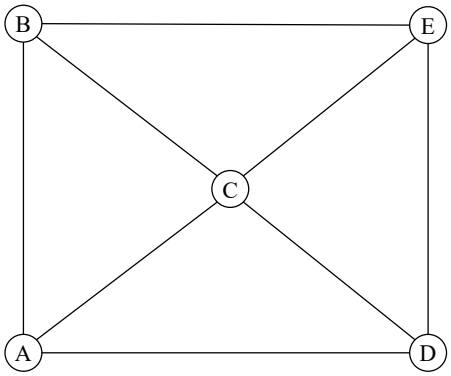
In the STG2, the number of failure scenarios considered grows quadratically with the number of links. For instance, in an OTN with 8 links shown in Figure 4.2, there are 1 working scenario and 64 failure scenarios (comprising 8 single-link

failure scenarios and 56 double-link failure scenarios). Figure 4.2 illustrates routing plans for 5 of these scenarios, while routing plans for the remaining 60 scenarios are provided in Appendix B.3. Notably, lightpaths l_3, l_4, l_5 in Figure 4.2 are pre-established and reserved to accommodate failure scenarios.

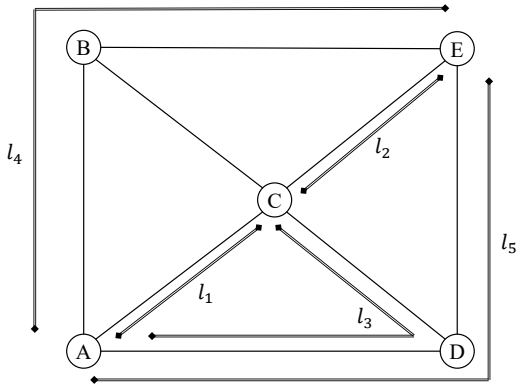
The STG2 aims to minimize the total number of lightpaths to be established while planning a route (as a concatenation of lightpaths) for each demand in every scenario. It ensures that backup routes in failure scenarios do not traverse failed links. This problem can be characterized as a two-stage optimization framework, where the second stage focuses exclusively on feasibility problems without incorporating objective functions. Specifically, the first stage determines which lightpaths should be established, while the second stage involves routing over these pre-established lightpaths to ensure that all demands are satisfied across all considered scenarios. In contrast to stochastic programming, the absence of objective functions in the second stage eliminates the need to consider the probability distribution of scenario occurrences.

4.1.2 Challenges for Solving Large-Scale STG2 and Needs for Developing Constructive Heuristic

The classic traffic grooming problem is NP-hard [Zhu and Mukherjee, 2002, Wang and Gu, 2008], and the network size significantly impacts the efficiency of solution methods. In related studies, only instances of no more than 300 nodes are solved (see Table 4.1). We aim to tackle large-scale STG2 with thousands of nodes, aligning with evolving practical requirements driven by the rapid expansion of OTNs. The increase in network size leads to a much larger decision space, which, together with complicated problem settings, poses greater challenges in optimizing routes for communication demands.



(a) An OTN with 5 nodes and 8 links. There are two available wavelengths.



Lightpath	Wavelength	Links
l_1	λ_1	AC
l_2	λ_1	CE
l_3	λ_1	AD-DC
l_4	λ_1	AB-BE
l_5	λ_2	AD-DE

(b) The set of established lightpaths in an optimal solution.

Route	Lightpaths	Route	Lightpaths
r_1	l_1	r_5	l_4
r_2	l_1-l_2	r_6	l_4-l_2
r_3	l_3	r_7	l_5
r_4	l_3-l_2		

Scenario	Route for (A,C,50 G)	Route for (A,E,50 G)
(0, 0)	r_1	r_2
(AC, 0)	r_3	r_4
(CE, 0)	r_1	r_5
(AC, CE)	r_3	r_5
(AB, CE)	r_1	r_7

(c) Routing for 5 scenarios.

Figure 4.2: Routing for 5 out of the total 65 scenarios. Routing plans for the remaining 60 scenarios are provided in Appendix B.3. Demands are (A, C, 50 Gbps) and (A, E, 50 Gbps). The bandwidth of a wavelength is 100 Gbps. $(0, 0)$ is the working scenario; $(e_1, 0)$ represents link e_1 fails; (e_1, e_2) represents links e_1 and e_2 fail successively.

Table 4.1: Largest scales of RWA and traffic grooming instances solved in the literature and this study

Reference	Problem	Failure Type	# Node	# Link
Jaumard and Daryalal [2017]	RWA	failure-free	90	350
Sebbah and Jaumard [2012]	RWA	single-link	43	71
Feng et al. [2010]	RWA	double-link	47	98
Zhu et al. [2003]	traffic grooming	failure-free	277	unknown
Niu et al. [2024]	traffic grooming	single-link	227	409
Assi et al. [2006]	traffic grooming	double-link	24	43
This study	traffic grooming	double-link	2600	3027

Notes. Assi et al. [2006] propose a restoration scheme for double-link failure scenarios, which does not guarantee protection for all such scenarios. In contrast, our approach ensures protection for all double-link failure scenarios.

Another challenge originates from the large number of failure scenarios considered. In a survivable OTN with m links, the number of failure scenarios under single-link failures is m. When double-link failures are additionally considered, the total number of failure scenarios increases by $m(m-1)$, resulting in a total of m^2 failure scenarios. This, together with Table 4.1, indicates that the number of failure scenarios considered in the literature is typically no more than 10,000. However, for the large-scale STG2 instances examined in this study, involving thousands of links, there are millions of failure scenarios to consider. This results in more decisions to make, constraints to verify, and data to store, thereby significantly complicating the problem.

Our study also incorporates several practical constraints that challenge algorithm design. For instance, consistent routing and bandwidth reservation constraints are essential for fast switching between working and backup routes, ensuring low-latency connections. These constraints create interdependency in routing and bandwidth consumption across scenarios, significantly complicating the algorithm design.

Limited computing resources, such as computing time and available memory, further increase the challenge of algorithm design for the STG2. Achieving short

computing time is essential for telecommunications service providers to optimize subsequent decisions based on the STG2 solutions. Additionally, major providers often run multiple computing tasks simultaneously on their platforms, restricting the memory and time available for solving STG2. In this study, a one-hour time limit and a 16 GB memory limit are required by our industry partner, Huawei Technologies. Preliminary experiments indicate that for their large-scale STG2 instances with thousands of nodes, the one-hour time limit is nearly reached just by verifying the solution feasibility. Additionally, importing linear programming relaxation models of these instances far exceeds the 16 GB memory limit. Consequently, methods based on iterative search or mathematical programming are not applicable. Therefore, we focus on developing an efficient constructive heuristic.

4.1.3 Related Works

4.1.3.1 Traffic Grooming without Considering Failures

Constructive heuristics have been developed for the traffic grooming problem without considering failures. Zhu et al. [2003] and Yao and Ramamurthy [2005b] construct solutions by sequentially planning routes for each demand. Zhu et al. [2003] introduce an auxiliary graph with a logical layer for lightpaths and multiple wavelength layers for wavelength-link pairs. Yao and Ramamurthy [2005b] propose a link-bundled auxiliary graph (LBAG), which consolidates all wavelength layers into a single physical layer. Both studies use shortest-path algorithms to construct routes. However, these are not applicable to our STG2 due to various side constraints that must be incorporated.

To solve the traffic grooming problem without considering failures, Chen et al. [2010] employ a three-phase method to construct solutions and groom traffic within a hub-and-spoke framework. They begin by decomposing the network into clusters, then determine endpoints of lightpaths for grooming intracluster and intercluster

traffic, and finally solve an RWA problem to decide the physical routing and wavelength of lightpaths. When adapted to our problem, the third phase may become infeasible due to various side constraints and interdependency between scenarios. Wu et al. [2015] construct a solution by sequentially finding routes for each demand, followed by optimizing the logical and physical routing separately to enhance the solution. However, they do not address optical reach constraints or limit the number of wavelengths, which are crucial in practice.

Heuristics based on mathematical programming and iterative search have also been applied to traffic grooming problems. Dawande et al. [2007] use heuristics based on column generation and Lagrangian relaxation, requiring that demands with different terminal nodes do not share the same lightpath. Vignac et al. [2016] apply Benders decomposition and Dantzig–Wolfe reformulation under strict assumptions regarding demand granularity, routing schemes, and grooming configurations. Wu et al. [2020] first construct an initial feasible solution and then iteratively search neighboring solutions to reduce the number of lightpaths. However, applying these methods to large-scale STG2 is too time-consuming and memory-intensive due to the extensive network sizes and numerous failure scenarios.

4.1.3.2 Survivable Traffic Grooming

Several different survivability schemes have been proposed for traffic grooming problems considering failures, broadly classified into two categories: Pre-planned protection and dynamic restoration [Somani, 2008]. Pre-planned protection schemes reserve resources that remain idle during the working scenario, ensuring recovery when failures occur. In contrast, dynamic restoration schemes do not reserve resources beforehand, instead searching for spare resources after a failure. Assi et al. [2006] explore a hybrid scheme for STG2, incorporating protection for single-link failures and restoration for double-link failures, although without providing

guaranteed protection for the latter. This study focuses on a protection scheme to enhance network reliability.

According to the literature, various protection schemes can be applied at the lightpath level (PAL) and the connection level (PAC). In PAL (PAC, respectively), all traffic on a lightpath (route, respectively) switches to a backup lightpath (route, respectively) upon failure. Since a route is a concatenation of lightpaths, PAL is a specialization of PAC and cannot achieve higher resource utilization than PAC. To minimize facility costs, we focus on PAC in this study. Yao and Ramamurthy [2005c] and Jaekel et al. [2008] explore survivable traffic grooming under single-link failures (STG1) using PAL, where each lightpath is accompanied by a link-disjoint backup lightpath. They identify these lightpaths using a shortest-path algorithm and an integer linear programming (ILP) formulation, respectively.

PAC can be further classified into full path protection (FPP) and partial path protection (PPP). FPP requires that working and backup routes are link-disjoint, while PPP does not. Yao and Ramamurthy [2005a] and Niu et al. [2024] investigate STG1 using FPP. Yao and Ramamurthy [2005a] focus on maximizing network throughput and propose two grooming heuristics to establish lightpaths and route traffic, either separately or jointly. Niu et al. [2024] aim to minimize the number of lightpaths while ensuring all demands are protected, proposing a heuristic that plans backup routes after establishing working routes for all demands. FPP has also been applied to STG2 [Yang, 2009] and RWA under double-link failures [Feng et al., 2010]. Jaekel et al. [2012] explore STG1 with PPP, presenting ILP formulations and a heuristic method that sequentially plans each failure scenario.

To the best of our knowledge, despite its importance, the STG2 with PPP has not been explored in the literature. As a less restrictive approach, PPP serves as a generalization of FPP, offering the potential for higher resource utilization

and cost savings. Therefore, we focus on PPP in this study, albeit with higher computational complexity.

4.1.4 Contributions

Existing studies reviewed in Section 4.1.3 typically adopt mathematical programming and iterative search methods. However, these methods cannot be directly applied to our study due to the large network sizes, extensive constraints, and imposed computing time and memory limits (1 hour and 16 GB). Consequently, as explained in Section 4.1.2, to effectively address large-scale STG2, we develope an efficient constructive heuristic (outlined in Figure 4.3) with the following key contributions:

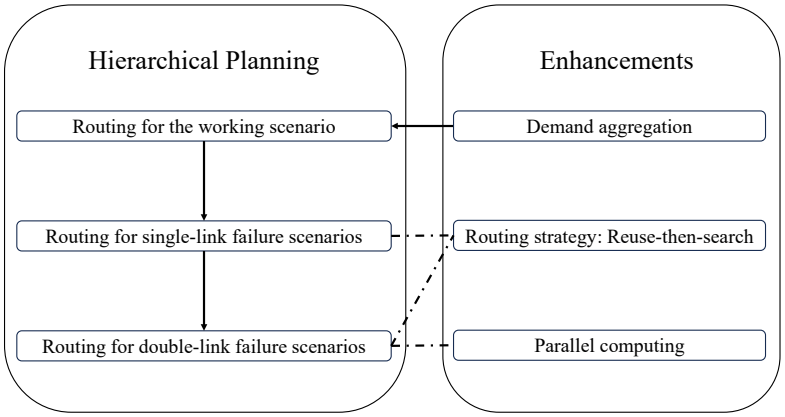


Figure 4.3: Outline of the hierarchical constructive heuristic and algorithm enhancements. The process begins with demand aggregation, followed by planning the working scenario, single-link failure scenarios, and double-link failure scenarios in sequence. The reuse-then-search routing strategy is utilized for single-link and double-link failure scenarios, while parallel computing is employed for planning double-link failure scenarios.

- We devise a hierarchical sequence of scenarios for constructing routes and assigning them to communication demands. This approach mitigates interdependency between scenarios, enabling our heuristic to efficiently manage

consistent routing constraints and validate bandwidth capacity constraints.

- We propose a reuse-then-search routing strategy to efficiently satisfy demands in a large-scale network. It first attempts to reuse existing routes to satisfy new demands without increasing the objective value. If it fails, a tailored labeling algorithm is employed to search for a new route.

- To manage the large number of demands and expedite the optimization, we aggregate multiple demands into a single one, assigning them the same routing plan. To minimize the number of aggregated demands while respecting capacity constraints, we introduce a bin-packing problem for each pair of terminal nodes and solve it with an efficient heuristic.

- To efficiently manage the large number of double-link failure scenarios, our newly proposed constructive heuristic can be parallelized by utilizing multiple threads to concurrently plan distinct scenarios. Several mechanisms are implemented to prevent the generation of substitutable lightpaths, mitigate race conditions (where multiple threads access and modify shared data simultaneously), and minimize thread idleness. To the best of our knowledge, this is the first parallel algorithm developed for survivable OTN problems.

- We conduct extensive computational experiments on large-scale STG2 instances provided by our industrial partner. The results demonstrate the effectiveness of our constructive heuristic and various enhancements. Within the one-hour time limit and 16 GB memory limit, our heuristic can solve instances with over 2,000 nodes, 3,000 links, 10,000 demands, and 9 million scenarios, improving the objective value of the best-known solutions by 18.6% on average, highlighting its significant potential for practical applications. Additionally, we generate a set of small-scale instances, where a general optimization solver, CPLEX, is able to solve only those with no more than 6

nodes and 10 links. For these small-scale instances, our heuristic can solve each one within 0.1 seconds, producing solutions with quality comparable to those obtained by CPLEX, with an average relative gap of only 5.6%. These results highlight the promising performance of our method in terms of both efficiency and solution quality.

In the remainder of this chapter, we present the problem statement in Section 4.2, the hierarchical constructive heuristic in Section 4.3, and the enhancements by parallel computing in Section 4.4. Our computational results are reported in Section 4.5. The chapter is summarized in Section 4.6.

4.2 Problem Description of STG2

The STG2 is defined on an OTN represented by an undirected graph $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of undirected links. Each link $e \in \mathcal{E}$ has an associated length $d(e)$ and a set of wavelengths Λ. Each wavelength $\lambda \in \Lambda$ has a bandwidth capacity B, a standard assumption widely adopted in the literature [Cartier et al., 2006, Bahri and Chamberland, 2008][1]. There is a set of demands $\mathcal{K}$, where each demand $k \in \mathcal{K}$ has a specific traffic volume $b_k \leq B$, requiring transmission between two terminal nodes $s_k \in \mathcal{N}$ and $t_k \in \mathcal{N}$. Figure 4.2 illustrates an example featuring 5 nodes, 8 links, 2 wavelengths, and 2 demands.

The STG2 involves route planning for demands across all possible scenarios involving zero, one, or two failed links. The working scenario (no link failures) is denoted as $(0, 0)$, the *level-1 scenario* where a single link e_1 fails is denoted as $(e_1, 0)$, and the *level-2 scenario* where two links e_1 and e_2 fail successively is denoted as (e_1, e_2). The set of all considered scenarios is represented by $\Omega = \{(0, 0)\} \cup \Omega^1 \cup \Omega^2$, where $\Omega^1 = \{(e_1, 0) : e_1 \in \mathcal{E}\}$ and $\Omega^2 = \{(e_1, e_2) : e_1, e_2 \in \mathcal{E}, e_1 \neq e_2\}$.

[1]Our algorithm can be adapted to accommodate non-uniform wavelength bandwidths, by initializing the capacity of each lightpath based on its corresponding wavelength.

The STG2 aims to minimize the number of established lightpaths, ensuring that each demand is assigned a route in every considered scenario, without traversing failed links. A lightpath with n links $l = (\lambda, (e_l^1, e_l^2, \ldots, e_l^n))$, defined by a wavelength $\lambda \in \Lambda$ and a sequence of distinct, end-to-end connected links $e_l^1, e_l^2, \ldots, e_l^n$, serves as a communication channel between two endpoints. This definition implies that a lightpath l has a bandwidth capacity of B and must occupy the same wavelength λ on traversed links $e_l^1, e_l^2, \ldots, e_l^n$, known as the *wavelength continuity constraints* [Martins et al., 2012, Lü et al., 2023]. For example, the lightpath $l_3 = (\lambda_1, (\text{AD}, \text{CD}))$ shown in Figure 4.2 occupies the same wavelength λ_1 on links AD and CD, and can transmit up to 100 Gbps of traffic between endpoints A and C. Additionally, due to signal attenuation, the *optical reach constraints* [Yıldız et al., 2018, Li and Aneja, 2020] require that the length of a lightpath l does not exceed the optical reach limit $\bar{d}$, i.e., $\sum_{e \in \mathcal{E}(l)} d(e) \leq \bar{d}$ where $\mathcal{E}(l) = \{e_l^1, e_l^2, \ldots, e_l^n\}$ is the set of links traversed by l. A route with n lightpaths $r = (l_r^1, l_r^2, \ldots, l_r^n)$ is a sequence of distinct, end-to-end connected lightpaths. For example, $r_2 = (l_1, l_2)$, depicted in Figure 4.2, is a route connecting nodes A and E. Denote $\mathcal{L}$ and $\mathcal{R}$ as the sets of all lightpaths and routes, respectively. The cardinalities $|\mathcal{L}|$ and $|\mathcal{R}|$ can grow exponentially with the network size. Denote $r_k^{e_1 e_2}$ as the route for demand $k \in \mathcal{K}$ used in scenario $(e_1, e_2) \in \Omega$.

In a failure scenario, each demand should be transmitted along a route without traversing any failed link. Specifically, for the routing plan of a demand $k \in \mathcal{K}$ depicted in Figure 4.4, we must have $e_1 \notin \mathcal{E}(r_k^{e_1 0})$ for scenario $(e_1, 0) \in \Phi_k = \{(e', 0) : e' \in \mathcal{E}(r_k^{00})\}$, $e_1, e_2 \notin \mathcal{E}(r_k^{e_1 e_2})$ for scenario $(e_1, e_2) \in \Theta_k(e_1) = \{(e_1, e'') : e'' \in \mathcal{E}(r_k^{e_1 0})\}$, and $e_1', e_2'' \notin \mathcal{E}(r_k^{e_1' e_2''})$ for scenario $(e_1', e_2'') \in \Psi_k = \{(e', e'') : e' \in \mathcal{E} \setminus \mathcal{E}(r_k^{00}), e'' \in \mathcal{E}(r_k^{00})\}$.

The STG2 also needs to consider several other side constraints. *Simple path constraints* require a route to be elementary, meaning it traverses each node at

most once. For example, in Figure 4.2, route $r = (l_3, l_5)$ connecting nodes C and E is not elementary, as it traverses node D twice. Although these constraints can reduce system disorders [Wu et al., 2015], they may increase the objective value. Therefore, we require only that working routes $(r_k^{00}, k \in \mathcal{K})$ be elementary.

Wavelength assignment constraints are widely considered for OTNs [Daryalal and Bodur, 2022], requiring that a wavelength on a link be occupied by at most one established lightpath. In other words, any two lightpaths l and l' using the same wavelength ($\lambda(l) = \lambda(l')$, where $\lambda(l)$ represents the wavelength occupied by lightpath l) cannot traverse a common link ($\mathcal{E}(l) \cap \mathcal{E}(l') = \emptyset$). For example, in Figure 4.2, lightpaths l_3 and l_5 must occupy different wavelengths (λ_1 and λ_2), because they share the link AD. Denote $\mathcal{L}(r)$ as the set of lightpaths included in route r. Note that different lightpaths in $\mathcal{L}(r)$ can occupy different wavelengths.

Consistent routing constraints are essential for fast route switching and low-latency communications, as they minimize the number of routes that need to be altered when a link fails. These constraints ensure that the route used before a failure continues to be used if it does not traverse the failed link. Specifically, for each demand $k \in \mathcal{K}$, the working route r_k^{00} should remain in use in level-1 scenarios $\bar{\Phi}_k = \{(e_1, 0) \in \Omega^1 : e_1 \in \mathcal{E} \setminus \mathcal{E}(r_k^{00})\}$ and level-2 scenarios $\bar{\Psi}_k = \{(e_1, e_2) \in \Omega^2 : e_1, e_2 \in \mathcal{E} \setminus \mathcal{E}(r_k^{00})\}$. Similarly, for each link $e_1 \in \mathcal{E}(r_k^{00})$, the level-1 route $r_k^{e_1 0}$ should remain in use in level-2 scenarios $\bar{\Theta}_k(e_1) = \{(e_1, e_2) \in \Omega^2 : e_2 \in \mathcal{E} \setminus \mathcal{E}(r_k^{e_1 0})\}$. This is illustrated in Figure 4.4. Due to consistent routing constraints, the pair of level-2 scenarios $(e_1, e_2) \in \Omega^2$ and (e_2, e_1), despite comprising the same set of failure links, differ in the sequence of failures and must therefore be treated as distinct. Consequently, the routing plan devised for one scenario cannot be directly applied to the other. This distinction arises because the routing plans for level-2 scenario (e_1, e_2) and (e_2, e_1) are contingent upon the corresponding level-1 scenarios $(e_1, 0)$

and $(e_2, 0)$, respectively, which may themselves entail different routing plans.

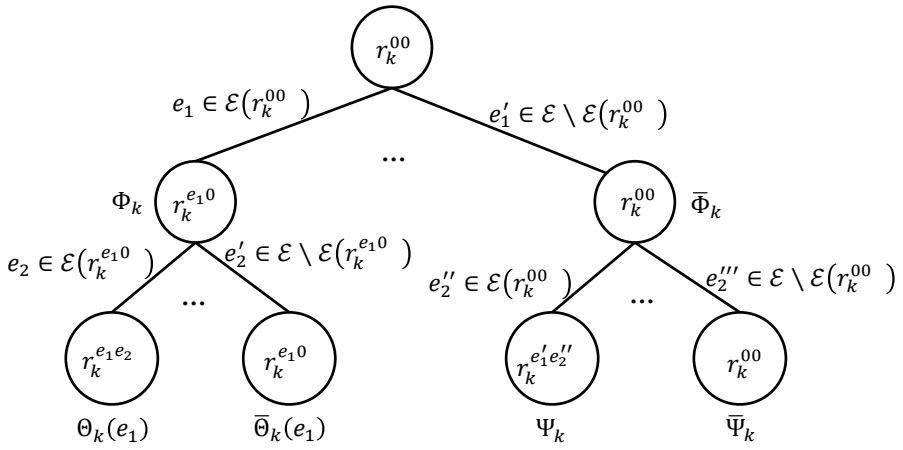


Figure 4.4: Route planning for demand $k \in \mathcal{K}$ in scenarios Ω, where failed links are represented on the edges. r_k^{00} is the working route, which is also used in scenarios $\bar{\Phi}_k$ and $\bar{\Psi}_k$ due to consistent routing constraints. $r_k^{e_1 0}$ is the backup route used in level-1 scenario $(e_1, 0)$, which is also used in scenarios $\bar{\Theta}_k(e_1)$ due to consistent routing constraints.

Bandwidth reservation constraints are also essential for fast route switching and low-latency communications. When failed links are repaired, all demands revert to their working routes, ensuring protection against future failures. These constraints require that the bandwidth on lightpaths consumed by demand $k \in \mathcal{K}$ in the working scenario be reserved. The reserved bandwidth can be reused by demand k in failure scenarios, but cannot be used by any other demand in any scenario. This ensures that demand k can revert to the working route without waiting for other demands to release bandwidth resources, thereby accelerating route switching. Consequently, the *bandwidth capacity constraints* require that the total bandwidth consumption, including reserved bandwidth, on each lightpath in any scenario must not exceed B.

For each demand $k \in \mathcal{K}$, we define $\mathcal{R}_k \subseteq \mathcal{R}$ as the subset of routes with endpoints the same with terminal nodes s_k and t_k, and $\bar{\mathcal{R}}_k \subseteq \mathcal{R}_k$ as the set

of elementary routes (traversing each node at most once) in $\mathcal{R}_k$. Let $x_l, l \in \mathcal{L}$ be a binary variable equal to one if and only if lightpath l is established. Let $y_{kl}^{e_1e_2}, k \in \mathcal{K}, l \in \mathcal{L}, (e_1, e_2) \in \Omega : e_1, e_2 \notin \mathcal{E}(l)$ be a binary variable equal to one if and only if lightpath l belongs to the route used by demand k in scenario (e_1, e_2). Let $z_{kr}^{00}, k \in \mathcal{K}, r \in \bar{\mathcal{R}}_k$ be a binary variable equal to one if and only if route r is the working route of demand k. Let $z_{kr}^{e_1e_2}, k \in \mathcal{K}, r \in \mathcal{R}_k, (e_1, e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(r)$ be a binary variable equal to one if and only if route r is the backup route of demand k in scenario (e_1, e_2). Accordingly, the STG2 problem can be formulated as follows.

$$\min \sum_{l \in \mathcal{L}} x_l, \tag{4.1a}$$

s.t.
$$\sum_{l \in \mathcal{L}: \lambda(l) = \lambda, e \in \mathcal{E}(l)} x_l \leq 1, \quad \forall \lambda \in \Lambda, e \in \mathcal{E}, \tag{4.1b}$$

$$\sum_{r \in \bar{\mathcal{R}}_k} z_{kr}^{00} = 1, \quad \forall k \in \mathcal{K}, \tag{4.1c}$$

$$\sum_{r \in \mathcal{R}_k : e_1, e_2 \notin \mathcal{E}(r)} z_{kr}^{e_1e_2} = 1, \quad \forall k \in \mathcal{K}, (e_1, e_2) \in \Omega^1 \cup \Omega^2, \tag{4.1d}$$

$$z_{kr}^{00} - z_{kr}^{e_1 0} \leq 0, \quad \forall k \in \mathcal{K}, r \in \bar{\mathcal{R}}_k, (e_1, 0) \in \Omega^1 : e_1 \notin \mathcal{E}(r), \tag{4.1e}$$

$$z_{kr}^{e_1 0} - z_{kr}^{e_1e_2} \leq 0, \quad \forall k \in \mathcal{K}, r \in \mathcal{R}_k, (e_1, e_2) \in \Omega^2 : e_1, e_2 \notin \mathcal{E}(r), \tag{4.1f}$$

$$y_{kl}^{00} - \sum_{r \in \bar{\mathcal{R}}_k : l \in \mathcal{L}(r)} z_{kr}^{00} = 0, \quad \forall k \in \mathcal{K}, l \in \mathcal{L}, \tag{4.1g}$$

$$y_{kl}^{e_1e_2} - \sum_{r \in \mathcal{R}_k : l \in \mathcal{L}(r), e_1, e_2 \notin \mathcal{E}(r)} z_{kr}^{e_1e_2} = 0, \quad \forall k \in \mathcal{K}, l \in \mathcal{L}, (e_1, e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(l), \tag{4.1h}$$

$$\sum_{k \in \mathcal{K}} b_k y_{kl}^{00} - B x_l \leq 0, \quad \forall l \in \mathcal{L}, \tag{4.1i}$$

$$\sum_{k \in \mathcal{K}} b_k \max\{y_{kl}^{00}, y_{kl}^{e_1e_2}\} - B x_l \leq 0, \quad \forall l \in \mathcal{L}, (e_1, e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(l), \tag{4.1j}$$

$$x_l \in \{0, 1\}, \quad \forall l \in \mathcal{L}, \tag{4.1k}$$

$$y_{kl}^{e_1e_2} \in \{0, 1\}, \quad \forall k \in \mathcal{K}, l \in \mathcal{L}, (e_1, e_2) \in \Omega : e_1, e_2 \notin \mathcal{E}(l), \tag{4.1l}$$

$$z_{kr}^{00} \in \{0,1\}, \quad \forall k \in \mathcal{K}, r \in \bar{\mathcal{R}}_k, \tag{4.1m}$$

$$z_{kr}^{e_1e_2} \in \{0,1\}, \quad \forall k \in \mathcal{K}, r \in \mathcal{R}_k, (e_1,e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(r). \tag{4.1n}$$

The objective function (4.1a) is to minimize the number of established lightpaths. Wavelength assignment constraints (4.1b) require that a wavelength on a link can be assigned to at most one established lightpath. Constraints (4.1c) ensure that each demand is assigned an elementary working route, while constraints (4.1d) require that each demand is assigned a backup route in each failure scenario, avoiding failed links. Consistent routing constraints (4.1e) impose that the working route should still be used as the backup route in a level-1 scenario $(e_1, 0) \in \Omega^1$ if it does not traverse the failed link e_1. Consistent routing constraints (4.1f) require that the backup route in a level-1 scenario $(e_1, 0) \in \Omega^1$ should still be used as the backup route in a level-2 scenario $(e_1, e_2) \in \Omega^2$ if it does not traverse the second failed link e_2. Constraints (4.1g) and (4.1h) link variables $\boldsymbol{y}$ and $\boldsymbol{z}$. Bandwidth capacity constraints are described with (4.1i) and (4.1j). Constraints (4.1k)-(4.1n) specify domains of variables. Note that both the wavelength continuity constraints and the optical reach constraints are respected according to the definition of $\mathcal{L}$.

The formulation (4.1a)–(4.1n) above for the STG2 can be reformulated as an equivalent ILP formulation by replacing constraints (4.1j) with constraints (4.2a)–(4.2d) below:

$$w_{kl}^{e_1e_2} \geq y_{kl}^{00}, \quad \forall l \in \mathcal{L}, (e_1,e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(l), \tag{4.2a}$$

$$w_{kl}^{e_1e_2} \geq y_{kl}^{e_1e_2}, \quad \forall l \in \mathcal{L}, (e_1,e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(l), \tag{4.2b}$$

$$\sum_{k \in \mathcal{K}} b_k w_{kl}^{e_1e_2} - Bx_l \leq 0, \quad \forall l \in \mathcal{L}, (e_1,e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(l), \tag{4.2c}$$

$$w_{kl}^{e_1e_2} \in \{0,1\}, \quad \forall l \in \mathcal{L}, (e_1,e_2) \in \Omega^1 \cup \Omega^2 : e_1, e_2 \notin \mathcal{E}(l), \tag{4.2d}$$

where $w_{kl}^{e_1e_2}$ is a binary variable equal to one if the bandwidth of b_k on lightpath l is consumed ($y_{kl}^{e_1e_2} = 1$) or reserved ($y_{kl}^{00} = 1$ and $y_{kl}^{e_1e_2} = 0$) for demand k in failure scenario $(e_1, e_2) \in \Omega^1 \cup \Omega^2$.

It is worth noting that for large-scale STG2 instances, which involve thousands of nodes as examined in this study, the above ILP formulation cannot be directly solved using general optimization solvers due to memory constraints encountered when importing their linear programming relaxation models. In this study, the ILP is utilized only for solving small-scale instances, which facilitates the performance assessment of our proposed heuristic. Moreover, designing decomposition methods to solve the ILP formulation is non-trivial due to the complexity of the constraints, rendering it computationally unaffordable, particularly given the extensive network sizes and the large number of failure scenarios. Nevertheless, by selectively relaxing certain constraints, decomposition methods can be utilized to derive lower bounds on the optimal objective values, which are valuable for evaluating solutions obtained from primal methods, and merit further investigation in future research. Furthermore, our preliminary numerical experiments indicate that verifying solution feasibility for large-scale STG2 instances nearly exhausts the one-hour time limit, making iterative search-based methods too time-consuming. To address these limitations, we propose a hierarchical constructive heuristic specifically designed to efficiently handle large-scale STG2 instances, as detailed in Section 4.3 below.

4.3 Hierarchical Constructive Heuristic

To efficiently solve large-scale STG2 instances (thousands of nodes and millions of failure scenarios) with limited computing time and memory, we propose a novel hierarchical constructive heuristic, outlined in Section 4.3.1. Section 4.3.2 illustrates the method for aggregating demands. In Section 4.3.3, we develop a labeling algorithm for finding routes, and Section 4.3.4 introduces a technique for

reusing generated routes. Our hierarchical constructive heuristic is further enhanced by efficient checks of bandwidth capacity constraints (see Appendix B.1.1) and efficient updates of bandwidth consumption (see Appendix B.1.2).

4.3.1 Algorithm Outline

The hierarchical constructive heuristic is outlined in Algorithm 5. It begins by aggregating demands (line 1) to reduce the problem size (Section 4.3.2), followed by sorting the sequence of demands (line 2). The solution is constructed hierarchically: First plan the working scenario (lines 3–5), then plan level-1 scenarios (lines 7–10), and finally plan level-2 scenarios (lines 12–14). In each scenario, routes for demands are planned sequentially. Working routes are determined using the labeling algorithm (Section 4.3.3). When planning level-1 and level-2 routes, to reduce the number of calls to the labeling algorithm, we first try to reuse routes generated during the solution process (see Section 4.3.4). If this fails, we then employ the labeling algorithm to generate a new route.

The hierarchical constructive heuristic offers several advantages. Firstly, it naturally satisfies consistent routing constraints by sequentially planning the working route, level-1 routes, and level-2 routes. Specifically, once demand $k \in \mathcal{K}'$ is assigned a working route r_k^{00}, this route is also assigned to the demand in level-1 scenarios $\bar{\Phi}_k$ and level-2 scenarios $\bar{\Psi}_k$. Similarly, when demand $k \in \mathcal{K}'$ is assigned a level-1 route $r_k^{e_1 0}$ in scenario $(e_1, 0) \in \Phi_k$, this route is also assigned in level-2 scenarios $\bar{\Theta}_k(e_1)$ by reserving bandwidth b_k on lightpaths $\mathcal{L}(r_k^{e_1 0})$ in scenarios $\bar{\Theta}_k(e_1)$. Thus, we only need to explicitly plan the working scenario, level-1 scenarios $\bigcup_{k\in\mathcal{K}'} \Phi_k$, and level-2 scenarios $\bigcup_{k\in\mathcal{K}'} \Psi_k \cup \Theta_k$, where $\Theta_k = \bigcup_{e_1\in\mathcal{E}(r_k^{00})} \Theta_k(e_1)$.

Secondly, while planning a route in one scenario can affect bandwidth consumption in others (due to bandwidth reservation and consistent routing constraints), the hierarchical constructive heuristic allows us to verify bandwidth capacity con-

Algorithm 5 Hierarchical Constructive Heuristic

Input: $\mathcal{N}, \mathcal{E}, \Lambda, \mathcal{K}, B, \bar{d}$
Output: Routes $r_k^{e_1 e_2}, k \in \mathcal{K}, (e_1, e_2) \in \Omega$

Aggregate demands $\mathcal{K}$ into $\mathcal{K}'$ (Section 4.3.2)
Sort demands $\mathcal{K}'$ in non-increasing order of traffic volume
for all demand $k \in \mathcal{K}'$ **do**
    Find the working route r_k^{00} with the labeling algorithm (Section 4.3.3)
    $\Phi_k \leftarrow \{(e_1, 0) : e_1 \in \mathcal{E}(r_k^{00})\}$; $\Psi_k \leftarrow \{(e_1, e_2) : e_1 \in \mathcal{E} \setminus \mathcal{E}(r_k^{00}), e_2 \in \mathcal{E}(r_k^{00})\}$
Sort scenarios Ω^1 in non-increasing order of the number of unplanned demands
for all scenario $(e_1, 0) \in \Omega^1$ **do**
    for all demand $k \in \mathcal{K}'$ where $(e_1, 0) \in \Phi_k$ **do**
        Find the level-1 route $r_k^{e_1 0}$ to protect k in scenario $(e_1, 0)$ by reusing an existing route (Section 4.3.4). If this fails, use the labeling algorithm (Section 4.3.3) to find a new route
        $\Theta_k(e_1) \leftarrow \{(e_1, e_2) : e_2 \in \mathcal{E}(r_k^{e_1 0})\}$
Sort scenarios Ω^2 in non-increasing order of the number of unplanned demands
for all scenario $(e_1, e_2) \in \Omega^2$ **do**
    for all demand $k \in \mathcal{K}'$ where $(e_1, e_2) \in \Psi_k \cup \Theta_k(e_1)$ **do**
        Find the level-2 route $r_k^{e_1 e_2}$ to protect k in scenario (e_1, e_2) by reusing an existing route (Section 4.3.4). If this fails, use the labeling algorithm (Section 4.3.3) to find a new route

straints across all scenarios by checking only the scenario being currently planned (see Appendix B.1.1). It also enables updating bandwidth consumption with fewer operations and less memory than a direct update approach (see Appendix B.1.2).

Moreover, once the working and all level-1 scenarios are planned, planning a route in a level-2 scenario imposes no constraints on other level-2 scenarios, allowing us to plan these scenarios in parallel (see Section 4.4).

4.3.2 Demand Aggregation by Bin Packing

To reduce the problem size and expedite the algorithm, we aggregate demands so that each batch is assigned the same routing plan. Demands $k_1, k_2, \ldots, k_n$ can be aggregated into a single demand k' if and only if they share the same terminal nodes, i.e., $\{s_{k_1}, t_{k_1}\} = \{s_{k_2}, t_{k_2}\} = \cdots = \{s_{k_n}, t_{k_n}\}$, and their total traffic volume does not exceed the bandwidth capacity of a wavelength, i.e., $b_{k_1} + b_{k_2} + \cdots + b_{k_n} \leq B$. For

the aggregated demand k', we have $s_{k'} = s_{k_1}, t_{k'} = t_{k_1}, b_{k'} = b_{k_1} + b_{k_2} + \cdots + b_{k_n}$, and demands $k_1, k_2, \ldots, k_n$ use the same route as k' in each scenario, i.e., $r_{k_1}^{e_1e_2} = r_{k_2}^{e_1e_2} = \cdots = r_{k_n}^{e_1e_2} = r_{k'}^{e_1e_2}, \forall (e_1, e_2) \in \Omega$.

For each pair of terminal nodes $\{s, t\}$, decisions on aggregating demands $\mathcal{K}_{st} = \{k \in \mathcal{K} : \{s_k, t_k\} = \{s, t\}\}$ are made by solving a bin-packing problem. Each demand $k \in \mathcal{K}_{st}$ has a weight b_k, and each bin has a capacity of B. The goal is to pack demands $\mathcal{K}_{st}$ using the minimum number of bins, thereby reducing the number of aggregated demands and the number of routing decisions. To efficiently solve this bin-packing problem, we first sort demands $\mathcal{K}_{st}$ in non-increasing order of weights, then apply the well-known first-fit policy: For each demand, place it in the first bin with sufficient remaining capacity; If no such bin exists, place it in a new bin. Demands within a bin are aggregated into a single demand, and we denote $\mathcal{K}'$ as the set of demands after aggregation.

4.3.3 Labeling Algorithm for Route Planning

For a demand $k \in \mathcal{K}'$ and scenario $(e_1, e_2) \in \Omega$, the labeling algorithm seeks to search a route $r_k^{e_1e_2}$ that can transmit demand k in scenario (e_1, e_2) while satisfying side constraints. Denote $\mathcal{L}'$ as the set of established lightpaths, and $y_{kl}^{e_1e_2}, k \in \mathcal{K}', l \in \mathcal{L}', (e_1, e_2) \in \Omega : e_1, e_2 \notin \mathcal{E}(l)$ as the binary variable that equals one if and only if lightpath l is used by demand k in scenario (e_1, e_2), i.e., $l \in \mathcal{L}(r_k^{e_1e_2})$. Then we know that $C_l^{e_1e_2} = \sum_{k \in \mathcal{K}'} b_k \max\{y_{kl}^{00}, y_{kl}^{e_1e_2}\}$ is the bandwidth consumption on lightpath $l \in \mathcal{L}'$ in scenario $(e_1, e_2) \in \Omega$, according to bandwidth capacity constraints (4.3). At the start of the constructive heuristic, $\mathcal{L}'$ is empty, and both $\boldsymbol{y} = (y_{kl}^{e_1e_2})_{k \in \mathcal{K}', l \in \mathcal{L}', (e_1,e_2) \in \Omega : e_1, e_2 \notin \mathcal{E}(l)}$ and $\boldsymbol{C} = (C_l^{e_1e_2})_{l \in \mathcal{L}', (e_1,e_2) \in \Omega}$ are zero vectors. The underlying graph of the labeling algorithm is detailed in Section 4.3.3.1, with arc costs defined in Section 4.3.3.2, label definition and extension developed in

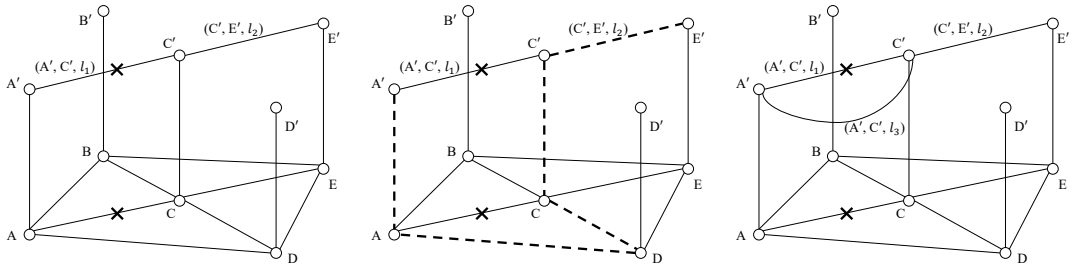
Section 4.3.3.3, and heuristic dominance rules provided in Section 4.3.3.4.

$$\sum_{k\in\mathcal{K}'} b_k \max\{y_{kl}^{00}, y_{kl}^{e_1e_2}\} \leq B, \quad \forall l \in \mathcal{L}', (e_1, e_2) \in \Omega : e_1, e_2 \notin \mathcal{E}(l). \tag{4.3}$$

4.3.3.1 Two-Layer Graph

To represent the relationships between the physical layer (links and wavelengths) and the logical layer (lightpaths) for STG2, we utilize a two-layer graph, named as LBAG and denoted by $\mathcal{G} = (\mathcal{N} \cup \mathcal{N}', \mathcal{A}_p \cup \mathcal{A}_l \cup \mathcal{A}_t \cup \mathcal{A}_r)$, which was initially proposed by Yao and Ramamurthy [2005b] for the problem without considering link failures. The physical layer consists of the set of physical nodes $\mathcal{N} = \{1, 2, \ldots, |\mathcal{N}|\}$ and the set of directed physical arcs $\mathcal{A}_p = \{(i_1, i_2, e), (i_2, i_1, e) : e \in \mathcal{E}, i_1, i_2 \text{ are endpoints of link } e\}$, where each link $e \in \mathcal{E}$ is associated with two physical arcs. Note that there may be multiple links between a pair of nodes. For each physical arc $a \in \mathcal{A}_p$, denote $e(a)$ as the associated link. The logical layer consists of the set of logical nodes $\mathcal{N}' = \{1', 2', \ldots, |\mathcal{N}|'\}$, where node $i' \in \mathcal{N}'$ is a copy of node $i \in \mathcal{N}$, and the set of directed logical arcs $\mathcal{A}_l = \{(i'_1, i'_2, l), (i'_2, i'_1, l) : l \in \mathcal{L}', i_1, i_2 \text{ are endpoints of lightpath } l\}$, where each lightpath $l \in \mathcal{L}'$ is associated with two logical arcs. Note that there may be multiple lightpaths between a pair of nodes. For each logical arc $a \in \mathcal{A}_l$, denote $l(a)$ as the associated lightpath and $\mathcal{E}(a)$ as the set of links traversed by lightpath $l(a)$. The sets $\mathcal{A}_t = \{(i', i) : i \in \mathcal{N}\}$ and $\mathcal{A}_r = \{(i, i') : i \in \mathcal{N}\}$ are transmitter arcs and receiver arcs, respectively, connecting the physical and logical layers.

To find a route for demand $k \in \mathcal{K}'$, we develop a labeling algorithm that searches for a path on the LBAG, connecting endpoints s'_k and t'_k. Traversing a logical arc $a \in \mathcal{A}_l$ indicates reusing an established lightpath $l(a)$. Sequential traversal of a transmitter arc $(i'_1, i_1) \in \mathcal{A}_t$, physical arcs $a_1, a_2, \ldots, a_n \in \mathcal{A}_p$, and a re-



(a) AC fails and l_1 is invalid. (b) Find a path (dotted) on the LBAG. (c) Establish a lightpath l_3.

Figure 4.5: Routing for demand (A, E, 50 Gbps) in scenario (AC, 0), where link AC fails. Figure 4.5(a) shows an LBAG where lightpath l_1 is invalid. Figure 4.5(b) shows that the labeling algorithm finds a path sequentially traversing arcs (A′, A), (A, D), (D, C), (C, C′), (C′, E′, l_2). Figure 4.5(c) shows that a new lightpath l_3 is established, and the demand will use route (l_3, l_2) in the scenario.

ceiver arc $(i_{n+1}, i'_{n+1}) \in \mathcal{A}_r$ implies that a new lightpath $(\lambda, (e(a_1), e(a_2), \ldots, e(a_n)))$ should be established, where λ is a wavelength unoccupied on these links, i.e., $\lambda \in \bigcap_{m=1,2,\ldots,n} \Lambda(e(a_m))$. Here, $\Lambda(e) = \Lambda \setminus \{\lambda(l) : l \in \mathcal{L}', e \in \mathcal{E}(l)\}$ represents the set of unoccupied wavelengths on link e. For example, Figure 4.5(b) shows that the labeling algorithm finds a path sequentially traversing arcs (A′, A), (A, D), (D, C), (C, C′), (C′, E′, l_2). Traversing the logical arc (C′, E′, l_2) indicates reusing the established lightpath l_2. Sequential traversal of the transmitter arc (A′, A), physical arcs (A, D), (D, C), and the receiver arc (C, C′) implies that a new lightpath should be established. The new lightpath is determined by links AD, DC and an unoccupied wavelength on them.

4.3.3.2 Arc Costs and Lower Bounds on Path Costs

To effectively guide the labeling algorithm to search for routes of good quality, we need to compute an estimated cost for each label, which will be explained in Section 4.3.3.3. To achieve this, we need to set arc costs according to Table 4.2, where $\alpha, \beta, \gamma, \theta$ are positive hyperparameters and $\mathbb{I}$ is an indicator function. The design rationales of arc cost settings are summarized as follows: We set α to regulate the

Table 4.2: Arc costs

Arc Type	Cost
Transmitter arc $a = (i', i) \in \mathcal{A}_t$	$c(a) = \alpha$
Receiver arc $a = (i, i') \in \mathcal{A}_r$	$c(a) = 0$
Physical arc $a = (i_1, i_2, e) \in \mathcal{A}_p$	$c(a) = 1 + \beta \left(1 - \frac{\lvert\Lambda(e(a))\rvert}{\lvert\Lambda\rvert}\right)^{\theta} + \gamma \mathbb{I}_{\chi}(e(a))$
Logical arc $a = (i'_1, i'_2, l) \in \mathcal{A}_l$	$c(a) = \lvert\mathcal{E}(l(a))\rvert + \gamma \lvert\mathcal{E}(l(a)) \cap \chi\rvert$

establishment of new lightpaths, configure β and θ to prevent extreme imbalances in wavelength occupation on links, and prioritize disjointness between the working route and level-1 routes by setting γ and χ. Specifically, when planning a level-1 route for demand $k \in \mathcal{K}'$, we set $\chi = \{e \in \mathcal{E}(r_k^{00}) : \text{route } r_k^{e0} \text{ has not been planned}\}$, i.e., the set of unprotected links in the working route. We set $\chi = \emptyset$ when planning the working route and level-2 routes. Moreover, the number of links traversed is considered in physical arc costs with the cost "1" and logical arc costs with the term $|\mathcal{E}(l(a))|$.

Based on the above cost settings, for any node $j \in \mathcal{N} \cup \mathcal{N}'$, we derive a lower bound on the minimum cost of paths connecting nodes j and destination t'_k. For a partial path ending at node j, we define its estimated total cost as the sum of its current cost and this lower bound. This estimate enhances the labeling algorithm in two ways. First, if the estimated total cost is no smaller than the cost of the incumbent solution, the partial path cannot yield an improving complete route and may therefore be safely pruned. Second, at each iteration of the labeling algorithm, the partial path with the smallest estimated total cost is chosen for extension. Because this estimate accounts for both the current path cost and a lower bound on the additional cost for extending the partial path to the destination, the algorithm gives priority to search directions that are closer to the destination, thereby improving the efficiency of the search procedure.

Since arc costs are nonnegative, zero is a trivial lower bound. To obtain a non-

trivial, tighter lower bound, we define the minimum hop $h(i_1, i_2)$ as the minimum number of links needed to connect nodes $i_1, i_2 \in \mathcal{N}$. Values of $h(i_1, i_2), i_1, i_2 \in \mathcal{N}$ are computed in the preprocessing stage of Algorithm 5, by setting the length of each link to one and running shortest-path algorithms. The minimum hop extends from the OTN to the LBAG, by defining $h(i'_1, i'_2) = h(i_1, i'_2) = h(i'_1, i_2) = h(i_1, i_2), i_1, i_2 \in \mathcal{N}$. Proposition 16 shows that $h(j, t'_k)$ is a valid lower bound on the minimum cost of paths connecting nodes j and t'_k.

Proposition 16. *$h(j, t'_k), j \in \mathcal{N} \cup \mathcal{N}'$ is a lower bound on the minimum cost of paths connecting nodes j and t'_k.*

Proof. Obviously, $h(i_1, i_2), i_1, i_2 \in \mathcal{N}$ satisfy triangle inequalities, i.e., $h(i_1, i_2) + h(i_2, i_3) \geq h(i_1, i_3), i_1, i_2, i_3 \in \mathcal{N}$. We define surrogate arc costs and show that they are nonnegative, i.e., $\bar{c}(a) \geq 0, a \in \mathcal{A}_p \cup \mathcal{A}_l \cup \mathcal{A}_t \cup \mathcal{A}_r$:

- For transmitter arc $a = (i', i) \in \mathcal{A}_t$, the surrogate cost $\bar{c}(a) = c(a) - h(i', t'_k) + h(i, t'_k) = c(a) = \alpha > 0$, since $h(i', t'_k) = h(i, t'_k)$.
- For receiver arc $a = (i, i') \in \mathcal{A}_r$, the surrogate cost $\bar{c}(a) = c(a) - h(i, t'_k) + h(i', t'_k) = c(a) = 0$, since $h(i, t'_k) = h(i', t'_k)$.
- For physical arc $a = (i_1, i_2, e) \in \mathcal{A}_p$, the surrogate cost $\bar{c}(a) = c(a) - h(i_1, t'_k) + h(i_2, t'_k) = c(a) - h(i_1, t_k) + h(i_2, t_k) \geq c(a) - h(i_1, i_2) = c(a) - 1 \geq 0$, where $h(i_1, i_2) + h(i_2, t_k) \geq h(i_1, t_k)$ due to triangle inequalities and $h(i_1, i_2) = 1$ since a single link e connects nodes i_1 and i_2.
- For logical arc $a = (i'_1, i'_2, l) \in \mathcal{A}_l$, the surrogate cost $\bar{c}(a) = c(a) - h(i'_1, t'_k) + h(i'_2, t'_k) \geq |\mathcal{E}(l(a))| - h(i_1, t_k) + h(i_2, t_k) \geq |\mathcal{E}(l(a))| - h(i_1, i_2) \geq 0$, where $h(i_1, i_2) + h(i_2, t_k) \geq h(i_1, t_k)$ due to triangle inequalities and $|\mathcal{E}(l(a))| \geq h(i_1, i_2)$ since lightpath $l(a)$ corresponds to a path connecting nodes i and j.

For the shortest path $(a_1, a_2, \ldots, a_n)$ connecting nodes $j \in \mathcal{N} \cup \mathcal{N}'$ and $t'_k, k \in \mathcal{K}'$ on the LBAG, assume that the endpoints of arcs $a_1, a_2, \ldots, a_n$ are $j_1 = j, j_2, \ldots, j_n, j_{n+1} = t'_k$. Then we have

$$\begin{aligned} & \sum_{m=1}^{n} \bar{c}(a_m) = \sum_{m=1}^{n} [c(a_m) - h(j_m, t'_k) + h(j_{m+1}, t'_k)] \\ = & \sum_{m=1}^{n} c(a_m) - h(j_1, t'_k) + h(j_{n+1}, t'_k) = \sum_{m=1}^{n} c(a_m) - h(j, t'_k) + h(t'_k, t'_k) \\ = & \sum_{m=1}^{n} c(a_m) - h(j, t'_k) \geq 0, \end{aligned}$$

where $\sum_{m=1}^{n} c(a_m)$ is the minimum cost of paths connecting nodes j and t'_k, and the inequality follows from that $\bar{c}(a) \geq 0, a \in \mathcal{A}_p \cup \mathcal{A}_l \cup \mathcal{A}_t \cup \mathcal{A}_r$. This implies that $h(j, t'_k)$ is a lower bound on the minimum cost of paths connecting nodes j and t'_k. □

4.3.3.3 Label Definition, Feasibility, and Extension

A label $L = (f, j, \bar{c}, \bar{\rho}, d, W, V, U)$ represents a path $\rho(L) = (a_1, a_2, \ldots, a_n = f)$ on the LBAG, where arc a_1 starts at node s'_k and arc f ends at node j. The estimated cost $\bar{c} = \sum_{a \in \rho(L)} c(a) + h(j, t'_k)$ is the sum of the current cost and a lower bound (defined in Section 4.3.3.2) on the additional cost required to reach the terminal node t'_k. $\bar{\rho}$ is the last physical segment, d is the length of this segment, and W is the set of wavelengths unoccupied on the segment. If arc f is not part of the physical layer, i.e., $f \in \mathcal{A}_l \cup \mathcal{A}_t \cup \mathcal{A}_r$, then $\bar{\rho} = \emptyset, d = 0, W = \Lambda$. Otherwise, if $f \in \mathcal{A}_p$, then $\bar{\rho} = (a_m, a_{m+1}, \ldots, a_n)$ where $a_m, a_{m+1}, \ldots, a_n \in \mathcal{A}_p$ and $a_{m-1} \in \mathcal{A}_t$, $d = \sum_{a \in \bar{\rho}} d(e(a))$, and $W = \bigcap_{a \in \bar{\rho}} \Lambda(e(a))$. $V = \{s_k\} \cup \bigcup_{a \in \rho(L) \cap (\mathcal{A}_p \cup \mathcal{A}_l)} \mathcal{N}(a)$ is the set of nodes traversed by the physical routing of path $\rho(L)$, where $\mathcal{N}(a)$ is the set of nodes traversed by the physical routing of arc a, excluding the first node. $U = \{e(a) : a \in \rho(L) \cap \mathcal{A}_p\}$ is the set of links associated with physical arcs traversed

by path $\rho(L)$.

Feasibility Conditions for Label Extension When finding a route for demand $k \in \mathcal{K}'$ in scenario $(e_1, e_2) \in \Omega$, the extension of a label $L_1 = (f_1, j_1, \bar{c}_1, \bar{\rho}_1, d_1, W_1, V_1, U_1)$ along an arc f_2 adjacent to node j_1 is deemed feasible if and only if all the following practical side constraints and optimality conditions are satisfied:

- Simple path constraints in the working scenario: If $(e_1, e_2) = (0, 0)$, then $V_1 \cap \mathcal{N}(f_2) = \emptyset$.
- Survivable constraints in failure scenarios: If $(e_1, e_2) \in \Omega^1 \cup \Omega^2$, then $e(f_2) \notin \{e_1, e_2\}$ when $f_2 \in \mathcal{A}_p$, and $\mathcal{E}(f_2) \cap \{e_1, e_2\} = \emptyset$ when $f_2 \in \mathcal{A}_l$.
- Optical reach constraints: If $f_2 \in \mathcal{A}_p$, then $d_1 + d(e(f_2)) \leq \bar{d}$.
- Wavelength continuity constraints: If $f_2 \in \mathcal{A}_p$, then $W_1 \cap \Lambda(e(f_2)) \neq \emptyset$.
- Bandwidth capacity constraints: If $f_2 \in \mathcal{A}_l$, then $C^{e_1 e_2}_{l(f_2)} + b_k \leq B$ when $(e_1, e_2) = (0, 0)$ and $C^{e_1 e_2}_{l(f_2)} + b_k(1 - y^{00}_{kl(f_2)}) \leq B$ when $(e_1, e_2) \in \Omega^1 \cup \Omega^2$.
- Necessary conditions for minimum-cost paths: If $f_2 \in \mathcal{A}_p$, then $e(f_2) \notin U_1$.

Label Extension If label $L_1 = (f_1, j_1, \bar{c}_1, \bar{\rho}_1, d_1, W_1, V_1, U_1)$ can be extended along arc f_2 with head j_1 and tail j_2, then the concatenation of them yields a label $L_2 = (f_2, j_2, \bar{c}_2, \bar{\rho}_2, d_2, W_2, V_2, U_2)$ where (i) $\bar{c}_2 = \bar{c}_1 + c(f_2) - h(j_1, t'_k) + h(j_2, t'_k)$, (ii) if $f_2 \in \mathcal{A}_l \cup \mathcal{A}_t \cup \mathcal{A}_r$, then $\bar{\rho}_2 = \emptyset, d_2 = 0, W_2 = \Lambda$, (iii) if $f_2 \in \mathcal{A}_p$, then $\bar{\rho}_2 = \bar{\rho}_1 \cup \{f_2\}, d_2 = d_1 + d(e(f_2)), W_2 = W_1 \cap \Lambda(e(f_2))$, (iv) $V_2 = V_1 \cup \mathcal{N}(f_2)$, and (v) if $f_2 \in \mathcal{A}_p$, then $U_2 = U_1 \cup \{e(f_2)\}$.

The extension procedure follows Algorithm 6, which iteratively extends labels of the smallest estimated costs to generate new labels and search routes. In line 9, a route $r^{e_1 e_2}_k$ is generated by sequentially concatenating lightpaths associated with

path $\rho(L_2)$, where new lightpaths might be established. Specifically, if $\rho(L_2) = (\ldots, a_m, a_{m+1}, \ldots, a_n, a_{n+1}, \ldots)$, where $a_{m-1} \in \mathcal{A}_t$, $a_m, a_{m+1}, \ldots, a_n \in \mathcal{A}_p$, and $a_{n+1} \in \mathcal{A}_r$, then a new lightpath $(\lambda, (e(a_m), e(a_{m+1}), \ldots, e(a_n)))$ is established, where $\lambda = \min \bigcap_{i=m,m+1,\ldots,n} \Lambda(e(a_i))$. This means choosing the wavelength with the smallest index among those unoccupied on links $e(a_m), e(a_{m+1}), \ldots, e(a_n)$, to increase the likelihood that there is a wavelength unoccupied on most links, thereby increasing the potential to apply dominance rule 3 (see Section 4.3.3.4). New lightpaths are added to $\mathcal{L}'$, and the LBAG $\mathcal{G}$ is updated accordingly. Moreover, we set $y_{kl}^{e_1e_2} = 1, \forall l \in \mathcal{L}(r_k^{e_1e_2})$.

Algorithm 6 Labeling Algorithm for Route Planning

Input: Established lightpaths $\mathcal{L}'$, bandwidth consumption $\boldsymbol{C}$, incomplete solution $\boldsymbol{y}$, LBAG $\mathcal{G}$, demand k, scenario (e_1, e_2)
Output: Route $r_k^{e_1e_2}$

Initialize the priority queue of labels as $\mathcal{S} = \{L_0\}$ where $L_0 = (\emptyset, s'_k, h(s'_k, t'_k), \emptyset, 0, \Lambda, \emptyset, \emptyset)$
while $\mathcal{S} \neq \emptyset$ **do**
    Find a label $L_1 = (f_1, j_1, \bar{c}_1, \bar{\rho}_1, d_1, W_1, V_1, U_1)$ where $\bar{c}_1 = \min\{\bar{c}(L) : L \in \mathcal{S}\}$, i.e., the label in $\mathcal{S}$ with the smallest estimated cost
    Let $\mathcal{S} \leftarrow \mathcal{S} \setminus \{L_1\}$
    for all arc $f_2 \in \mathcal{G}$ adjacent to node j_1 **do**
        if L_1 can be extended along f_2 **then** (Capacity check, Appendix B.1.1)
            Concatenate L_1 and f_2 to form a label $L_2 = (f_2, j_2, \bar{c}_2, \bar{\rho}_2, d_2, W_2, V_2, U_2)$
            if $j_2 = t'_k$ **then**
                According to path $\rho(L_2)$, generate route $r_k^{e_1e_2}$ and update $\mathcal{L}'$, $\mathcal{G}$, $\boldsymbol{y}$
                Update bandwidth $\boldsymbol{C}_l, \forall l \in \mathcal{L}(r_k^{e_1e_2})$ (Appendix B.1.2)
                Terminate the labeling algorithm
            else
                Apply dominance rules to L_2 and labels in $\mathcal{S}$ (Section 4.3.3.4)
                Let $\mathcal{S} \leftarrow \mathcal{S} \cup \{L_2\}$ if L_2 has not been removed and $W_2 \neq \emptyset$

4.3.3.4 Heuristic Dominance Rules

To reduce the number of labels and accelerate the exploration, we propose several heuristic dominance rules as shown in Table 4.3. They are relaxations of exact dominance rules so that fewer labels need to be explored at the sacrifice of optimality.

Table 4.3: Heuristic dominance rules between two labels $L_1 = (f_1, j_1, \bar{c}_1, \bar{\rho}_1, d_1, W_1, V_1, U_1)$ and $L_2 = (f_2, j_2, \bar{c}_2, \bar{\rho}_2, d_2, W_2, V_2, U_2)$

	Rule 1	Rule 2	Rule 3
Condition (i): $j_1 = j_2, j_1 \in \mathcal{N}'$	✓		
Condition (ii): $j_1 = j_2, j_1 \in \mathcal{N}$		✓	✓
Condition (iii): $\bar{c}_1 \leq \bar{c}_2$	✓	✓	✓
Condition (iv): $d_1 \leq d_2$		✓	✓
Condition (v): $\bigcap_{e \in \mathcal{E} \setminus \{e_1, e_2\}: \Lambda(e) \neq \emptyset} \Lambda(e) \neq \emptyset$			✓

The design rationales are summarized as follows: Conditions (i) and (ii) require that both labels end at the same node, with the node being on the logical and physical layer, respectively; Condition (iii) stipulates that label L_1 must have an equal or smaller cost; Condition (iv) addresses optical reach constraints, applicable to labels ending at a physical node.

For two labels $L_1 = (f_1, j_1, \bar{c}_1, \bar{\rho}_1, d_1, W_1, V_1, U_1)$ and $L_2 = (f_2, j_2, \bar{c}_2, \bar{\rho}_2, d_2, W_2, V_2, U_2)$: If conditions (i) and (iii) specified by Rule 1 are satisfied, we remove L_2 from the priority queue $\mathcal{S}$; If conditions (ii)–(iv) specified by Rule 2 are satisfied, we set $W_2 \leftarrow W_2 \setminus W_1$, and further remove L_2 from the priority queue $\mathcal{S}$ in case that W_2 becomes empty.

Moreover, when condition (v) is satisfied, Rule 2 can be lifted to Rule 3. In this case, $\bigcap_{e \in \mathcal{E}'} \Lambda(e) \neq \emptyset$, i.e., there is a wavelength unoccupied on all links in $\mathcal{E}'$, where $\mathcal{E}' = \{e \in \mathcal{E} \setminus \{e_1, e_2\} : \Lambda(e) \neq \emptyset\}$ is the set of non-failed links with unoccupied wavelengths. Since any path generated by the labeling algorithm must not traverse physical arcs associated with links in $\mathcal{E} \setminus \mathcal{E}'$ and can traverse physical arcs associated with each link $e \in \mathcal{E}'$ at most once, wavelength continuity constraints are always satisfied by assigning the newly established lightpath with a wavelength $\lambda \in \bigcap_{e \in \mathcal{E}'} \Lambda(e)$. Therefore, when conditions (ii)–(v) specified by Rule 3 are satisfied, we directly remove L_2 from the priority queue $\mathcal{S}$.

4.3.4 Reuse-then-Search Strategy for Route Planning

To reduce the number of calls to the labeling algorithm and enhance efficiency, we adopt a reuse-then-search strategy for route planning. As shown in lines 9 and 14 of Algorithm 5, the labeling algorithm is called only when no existing route can be reused. Algorithm 7 shows how to identify routes that can be reused. For each pair of terminal nodes $\{s, t\}$, denote $\mathcal{R}'_{st}$ as the set of existing routes that have been generated with endpoints s and t. A route $r \in \mathcal{R}'_{st}$ can be reused to protect demand k in the failure scenario $(e_1, e_2) \in \Phi_k \cup \Psi_k \cup \Theta_k$ if and only if: $\{s_k, t_k\} = \{s, t\}$, route r does not traverse any failed link in $\{e_1, e_2\} \cap \mathcal{E}$, and bandwidth capacity constraints are satisfied.

Algorithm 7 Route Reuse

Input: Established lightpaths $\mathcal{L}'$, bandwidth consumption $\boldsymbol{C}$, incomplete solution $\boldsymbol{y}$, demand k, failure scenario $(e_1, e_2) \in \Phi_k \cup \Psi_k \cup \Theta_k$
Output: Route $r_k^{e_1 e_2}$

1: **for all** route $r \in \mathcal{R}'_{s_k t_k}$ **and** $\mathcal{E}(r) \cap \{e_1, e_2\} = \emptyset$ **do**
2: **if** Route r satisfy bandwidth capacity constraints **then** (Appendix B.1.1)
3: Let $r_k^{e_1 e_2} \leftarrow r$, i.e., demand k uses route r in scenario (e_1, e_2)
4: Let $y_{kl}^{e_1 e_2} \leftarrow 1, \forall l \in \mathcal{L}(r_k^{e_1 e_2})$
5: Update bandwidth consumption $\boldsymbol{C}_l, \forall l \in \mathcal{L}(r_k^{e_1 e_2})$ (Appendix B.1.2)
6: Terminate with successful reuse of route $r_k^{e_1 e_2}$
7: Terminate without reuse of routes

4.4 Enhancement by Parallel Computing

For the STG2 instances (with thousands of links) addressed in this study, millions of scenarios are considered. With the continuous growth of network sizes, the number of scenarios in practical STG2 instances may further increase. To manage the large problem sizes and reduce computing time, we parallelize our newly proposed constructive heuristic by employing multithreaded parallel computing, a common approach for solving large-scale optimization problems [Schryen, 2020]. Specifically, as outlined in Algorithm 8, we adapt the hierarchical constructive heuristic by

planning level-2 scenarios in parallel. This approach is justified for two reasons. First, the number of level-2 scenarios ($|\mathcal{E}|^2 - |\mathcal{E}|$) significantly exceeds that of other scenarios ($1 + |\mathcal{E}|$), making them the primary bottleneck. Second, once the working and level-1 scenarios are planned, the interdependency between level-2 scenarios decreases, allowing for parallel planning.

Nevertheless, two challenges persist. First, different threads may simultaneously establish substitutable lightpaths, potentially increasing the objective value. Second, race conditions (multiple threads access and modify shared data concurrently) necessitate thread-safe data structures and operations, potentially compromising algorithm efficiency. To address these challenges, we adopt a "master-slave" framework, executing multiple slave threads and a single master thread alternately. In each iteration, the slave threads plan different scenarios in parallel without establishing new lightpaths. The master thread, capable of establishing new lightpaths, handles scenarios that the slave threads cannot plan (referred to as "blocked" scenarios), due to insufficient lightpaths. This approach not only resolves the first challenge, but also mitigates race conditions, as $\mathcal{L}'$ and $\mathcal{G}$ become constant when running slave threads in parallel. To further avoid race conditions when accessing and modifying scenario statuses (indicating whether a scenario has been planned successfully, is blocked, or has not been planned), we assign slave threads disjoint pools of scenarios before running them in parallel. We also design stopping criteria to minimize thread idleness.

Algorithm 8 operates as follows. After planning the working scenario and level-1 scenarios (lines 1–11 of Algorithm 5), it iterates until all level-2 scenarios are planned. Each iteration consists of three main phases:

- Phase 1 (lines 4–5). In line 4, we clear the set of blocked scenarios $\tilde{\Gamma}$, and initialize the bandwidth consumption in blocked scenarios $\tilde{\boldsymbol{C}}$. In line 5, we

initialize the pool of scenarios Γ_t for slave thread $t = 1, 2, \ldots, M$, as scenarios in $\hat{\Omega}^2$ with indices $\{t + iM : i \in \mathbb{Z}_+, i < NS, t + iM \leq |\hat{\Omega}^2|\}$. Here, M is the number of slave threads, NS is the size limit on a scenario pool, and $\mathbb{Z}_+$ is the set of non-negative integers. This approach balances the workload of slave threads, as scenarios in $\hat{\Omega}^2$ are sorted in non-increasing order of the number of remaining demands to be planned.

- Phase 2 (line 6). Plan scenarios in $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$ using slave threads $1, 2, \ldots, M$ in parallel, without establishing new lightpaths. The returned values for each slave thread t include the set of successfully planned scenarios $\bar{\Gamma}_t$, the set of blocked scenarios $\tilde{\Gamma}_t$, and the bandwidth consumption $\tilde{\boldsymbol{C}}(t)$ in blocked scenarios. Stopping criteria for slave threads are detailed in Appendix B.2.1. Particularly, once a thread terminates, all other threads will promptly terminate to minimize thread idleness.

- Phase 3 (line 8). Use the master thread to plan blocked scenarios $\tilde{\Gamma}$ that slave threads could not handle, allowing to establish new lightpaths. For details, see Appendix B.2.2.

Algorithm 8 Enhancement by Parallel Computing

Input: $\mathcal{N}, \mathcal{E}, \Lambda, \mathcal{K}, B, \bar{d}$, and hyperparameters M, NS
Output: Routes $r_k^{e_1 e_2}, k \in \mathcal{K}, (e_1, e_2) \in \Omega$

1: Plan the working scenario and level-1 scenarios as lines 1–11 of Algorithm 5
2: Initialize the set of level-2 scenarios to plan: $\hat{\Omega}^2 \leftarrow \Omega^2$
3: **while** $\hat{\Omega}^2 \neq \emptyset$ **do**
4: Let blocked scenarios $\tilde{\Gamma} \leftarrow \emptyset$, bandwidth consumption in blocked scenarios $\tilde{\boldsymbol{C}} \leftarrow \emptyset$
5: Initialize scenario pools $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$ for threads $1, 2, \ldots, M$
6: Plan scenarios in $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$ by slave threads $1, 2, \ldots, M$ in parallel: $\bar{\Gamma}_t, \tilde{\Gamma}_t, \tilde{\boldsymbol{C}}(t) \leftarrow$ SlavePlan(Γ_t) (Slave threads in parallel, see B.2.1)
7: Update blocked scenarios and bandwidth consumption in these scenarios: $\tilde{\Gamma} \leftarrow \bigcup_{t=1}^{M} \tilde{\Gamma}_t, \tilde{\boldsymbol{C}} \leftarrow \bigcup_{t=1}^{M} \tilde{\boldsymbol{C}}(t)$
8: MasterPlan$(\tilde{\Gamma}, \tilde{\boldsymbol{C}})$ (Plan blocked scenarios, see B.2.2)
9: Update unplanned scenarios: $\hat{\Omega}^2 \leftarrow \hat{\Omega}^2 \setminus \bigcup_{t=1}^{M} \bar{\Gamma}_t \setminus \tilde{\Gamma}$

4.5 Computational Experiments

The computational experiments conducted in this study have three primary objectives: (i) to assess the performance of our proposed solution method on large-scale STG2 instances by comparing it with the benchmark method provided by our industry partner, Huawei Technologies; (ii) to evaluate the effectiveness of the algorithmic enhancements; and (iii) to measure the optimality gap achieved by our method on instances that can be solved by the general optimization solver CPLEX. The benchmark method provided by our industry partner is also a constructive heuristic with post-processing to enhance the objective value, and its details are unavailable due to confidentiality. During our study, we developed several other constructive heuristics, including the one that plans demands sequentially with an outer loop iterating over demands and an inner loop over scenarios. Nonetheless, our computational results indicate that these alternative heuristics yield solutions inferior to those from the benchmark method.

We implement our hierarchical constructive heuristic in C++ and conduct experiments on a server equipped with Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz. In our hierarchical constructive heuristic, for the hyperparameters related to arc costs (Section 4.3.3.2), we set $\alpha = 32$, $\beta = 32$, $\gamma = 2$, and $\theta = 3$ based on preliminary experiments. In the parallel algorithm (Section 4.4), we set the size limit of the scenario pool for each slave thread to NS =10,000. Our algorithm is compared first with the benchmark method provided by our industry partner, and subsequently with the commercial solver CPLEX.

4.5.1 Comparison with Benchmark method

In this section, we compare our algorithm with the benchmark method on 16 large-scale instances provided by our industry partner. These instances comprise

networks with 1,000–2,600 nodes, 1,928–3,091 links, and 10,020–12,980 demands, while accounting for all single-link and double-link failure scenarios. Notably, the number of double-link failure scenarios exceeds 9 million in instances with more than 3,000 links. In each instance, the number of available wavelengths is $|\Lambda| = 120$ and the bandwidth of each wavelength is $B = 100$ Gbps. The memory is limited to 16 GB as required by our industry partner.

Table 4.4 presents the overall results. CH-1 denotes the hierarchical constructive heuristic without parallelism, and CH-M (with $M \geq 2$) denotes the enhanced variant with M slave threads. For each algorithm, column "Obj" reports the objective value, and "Time" reports the computing time. In comparison to the benchmark method, Imp_B measures the improvement in objective value (i.e., $Imp_B = \frac{\text{Obj of Benchmark - Obj of CH-}M}{\text{Obj of Benchmark}}$), and TR_B represents the ratio of computing time (i.e., $TR_B = \frac{\text{Time of Benchmark}}{\text{Time of CH-}M}$).

The results show that our algorithms outperform the benchmark method in terms of objective value, with the exception of one instance ($|\mathcal{N}|$ =2,000). CH-8 achieves the best overall performance, yielding an average objective value improvement of 18.6% compared with the benchmark. In terms of efficiency, CH-8 is 66.5 times faster than the benchmark method. Specifically, CH-8 averages 14 minutes per instance (with a maximum of 36 minutes), whereas the benchmark method averages 11.3 hours per instance (with a maximum of 21.1 hours). Besides, while CH-1 already provides large speedups over the benchmark, moving to CH-4 and CH-8 further reduces computing time with negligible differences in objective value.

Table 4.5 evaluates the effectiveness of parallelization in detail, by displaying the computing time using different numbers of threads. "Time$_i$" denotes the computing time of planning level-i scenarios. For brevity, the single-threaded

Table 4.4: Computational results of the benchmark method and our algorithms

Instance				Benchmark		CH-1		CH-4		CH-8	
$\|\mathcal{N}\|$	$\|\mathcal{E}\|$	$\|\mathcal{K}\|$	d (km)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)
1000	1928	10020	360	7151	51051	5910	2683.6	5894	694.5	5869	404.7
1000	1928	10020	600	7222	37080	5879	2687.2	5864	636.1	5872	383.3
1000	1928	10020	900	6870	34603	5208	2128.2	5171	550.3	5186	309.3
1000	1928	10020	1500	6728	34305	4756	1535.6	4785	492.9	4768	289.0
1200	2365	10390	600	7927	37137	6605	3726.4	6573	1024.8	6597	608.7
1200	2365	10390	900	7615	28259	5868	2548.6	5827	792.7	5846	471.3
1200	2365	10390	1500	7685	28017	5457	2709.0	5523	681.2	5501	415.5
1600	3091	10070	600	8260	73270	6703	5248.6	6652	1584.8	6689	847.0
1600	3091	10070	900	7473	75978	5746	3698.0	5766	1287.4	5745	663.0
2000	3083	12980	600	7596	20845	8602	16928.1	8526	3202.5	8547	1766.9
2200	3052	11054	600	10405	37137	9045	19736.1	8949	3455.0	8902	2155.6
2200	3052	11054	900	8540	44990	6896	7831.3	6861	2131.7	6886	1300.8
2200	3052	11054	1500	8208	38184	5553	6783.7	5508	1609.7	5492	1002.8
2600	3027	10930	600	10151	68144	8831	9579.3	8737	1884.3	8765	1267.3
2600	3027	10930	900	7729	27941	6663	5381.6	6563	1223.4	6597	806.1
2600	3027	10930	1500	5749	18637	5000	3442.5	4948	1019.9	4984	613.3
Average Imp_B and TR_B						18.2%	10.6	18.6%	38.9	18.6%	66.5

phases (Time_0 and Time_1) are only reported for CH-1, as they are not affected by parallelization and differ slightly in different settings. "*Speedup*" indicates the acceleration factor over CH-1 when planning level-2 scenarios in parallel, i.e., $Speedup = \frac{\text{Time}_2 \text{ of CH-1}}{\text{Time}_2 \text{ of CH-}M}$.

When increasing the number of slave threads from 1 (CH-1) to 4 (CH-4), the average speedup improves from 1.0 to 4.3, which is even better than linear scaling. This fluctuation may be caused by parallel computing altering the sequencing of planning scenarios. Further increasing to 8 slave threads (CH-8) leads to an average speedup of 7.6, which is still near-linear but slightly less than the ideal factor due to parallel overhead. The single-threaded phases contribute marginally to the total computing time compared with Time_2. In addition, the average relative gap between the maximum and minimum objective values under different thread settings is 0.84%, demonstrating that parallelization accelerates computation without compromising solution quality.

To further evaluate the impact of different enhancements, we present computational results obtained by deactivating specific components of our proposed

Table 4.5: Computational results of algorithms with different numbers of slave threads

Instance				CH-1				CH-4		CH-8	
$\|\mathcal{N}\|$	$\|\mathcal{E}\|$	$\|\mathcal{K}\|$	d (km)	Obj	Time_0 (s)	Time_1 (s)	Time_2 (s)	Obj	Time_2 (s)	Obj	Time_2 (s)
1000	1928	10020	360	5910	17.19	49.3	2614.1	5894	629.6	5869	338.1
1000	1928	10020	600	5879	16.41	46.5	2621.4	5864	571.9	5872	324.0
1000	1928	10020	900	5208	14.77	36.5	2074.2	5171	496.5	5186	257.4
1000	1928	10020	1500	4756	13.22	35.1	1484.8	4785	438.0	4768	238.1
1200	2365	10390	600	6605	24.12	62.4	3635.7	6573	940.7	6597	524.1
1200	2365	10390	900	5868	21.21	42.7	2481.0	5827	720.7	5846	398.2
1200	2365	10390	1500	5457	21.38	49.4	2634.5	5523	611.8	5501	346.2
1600	3091	10070	600	6703	33.59	66.5	5143.0	6652	1471.9	6689	742.5
1600	3091	10070	900	5746	30.52	50.1	3612.0	5766	1190.4	5745	569.5
2000	3083	12980	600	8602	42.98	144.2	16732.7	8526	3011.9	8547	1586.6
2200	3052	11054	600	9045	38.78	156.2	19533.2	8949	3239.1	8902	1945.7
2200	3052	11054	900	6896	34.09	88.0	7702.1	6861	1963.7	6886	1129.6
2200	3052	11054	1500	5553	35.31	92.3	6648.8	5508	1471.2	5492	866.9
2600	3027	10930	600	8831	20.28	76.8	9475.8	8737	1783.4	8765	1155.5
2600	3027	10930	900	6663	19.13	60.9	5295.3	6563	1134.1	6597	715.3
2600	3027	10930	1500	5000	14.31	54.3	3367.9	4948	941.7	4984	532.5
Average *Speedup*							1.0		4.3		7.6

algorithm. Table 4.6 summarizes the notations used for the algorithm with various configurations. All these algorithm variants are enhanced with parallel computing (Section 4.4) using 8 slave threads, with CH equivalent to CH-8 for brevity. CH-D deactivates demand aggregation (Section 4.3.2). CH-B do not use lower bounds on path costs in the labeling algorithm (Section 4.3.3.2). Specifically, the estimated cost of a label associated with path $\rho = (a_1, a_2, \ldots, a_n)$ ending at node j, is computed as $\bar{c} = \sum_{a \in \rho} c(a)$ instead of $\bar{c} = \sum_{a \in \rho} c(a) + h(j, t'_k)$. CH-R does not apply the reuse-then-search strategy (Section 4.3.4), and CH-R-DP further deactivates dominance rules on the physical layer, i.e., Rule 2 and Rule 3 in Table 4.3. In CH-R-DP, we do not allow reusing generated routes when planning a route, so that the labeling algorithm is always called in order to show the effectiveness of dominance rules. Tests on Rule 1 in Table 4.3 are not reported, since preliminary experiments show that disabling it leads to an out-of-memory error, even for the smallest-scale instances (1,000 nodes) within a 128 GB memory limit.

In Tables 4.7 and 4.8, compared to CH, Imp_C represents the improvement in objective value, and TR_C represents the ratio of computing times. All average

Table 4.6: Hierarchical constructive heuristic with various enhancements

Notation	Demand Aggregation	Lower Bounds on Path Costs	Reuse-then-Search	Dominance Rules 2 and 3
CH	✓	✓	✓	✓
CH-D		✓	✓	✓
CH-B	✓		✓	✓
CH-R	✓	✓		✓
CH-R-DP	✓	✓		

values are computed over instances where feasible solutions are obtained. Table 4.7 evaluates the effectiveness of demand aggregation. The number of aggregated demands is shown in column "$|\mathcal{K}'|$", which is approximately one-tenth of the original. Without demand aggregation, CH-D cannot solve four instances within the memory limit of 16 GB due to the excessive number of generated routes for the large demand set. Additionally, CH-D achieves solutions that are 3.6% worse on average, while requiring 3.8 times the computing time.

Table 4.7: Computational results of disabling/allowing demand aggregation

Instance				CH-D				CH		
$\|\mathcal{N}\|$	$\|\mathcal{E}\|$	$\|\mathcal{K}\|$	d (km)	Obj	Imp_C	Time (s)	TR_C	Obj	Time (s)	$\|\mathcal{K}'\|$
1000	1928	10020	360	5889	-0.3%	1491.9	3.7	5869	404.7	1002
1000	1928	10020	600	5898	-0.4%	1290.4	3.4	5872	383.3	1002
1000	1928	10020	900	5083	2.0%	1142.1	3.7	5186	309.3	1002
1000	1928	10020	1500	4686	1.7%	879.5	3.0	4768	289.0	1002
1200	2365	10390	600	6681	-1.3%	2089.8	3.4	6597	608.7	1039
1200	2365	10390	900	5836	0.2%	1446.7	3.1	5846	471.3	1039
1200	2365	10390	1500	5459	0.8%	1266.1	3.0	5501	415.5	1039
1600	3091	10070	600	6640	0.7%	3098.2	3.7	6689	847.0	1007
1600	3091	10070	900	5605	2.4%	2356.8	3.6	5745	663.0	1007
2000	3083	12980	600	-	-	-	-	8547	1766.9	1298
2200	3052	11054	600	-	-	-	-	8902	2155.6	1106
2200	3052	11054	900	-	-	-	-	6886	1300.8	1106
2200	3052	11054	1500	5853	-6.6%	4214.6	4.2	5492	1002.8	1106
2600	3027	10930	600	-	-	-	-	8765	1267.3	1028
2600	3027	10930	900	8229	-24.7%	4855.5	6.0	6597	806.1	1028
2600	3027	10930	1500	5864	-17.7%	2858.7	4.7	4984	613.3	1028
Average Imp_C and TR_C					-3.6%		3.8			

Notes. -: Out of 16 GB memory.

Table 4.8 evaluates the performance of different routing techniques. For CH-R-

DP and CH-R, due to prohibitive computing times in the first nine instances, we do not test larger-scale instances. It is observed that without the reuse-then-search strategy, CH-R requires 34.2 times of computing time on average, while CH-R-DP performs worse due to the absence of dominance rules 2 and 3. Additionally, CH-R-DP can solve only two instances within the memory limit, highlighting the importance of dominance rules 2 and 3 in reducing memory requirements. Furthermore, incorporating lower bounds on path costs provides a substantial improvement in computing time, with CH using only about half the computing time of CH-B.

Table 4.8: Computational results of adopting different routing techniques

Instance				CH-R-DP		CH-R		CH-B		CH	
$\|\mathcal{N}\|$	$\|\mathcal{E}\|$	$\|\mathcal{K}\|$	d (km)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)
1000	1928	10020	360	6019	16428	6114	13697	5849	499.5	5869	404.7
1000	1928	10020	600	6061	17127	6037	14621.3	5832	444.5	5872	383.3
1000	1928	10020	900	-	-	5336	10834.3	5186	389.8	5186	309.3
1000	1928	10020	1500	-	-	4908	7602.5	4789	359.5	4768	289
1200	2365	10390	600	-	-	6776	22332	6619	751.4	6597	608.7
1200	2365	10390	900	-	-	6044	19022.9	5844	565.9	5846	471.3
1200	2365	10390	1500	-	-	5618	12075.3	5489	533.7	5501	415.5
1600	3091	10070	600	*	*	6900	37486.1	6653	994	6689	847
1600	3091	10070	900	*	*	5856	29997.3	5718	890.3	5745	663
2000	3083	12980	600	*	*	*	*	8551	2662.3	8547	1766.9
2200	3052	11054	600	*	*	*	*	8862	4005.4	8902	2155.6
2200	3052	11054	900	*	*	*	*	6805	2245.1	6886	1300.8
2200	3052	11054	1500	*	*	*	*	5519	1517.9	5492	1002.8
2600	3027	10930	600	*	*	*	*	8824	2434.3	8765	1267.3
2600	3027	10930	900	*	*	*	*	6599	1473.2	6597	806.1
2600	3027	10930	1500	*	*	*	*	4905	968.1	4984	613.3
Average Imp_C and TR_C				-2.8%	42.6	-2.9%	34.2	0.2%	1.4		

Notes. -: Out of 16 GB memory; *: Not tested, as the algorithm variants cannot solve smaller-scale instances (with 1,000 nodes) within the memory and time limits

4.5.2 Comparison with CPLEX

For small-scale STG2 instances, whose ILP formulation presented in Section 4.2 can be solved by the general optimization solver CPLEX, we can evaluate the optimality gap for our algorithm. We generated such small-scale instances randomly, using the parameters summarized in Table 4.9. Each instance is represented by a notation that consists of the number of nodes $|\mathcal{N}|$, the number of links $|\mathcal{E}|$, the

number of demands $|\mathcal{K}|$, and an index. For example, "random-5-8-2-1" represents the first instance with 5 nodes, 8 links, and 2 demands. To better illustrate the computational effort of solving the ILP formulation, CPLEX is configured with 32 threads, a memory limit of 300 GB, and a time limit of 18,000 seconds.

Table 4.9: Parameters of randomly generated instances

Parameter	Description	Value
$\|\mathcal{N}\|$	number of nodes	random number in $\{5, 6, 7, 8\}$
$\|\mathcal{E}\|$	number of edges	random number in $[\lceil 1.5\|\mathcal{N}\|\rceil, \min(2\|\mathcal{N}\|, 11)]$
$\|\mathcal{K}\|$	number of demands	random number in $\{2, 3, 4\}$
B	bandwidth capacity	100
b_k	traffic volume of demand k	random number in $\{10, 20, 30, 40, 50\}$
d_e	the length of link e	1
$\bar{d}$	optical reach limit	2
$\|\Lambda\|$	number of wavelengths	2

Notes. The number of edges $|\mathcal{E}|$ is selected to ensure the existence of feasible solutions while approximately aligning with CPLEX's computational solvability.

The detailed results are presented in Table 4.10. For CPLEX, columns "UB_{CPX}" and "LB_{CPX}" present the upper and lower bounds on the optimal objective value obtained by CPLEX within the time limit, while "Time_{CPX}" reports the computing time in seconds. For the CH algorithm, column "Obj_{CH}" indicates the objective value achieved. Column "$\text{Gap}_{\text{CH}}^{\text{UB}}$" shows the relative gap between "Obj_{CH}" and "UB_{CPX}", while "$\text{Gap}_{\text{CH}}^{\text{LB}}$" reflects the relative gap between "Obj_{CH}" and "LB_{CPX}". Notably, the CH algorithm computes results in less than 0.1 seconds for all instances.

There are totally 41 instances tested. Among them, CPLEX produces feasible solutions for only 23 instances. In general, the ILP formulation becomes extremely challenging when $|\mathcal{N}| \geq 6$, $|\mathcal{E}| \geq 10$, and $|\mathcal{K}| \geq 3$, even though other parameters remain small. Of the 23 instances where CPLEX provides feasible solutions, it costs an average computing time of 6,493.4 seconds and fails to solve 6 of them to optimality. In contrast, the CH algorithm consistently produces high-quality solutions within 0.1 seconds for all instances. On average, the CH algorithm achieves a relative gap of 5.6% compared to the upper bound provided by CPLEX

and a gap of 11.0% compared to the lower bound. These results highlight that CH significantly outperforms CPLEX in computational efficiency while maintaining small optimality gap and good solution quality.

4.6 Summary

This study investigates a large-scale STG2 problem with partial path protection, a critical yet unexplored problem in the literature. This problem is essential for minimizing network costs while ensuring reliable telecommunications services. The combination of large-scale instances, practical constraints, and limited computing resources makes it significantly more challenging. To effectively manage the inter-dependency between scenarios imposed by various constraints, we have developed a hierarchical constructive heuristic. It involves planning the working scenario first, followed by all level-1 scenarios, and finally all level-2 scenarios. This hierarchical sequence offers several advantages: It efficiently handles consistent routing constraints through bandwidth reservation, allows bandwidth capacity constraints to be verified by querying bandwidth consumption only in the scenario being currently planned, facilitates bandwidth updates requiring fewer operations and less memory, and mitigates race conditions during parallel planning of level-2 scenarios.

To address the challenges of large problem sizes, we propose several strategies. To reduce the demand set, we aggregate demands using bin-packing solutions. For efficient route searching in large-scale networks, the labeling algorithm is enhanced by incorporating lower bounds on path costs and heuristic dominance rules, alongside a reuse-then-search routing strategy to reduce calls to the labeling algorithm. To manage the large number of level-2 scenarios, we parallelize our constructive heuristic to develop a multithreaded parallel algorithm, with mechanisms designed to optimize the objective value, mitigate race conditions, and reduce thread idleness.

The computational results indicate that our method efficiently solves large-scale STG2 instances, comprising over 2,000 nodes, 3,000 links, 10,000 demands, and 9 million failure scenarios, within an hour of computing time and 16 GB of memory. This performance significantly surpasses the scales addressed in existing traffic grooming studies. Compared to the benchmark method from our industry partner, our approach yields an average improvement of 18.6% in objective value and is 66.5 times faster in terms of computing time. Additionally, the results validate the effectiveness of the algorithmic enhancements proposed in this study. On small-scale instances, our heuristic delivers solutions with an average relative gap of only 5.6% compared to those obtained by CPLEX, all within 0.1 seconds, whereas CPLEX requires several hours of computation. These findings underscore the superiority of our method in objective value optimization, scalability, and computational efficiency.

Table 4.10: Computational results of CH and CPLEX on small-scale instances

Instance	UB_{CPX}	LB_{CPX}	$Time_{CPX}$ (s)	Obj_{CH}	Gap_{CH}^{UB}	Gap_{CH}^{LB}
random-5-8-2-1	5	5	570.9	5	0.0%	0.0%
random-5-8-3-1	5	5	143.1	5	0.0%	0.0%
random-5-8-3-2	6	6	1347.7	7	16.7%	14.3%
random-5-8-3-3	8	8	8810.3	9	12.5%	11.1%
random-5-8-3-4	8	8	3998.2	10	25.0%	20.0%
random-5-8-3-5	6	6	1640.3	6	0.0%	0.0%
random-5-8-3-6	5	5	2233.1	6	20.0%	16.7%
random-5-9-2-1	5	5	1974.7	5	0.0%	0.0%
random-5-9-3-1	7	5	18000.0	7	0.0%	28.6%
random-5-9-3-2	9	6	18000.0	9	0.0%	33.3%
random-5-9-3-3	5	5	8485.8	5	0.0%	0.0%
random-5-9-3-4	8	5	18000.0	8	0.0%	37.5%
random-6-9-2-1	5	5	616.6	5	0.0%	0.0%
random-6-9-2-2	6	6	513.6	6	0.0%	0.0%
random-6-9-2-3	5	5	506.0	5	0.0%	0.0%
random-6-9-3-1	8	8	2218.4	8	0.0%	0.0%
random-6-9-3-2	8	8	2895.8	8	0.0%	0.0%
random-6-9-3-3	8	8	1267.2	9	12.5%	11.1%
random-6-9-3-4	8	8	2633.3	10	25.0%	20.0%
random-6-9-3-5	6	6	1494.1	6	0.0%	0.0%
random-6-9-4-1	10	8	18000.0	10	0.0%	20.0%
random-6-9-4-2	9	8	18000.0	9	0.0%	11.1%
random-6-10-2-1	6	5	18000.0	7	16.7%	28.6%
random-6-10-3-1	-	-	-	9	-	-
random-6-10-3-2	-	-	-	9	-	-
random-6-10-3-3	-	-	-	9	-	-
random-6-10-3-4	-	-	-	10	-	-
random-6-10-4-1	-	-	-	9	-	-
random-6-10-4-2	-	-	-	8	-	-
random-6-10-4-3	-	-	-	11	-	-
random-6-11-3-1	-	-	-	8	-	-
random-6-11-3-2	-	-	-	9	-	-
random-6-11-4-1	-	-	-	9	-	-
random-7-11-2-1	-	-	-	6	-	-
random-7-11-3-1	-	-	-	9	-	-
random-7-11-3-2	-	-	-	9	-	-
random-7-11-3-3	-	-	-	11	-	-
random-7-11-4-1	-	-	-	10	-	-
random-7-11-4-2	-	-	-	9	-	-
random-8-12-2-1	-	-	-	7	-	-
random-8-12-4-1	-	-	-	10	-	-
Average			6493.4		5.6%	11.0%

Notes. -: Out of 16 GB memory; The computing time of CH is below 0.1 seconds for all these instances.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

This thesis addresses several critical computational challenges in the fields of routing and network design through the development of novel algorithms and optimization techniques. These fields collectively underpin the infrastructure of network systems and facilitate the efficient movement of resources, services, and information in an increasingly interconnected world. Optimizing such problems is crucial for ensuring the efficiency and reliability of network services. However, solving these problems is challenging due to the complexity of operational constraints, the vastness of solution search spaces, and the stringent requirements for system survivability.

In this thesis, we address three specific challenges in the field of routing and network design. The first study (Chapter 2) addresses the multitrip scheduling challenge in vehicle routing by developing an enhanced exact algorithm for the CMTVRPTW. This problem extends the classical VRPTW by allowing vehicles to perform multiple trips within the planning horizon. Specifically, after serving a sequence of customers and returning to the depot, a vehicle may be replenished and

dispatched again. While this extension improves vehicle utilization and aligns with the operational requirements of many real-world logistics networks, it significantly increases computational challenges by introducing a larger number of decision variables and constraints. To address this challenge, we develop an enhanced exact algorithm that constructs the initial set of trips from routes enumerated by Yang [2025] and obtains the optimal solution by solving the trip-based IP formulation. This approach improves upon the state-of-the-art exact algorithm [Yang, 2025], as the trip-based formulation enables a more efficient algorithm for the constraint separation problem compared to the superstructure-based formulation. To further enhance computational efficiency, we reduce the number of variables by eliminating trips using the dominance rule, variable fixing technique, and optimality property of vehicle departure times. Additionally, a dynamic time discovery strategy is introduced to reduce the number of identified time points, thereby decreasing the number of constraints in the formulation. Computational experiments on benchmark instances with up to 200 customers, as well as on extended instances, highlight the effectiveness of our algorithm. On 14 benchmark instances previously unsolved or considered difficult, our algorithm outperforms the state-of-the-art method proposed by Yang [2025] in terms of objective values, optimality gaps, and computational efficiency. Specifically, our algorithm consistently achieves smaller or equal incumbent objective values, proves optimality for 6 more previously unsolved instances, and reduces average computing time by over 20%. Furthermore, our algorithm exhibits superior performance on extended problem instances, further highlighting its scalability.

In the second study (Chapter 3), we propose novel optimality cuts and domain reduction techniques to reduce the solution search space for integer programs, with a particular focus on arc fixing techniques for the vehicle routing problem. Integer programming formulations for combinatorial optimization problems, such as the

vehicle routing problem, face the significant computational challenge due to the exponential growth of the solution space. To address this challenge, valid cuts and domain reduction techniques are commonly employed to prune the search space. However, existing approaches in the literature often depend on problem-specific structures or require the computation of feasible LP duals, which can constrain their broader applicability. In this study, we introduce LUCs, a novel class of optimality cuts derived from Lagrangian duals, including infeasible LP duals. We utilize LUCs to strengthen RCCs and to derive new domain reduction techniques. Additionally, we leverage LUCs to develop new arc fixing techniques for the VRPTW, based on Lagrangian duals and the tailored dual-picking strategy. Computational experiments demonstrate that compared to the feasible-LP-dual approach studied in the literature, LUC-based arc fixing reduces the average number of remaining arcs to 47.9%, while the average number of enumerated routes decreases to 31.3%. For several instances, the reduction in enumerated routes reaches orders of magnitude, underscoring the substantial effectiveness of the proposed methods.

In the third study (Chapter 4), we address the challenge of scalability and survivability against double-link failures in large-scale telecommunications networks by introducing an efficient heuristic for the STG2. Optimizing such networks is essential to minimize facility costs and maintain service reliability, but the combination of large network sizes and survivability requirements presents a substantial computational challenge. Even for simplified problem settings, such as those that consider only a subset of practical operational constraints and single-link failures, the network sizes studied in the literature typically involve no more than 300 nodes. In this study, we propose a hierarchical constructive heuristic to effectively manage the interdependency between scenarios imposed by practical constraints. It involves planning the working scenario first, followed by all level-1 scenarios, and finally

all level-2 scenarios. To address the challenges of large problem sizes, we propose several strategies: (i) To reduce the demand set, we aggregate demands using bin-packing solutions; (ii) For efficient route searching in large-scale networks, the labeling algorithm is enhanced by incorporating lower bounds on path costs and heuristic dominance rules, alongside a reuse-then-search routing strategy to reduce calls to the labeling algorithm; (iii) To manage the large number of level-2 scenarios, we parallelize our constructive heuristic to develop a multithreaded parallel algorithm, with mechanisms designed to optimize the objective value, mitigate race conditions, and reduce thread idleness. The computational results indicate that our method efficiently solves large-scale STG2 instances, comprising over 2,000 nodes, 3,000 links, 10,000 demands, and 9 million failure scenarios, within an hour of computing time and 16 GB of memory. This performance significantly surpasses the scales addressed in existing traffic grooming studies. Compared to the benchmark method from our industry partner, our approach yields an average improvement of 18.6% in objective value and is 66.5 times faster in terms of computing time.

Collectively, the research presented in this thesis advances the state-of-the-art in routing and network design problems by introducing a set of novel algorithms and optimization techniques. Through the development of an enhanced exact algorithm, innovative domain reduction techniques, and an efficient heuristic, these contributions share a unified focus: overcoming the computational challenges posed by complex operational constraints and the exponential growth of solution search spaces inherent in routing and network design problems. By integrating mathematical programming with tailored algorithmic strategies, the proposed methodologies achieve significant improvements in both solution quality and computational efficiency, even for instances previously considered intractable. Ultimately, this thesis contributes to the development of advanced algorithms and optimization techniques that address critical computational challenges, thereby enhancing the efficiency,

scalability, and reliability of modern network infrastructure.

5.2 Future Research Directions

This thesis has presented novel algorithms and optimization techniques to address critical computational challenges in routing and network design problems. While the proposed methodologies have advanced the state of the art, they also open several promising avenues for future research, underscoring the potential for extending the proposed approaches to address a broader range of problems and to develop additional advanced solution methodologies in related domains.

One direction for future work is the adaptation of the methodologies developed in this thesis to other problems. For instance, the trip-based formulation proposed for the CMTVRPTW can be extended to emerging problems such as the drone routing problem. Similarly, the LUC-based optimization techniques can be applied to other combinatorial optimization problems, including crew scheduling and assignment problems. For the STG2 heuristic, the strategies developed in this study can be adapted to optimize other types of large-scale networks, such as power or logistics networks, particularly under failure scenarios. These extensions would further validate the applicability of the techniques developed in this thesis.

Another direction for future research is to enhance the scalability and efficiency of the proposed algorithms by integrating other advanced computational methods. For the CMTVRPTW, developing new valid cuts has the potential to effectively strengthen the LP bound and improve solution efficiency. LUCs present opportunities to design novel domain reduction techniques, particularly through integration with classical cuts such as capacity cuts. For the STG2 heuristic, distributed computing and post-processing techniques could be employed to handle even larger-scale networks and improve solution quality. Moreover, refining relaxations to

calculate tighter lower bounds on the optimal objective value could help assess the quality of incumbent solutions and derive near-optimal solutions for the STG2.

In summary, the future research directions outlined above aim to build on the contributions of this thesis by extending the applicability of its algorithms, incorporating cutting-edge computational techniques, and exploring innovative solution methods. Collectively, these efforts will further advance the study of routing and network design problems, driving the development of new algorithms and optimization techniques to address increasingly complex network systems.

Appendix A

Supplements for Chapter 2

A.1 Equivalent Integer-Time Instance

In this section, we demonstrate that time data (travel times, service times, and endpoints of time windows) can be assumed to be integers without loss of generality. Consider a feasible instance with rational time data (referred to as instance A). There exists a rational number M such that multiplying all time data by M results in integers with a greatest common divisor of 1. Construct a new instance (referred to as instance B) by scaling all time data in instance A by M. Then any feasible solution to instance B can be transformed into a feasible solution to instance A with the equal objective value by dividing the departure times of the selected routes by M. Similarly, any feasible solution to instance A can be transformed into a feasible solution to instance B with the equal objective value by multiplying the departure times of the selected routes by M. Consequently, any optimal solution to instance B can be transformed to an optimal solution to instance A by scaling the departure times.

According to Azi et al. [2007], determining the values of e_r, l_r, and d_r for

$r \in \mathcal{R}$ involves only the addition and subtraction of time data. Consequently, under the assumption that all time data are integers, it follows that $\{e_r, l_r : r \in \mathcal{R}\} \subseteq \{a_0, a_0 + 1, \ldots, b_0\} \subseteq \mathbb{Z}_+$ and $\{d_r : r \in \mathcal{R}\} \subseteq \mathbb{Z}_{++}$, where $\mathbb{Z}_+$ is the set of nonnegative integers and $\mathbb{Z}_{++}$ is the set of positive integers.

A.2 Enhancement for Solving Trip-Based IP Formulation

As outlined in Algorithm 9, we enhance the algorithm for solving the trip-based IP formulation by additionally incorporating side constraints (A.1) adapted from the route-based formulation. At each iteration of Algorithm 9, we first solve $SF(\mathcal{S}_3, \mathcal{T})$ using an IP solver, where $\mathcal{S}_3$ is replaced with a subset of trips $\{(r, \tau) : r \in \mathcal{R}_3, \tau \in \mathcal{D}(\mathcal{T}, \mathcal{G}_r^3)\}$ according to Proposition 10. If the optimal solution $\boldsymbol{y}$ to $SF(\mathcal{S}_3, \mathcal{T})$ satisfies all constraints (2.8c) corresponding to the time points $\{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$, then it is feasible to $SF(\mathcal{S}_3, \mathcal{H})$ by Proposition 2. In this case, $\boldsymbol{y}$ is also optimal to the CMTVRPTW, since $SF(\mathcal{S}_3, \mathcal{T})$ is a relaxation of $SF(\mathcal{S}_3, \mathcal{H})$.

Define $\boldsymbol{x}^{\boldsymbol{y}}$ as the route-based solution associated with $\boldsymbol{y}$, where $x_r^{\boldsymbol{y}} = \sum_{s \in \mathcal{S}_0 : r_s = r} y_s$ for each $r \in \mathcal{R}_3$. If $\boldsymbol{x}^{\boldsymbol{y}}$ satisfies the constraint (2.1c) corresponding to $\bar{\mathcal{R}} = \{r \in \mathcal{R}_3 : x_r^{\boldsymbol{y}} > 0\}$, then $\boldsymbol{x}^{\boldsymbol{y}}$ is a feasible route-based solution. Furthermore, $\boldsymbol{x}^{\boldsymbol{y}}$ is optimal to the CMTVRPTW, since it has the equal objective value with $\boldsymbol{y}$ and $SF(\mathcal{S}_3, \mathcal{T})$ is a relaxation of $SF(\mathcal{S}_3, \mathcal{H})$. Otherwise, a time point associated with a constraint (2.8c) violated by $\boldsymbol{y}$ is added to $\mathcal{T}$, and the constraint (A.1) corresponding to $\bar{\mathcal{R}}$ is incorporated into $SF(\mathcal{S}_3, \mathcal{T})$.

$$\sum_{s \in \mathcal{S}_3 : r_s \in \bar{\mathcal{R}}} y_s \leq \phi(K, \bar{\mathcal{R}}), \quad \forall \bar{\mathcal{R}} \subseteq \mathcal{R}. \tag{A.1}$$

Algorithm 9 Enhancement for Solving Trip-Based IP Formulation

while no optimal solution to the CMTVRPTW has been found **do**
  Solve $SF(\mathcal{S}_3, \mathcal{T})$ to obtain its optimal solution $\boldsymbol{y}$
  if $\boldsymbol{y}$ satisfies all constraints (2.8c) for $t \in \{\tau_s : s \in \mathcal{S}(\boldsymbol{y})\}$ **then**
    $\boldsymbol{y}$ is an optimal trip-based solution to the CMTVRPTW
  else if $\boldsymbol{x^y}$ satisfies the constraint (2.1c) for $\bar{\mathcal{R}} = \{r \in \mathcal{R}_3 : x_r^{\boldsymbol{y}} > 0\}$ **then**
    $\boldsymbol{x^y}$ is an optimal route-based solution to the CMTVRPTW
  else
    Add into $\mathcal{T}$ a time point corresponding to a constraint (2.8c) violated by $\boldsymbol{y}$
    Add into $SF(\mathcal{S}_3, \mathcal{T})$ the constraint (A.1) corresponding to $\bar{\mathcal{R}} = \{r \in \mathcal{R}_3 : x_r^{\boldsymbol{y}} > 0\}$

A.3 Computational Results on Easy Instances

Table A.1 summarizes the computational results for 66 easy instances. In this table, "#Inst" represents the number of instances in each group. "$\chi(LRF)$" and "$\chi(LSF)$" denote the objective values of the route-based and trip-based LP formulations, respectively, both enhanced with valid cuts. In this table, the values for "$|\mathcal{R}_1|$", "$\chi(LRF)$", "CT_{IP}", "CT_{tot}", "$|\mathcal{R}_3|$", "$\Delta_R(\%)$", and "$\chi(LSF)$" are averaged over each group of instances.

Table A.1: Computational results on easy instances

Group	#Inst	$\lvert\mathcal{N}\rvert$	Yang [2025]				Our Algorithm				
			$\lvert\mathcal{R}_1\rvert$	$\chi(LRF)$	CT_{IP}	CT_{tot}	$\lvert\mathcal{R}_3\rvert$	$\Delta_R(\%)$	$\chi(LSF)$	CT_{IP}	CT_{tot}
C2_2	9	140	16577	3350.5	0.03	**0.05**	11084	33.1	3352.3	0.02	0.10
R2_2	9	140	36195	3686.5	0.35	0.42	26937	25.6	3689.0	0.14	**0.26**
RC2_2	9	140	35831	3544.6	0.16	**0.40**	30221	15.7	3545.7	0.13	0.42
C2_2	6	170	9499	3951.2	0.01	**0.04**	4216	55.6	3953.1	0.01	0.10
R2_2	6	170	50330	4193.7	0.39	0.60	46019	8.6	4194.5	0.34	0.60
RC2_2	6	170	66838	4219.5	0.77	1.44	57733	13.6	4219.6	0.47	**1.26**
C2_2	9	200	5801	4536.8	0.00	**0.05**	2704	53.4	4538.3	0.00	0.14
R2_2	6	200	39732	4852.5	0.28	0.58	34118	14.1	4852.9	0.16	**0.56**
RC2_2	6	200	87975	4790.2	0.42	**1.28**	76304	13.3	4790.7	1.03	2.07

Notes. The unit for computing time is hours. All these instances are solved to optimality by both Yang [2025] and our algorithm.

Table A.1 shows that both methods are capable of solving these easy instances to optimality within a few hours, with our algorithm achieving performance comparable to the benchmark method. Specifically, our algorithm requires shorter CT_{IP}

for 6 groups and longer CT_{IP} for only one group, indicating higher efficiency in solving the IP formulation compared to the benchmark. However, our algorithm takes longer CT_{tot} for 5 groups and shorter CT_{tot} for only 3 groups, reflecting slightly lower efficiency in total computing times. This is because these easy instances can be solved very quickly, and our algorithm requires additional time to compute the trip-based LP formulation. Nonetheless, by solving the trip-based LP formulation, our algorithm achieves better LP bounds for 8 groups, with up to 0.14% for certain instances. Furthermore, the average number of routes considered in our algorithm is reduced by 8.6% to 55.6%, demonstrating the effectiveness route elimination.

Appendix B

Supplements for Chapter 4

B.1 More Details on Hierarchical Constructive Heuristic

B.1.1 Efficient Check of Bandwidth Capacity Constraints

When finding a route for a demand in a scenario, both the labeling algorithm and the route reuse algorithm involve checking bandwidth capacity constraints. For this, we first introduce a direct check approach, which is time-consuming, and then present a more efficient partial check approach, which significantly reduces the number of inequalities to check.

In the direct check approach shown in Table B.1, when assigning a route to demand $k \in \mathcal{K}'$ in the working scenario (level-1 scenario, level-2 scenario, respectively), inequalities (B.1a) and (B.1b) ((B.1c) and (B.1d), (B.1e), respectively) need to be checked to determine whether bandwidth capacity constraints are satisfied, where each $C_l^{e_1 e_2}$ is defined in Section 4.3.3. Inequalities (B.1b) are considered for bandwidth reservation constraints, and inequalities (B.1d) are considered due to

consistent routing constraints (i.e., demand k should continue using route $r_k^{e_1 0}$ in scenarios $\bar{\Theta}_k(e_1)$).

$$C_l^{00} + b_k \leq B, \qquad \forall l \in \mathcal{L}(r_k^{00}). \tag{B.1a}$$

$$C_l^{e_1 e_2} + b_k \leq B, \qquad \forall l \in \mathcal{L}(r_k^{00}), (e_1, e_2) \in \Omega^1 \cup \Omega^2. \tag{B.1b}$$

$$C_l^{e_1 0} + b_k(1 - y_{kl}^{00}) \leq B, \qquad \forall l \in \mathcal{L}(r_k^{e_1 0}). \tag{B.1c}$$

$$C_l^{e_1 e_2} + b_k(1 - y_{kl}^{00}) \leq B, \qquad \forall l \in \mathcal{L}(r_k^{e_1 0}), (e_1, e_2) \in \bar{\Theta}_k(e_1). \tag{B.1d}$$

$$C_l^{e_1 e_2} + b_k(1 - y_{kl}^{00}) \leq B, \qquad \forall l \in \mathcal{L}(r_k^{e_1 e_2}). \tag{B.1e}$$

The large cardinalities of $\mathcal{E}$ and $\mathcal{E} \setminus \mathcal{E}(r_k^{e_1 0})$ in large-scale STG2 instances lead to a large number of inequalities (B.1b) and (B.1d) ($|\mathcal{L}(r_k^{00})||\mathcal{E}|^2$ and $|\mathcal{L}(r_k^{e_1 0})||\mathcal{E} \setminus \mathcal{E}(r_k^{e_1 0})|$, respectively).

Proposition 17 established below enables us to use the partial check approach specified in Table B.1, eliminating the need to verify inequalities (B.1b) and (B.1d).

Proposition 17. *When executing lines 3–5 of Algorithm 5, we have*

$$C_l^{e_1 e_2} = C_l^{00}, \forall l \in \mathcal{L}, (e_1, e_2) \in \Omega^1 \cup \Omega^2. \tag{B.2}$$

When executing lines 7–10 of Algorithm 5, we have

$$C_l^{e_1 e_2} \leq C_l^{e_1 0}, \forall l \in \mathcal{L}, (e_1, e_2) \in \Omega^2. \tag{B.3}$$

Proof. Part (i). Algorithm 5 starts route planning at line 3, thus $C_l^{e_1 e_2} = 0, \forall l \in \mathcal{L}, (e_1, e_2) \in \Omega$ before line 3, i.e., relations (B.4) hold before planning the working routes for demands.

$$C_l^{e_1 e_2} = C_l^{00}, \forall l \in \mathcal{L}, (e_1, e_2) \in \Omega^1 \cup \Omega^2. \tag{B.4}$$

Throughout lines 3–5, Algorithm 5 iteratively plans the working route for each demand. When assigning route r_k^{00} to demand $k \in \mathcal{K}'$ in the working scenario, the bandwidth consumption on lightpaths in the working scenario should be updated according to operations (B.5a), and the bandwidth consumption on lightpaths in failure scenarios should be updated according to operations (B.5b) due to bandwidth reservation constraints.

$$C_l^{00} \leftarrow C_l^{00} + b_k, \qquad \forall l \in \mathcal{L}(r_k^{00}), \tag{B.5a}$$

$$C_l^{e_1e_2} \leftarrow C_l^{e_1e_2} + b_k, \qquad \forall l \in \mathcal{L}(r_k^{00}), (e_1, e_2) \in \Omega^1 \cup \Omega^2. \tag{B.5b}$$

Operations (B.5a) and (B.5b) indicate that planning working routes for demands (lines 3–5) does not change the fulfillment of relations (B.4), which, together with the fact that relations (B.4) hold before line 3, implies that relations (B.4) hold throughout lines 3–5.

Part (ii). Operations in line 6 do not change the fulfillment of relations (B.4), which, together with the fact that relations (B.4) hold throughout lines 3–5, implies that relations (B.6) hold before line 7.

$$C_l^{e_1e_2} \leq C_l^{e_10}, \forall l \in \mathcal{L}, (e_1, e_2) \in \Omega^2. \tag{B.6}$$

Throughout lines 7–10, Algorithm 5 iteratively plans level-1 routes for demands. When assigning route $r_k^{e_10}$ to demand $k \in \mathcal{K}'$ in a level-1 scenario $(e_1, 0) \in \Omega^1$, the bandwidth consumption on lightpaths in the level-1 scenario $(e_1, 0)$ should be updated according to operations (B.7a), and the bandwidth consumption on lightpaths in level-2 scenarios $\bar{\Theta}_k(e_1) = \{(e_1, e_2) \in \Omega^2 : e_2 \in \mathcal{E} \setminus \mathcal{E}(r_k^{e_10})\}$ should be updated according to operations (B.7b) due to consistent routing constraints

Table B.1: Inequalities for checking bandwidth capacity constraints

	Direct Check	Partial Check
Assign r_k^{00} in scenario $(0,0)$	(B.1a) and (B.1b)	(B.1a)
Assign $r_k^{e_1 0}$ in scenario $(e_1,0) \in \Phi_k$	(B.1c) and (B.1d)	(B.1c)
Assign $r_k^{e_1 e_2}$ in scenario $(e_1,e_2) \in \Psi_k \cup \Theta_k$	(B.1e)	(B.1e)

which require that demand k should still use route $r_k^{e_1 0}$ in scenarios $\bar{\Theta}_k(e_1)$.

$$C_l^{e_1 0} \leftarrow C_l^{e_1 0} + b_k(1 - y_{kl}^{00}), \qquad \forall l \in \mathcal{L}(r_k^{e_1 0}), \tag{B.7a}$$

$$C_l^{e_1 e_2} \leftarrow C_l^{e_1 e_2} + b_k(1 - y_{kl}^{00}), \qquad \forall l \in \mathcal{L}(r_k^{e_1 0}), (e_1,e_2) \in \bar{\Theta}_k(e_1). \tag{B.7b}$$

Operations (B.7a) and (B.7b) indicate that planning level-1 routes for demands does not change the fulfillment of relations (B.6), which, together with the fact that relations (B.6) hold before line 7, implies that (B.4) hold throughout lines 7–10. □

The correctness of the partial check approach is based on the following reasons. When planning working routes (lines 3–5 of Algorithm 5), satisfying inequalities (B.1a) ensures the fulfillment of inequalities (B.1b), as per relations (B.2). Similarly, when planning level-1 routes (lines 7–10 of Algorithm 5), satisfying inequalities (B.1c) ensures the fulfillment of inequalities (B.1d), according to relations (B.3).

The partial check approach is more efficient than the direct check approach. When assigning the working route r_k^{00} to demand k, the number of inequalities to check reduces from $|\mathcal{L}(r_k^{00})|(1 + |\mathcal{E}|^2)$ to $|\mathcal{L}(r_k^{00})|$, and when assigning the level-1 route $r_k^{e_1 0}$ to demand k, the number of inequalities to check decreases from $|\mathcal{L}(r_k^{e_1 0})|(1 + |\mathcal{E} \setminus \mathcal{E}(r_k^{e_1 0})|)$ to $|\mathcal{L}(r_k^{e_1 0})|$.

In the partial check approach, only bandwidth capacity in the scenario being currently planned needs verification. Therefore, when finding a route for demand k in scenario $(e_1,e_2) \in \Omega$ with the labeling algorithm, to determine whether the

extension of a label L_1 along a logical arc $f_2 \in \mathcal{A}_l$ satisfies bandwidth capacity constraints, we only need to check the inequality $C^{e_1e_2}_{l(f_2)} + b_k \leq B$ for $(e_1, e_2) = (0, 0)$ and the inequality $C^{e_1e_2}_{l(f_2)} + b_k(1 - y^{00}_{kl(f_2)}) \leq B$ for $(e_1, e_2) \in \Omega^1 \cup \Omega^2$.

B.1.2 Efficient Update of Bandwidth Consumption

After assigning a route $r^{e_1e_2}_k$ to a demand $k \in \mathcal{K}'$ in scenario $(e_1, e_2) \in \Omega$, the bandwidth consumption on associated lightpaths $\boldsymbol{C}_l, l \in \mathcal{L}(r^{e_1e_2}_k)$ need to be updated for their usage in the check of bandwidth capacity constraints. For this, we first introduce a direct update approach, which is time-consuming, and then present an incremental update approach, which requires significantly fewer update operations and less memory.

The direct update approach stores the bandwidth consumption $C^{e_1e_2}_l$ for each lightpath $l \in \mathcal{L}'$ and each scenario $(e_1, e_2) \in \Omega$. When assigning a route to demand $k \in \mathcal{K}'$ in a scenario, the bandwidth consumption on lightpaths is updated according to Table B.2, where operations (B.8b) are performed for bandwidth reservation constraints, and operations (B.8d) are performed due to consistent routing constraints.

$$C^{00}_l \leftarrow C^{00}_l + b_k, \qquad \forall l \in \mathcal{L}(r^{00}_k). \tag{B.8a}$$

$$C^{e_1e_2}_l \leftarrow C^{e_1e_2}_l + b_k, \qquad \forall l \in \mathcal{L}(r^{00}_k), (e_1, e_2) \in \Omega^1 \cup \Omega^2. \tag{B.8b}$$

$$C^{e_10}_l \leftarrow C^{e_10}_l + b_k(1 - y^{00}_{kl}), \qquad \forall l \in \mathcal{L}(r^{e_10}_k). \tag{B.8c}$$

$$C^{e_1e_2}_l \leftarrow C^{e_1e_2}_l + b_k(1 - y^{00}_{kl}), \qquad \forall l \in \mathcal{L}(r^{e_10}_k), (e_1, e_2) \in \bar{\Theta}_k(e_1). \tag{B.8d}$$

$$C^{e_1e_2}_l \leftarrow C^{e_1e_2}_l + b_k(1 - y^{00}_{kl}), \qquad \forall l \in \mathcal{L}(r^{e_1e_2}_k). \tag{B.8e}$$

The large cardinalities of $\mathcal{E}$ and $\mathcal{E} \setminus \mathcal{E}(r^{e_10}_k)$ in large-scale STG2 instances can lead to a significant number of operations (B.8b) and (B.8d) ($|\mathcal{L}(r^{00}_k)||\mathcal{E}|^2$

Table B.2: Operations for updating bandwidth consumption

	Direct Update	Incremental Update
Assign r_k^{00} in scenario $(0,0)$	(B.8a) and (B.8b)	(B.8a)
Assign $r_k^{e_1 0}$ in scenario $(e_1,0) \in \Phi_k$	(B.8c) and (B.8d)	(B.9a) and (B.10)
Assign $r_k^{e_1 e_2}$ in scenario $(e_1,e_2) \in \Psi_k \cup \Theta_k$	(B.8e)	(B.8e)

and $|\mathcal{L}(r_k^{e_1 0})||\mathcal{E} \setminus \mathcal{E}(r_k^{e_1 0})|$, respectively). However, in Algorithm 5, bandwidth consumption can be updated with the incremental update approach specified in Table B.2, reducing both memory usage and the number of update operations. Here, $\Delta_l^{e_1 e_2} = C_l^{e_1 e_2} - C_l^{00}, l \in \mathcal{L}', (e_1,e_2) \in \Omega^1 \cup \Omega^2$ represents the incremental bandwidth consumption in scenario (e_1,e_2) compared to the working scenario, and $\delta_l^{e_1 e_2} = \Delta_l^{e_1 0} - \Delta_l^{e_1 e_2}, l \in \mathcal{L}', (e_1,e_2) \in \Omega^2$ indicates the over-reserved bandwidth in scenario (e_1,e_2) compared to the level-1 scenario $(e_1,0)$.

Our incremental update approach works as follows. According to Proposition 17, when assigning route r_k^{00} to demand $k \in \mathcal{K}'$ in the working scenario (lines 3–5 of Algorithm 5), we perform operations (B.8a) and keep Δ as a zero vector, which is equivalent to performing operations (B.8a) and (B.8b).

By the definition of Δ, we know that operations (B.8c) and (B.8d) are equivalent to (B.9a) and (B.9b).

$$\Delta_l^{e_1 0} \leftarrow \Delta_l^{e_1 0} + b_k(1 - y_{kl}^{00}), \qquad \forall l \in \mathcal{L}(r_k^{e_1 0}), \tag{B.9a}$$

$$\Delta_l^{e_1 e_2} \leftarrow \Delta_l^{e_1 e_2} + b_k(1 - y_{kl}^{00}), \qquad \forall l \in \mathcal{L}(r_k^{e_1 0}), (e_1,e_2) \in \bar{\Theta}_k(e_1). \tag{B.9b}$$

Additionally, by the definition of δ, performing operations (B.9a) and (B.9b) is equivalent to performing (B.9a) and (B.10).

$$\delta_l^{e_1 e_2} \leftarrow \delta_l^{e_1 e_2} + b_k(1 - y_{kl}^{00}), \qquad \forall l \in \mathcal{L}(r_k^{e_1 0}), (e_1,e_2) \in \Theta_k(e_1). \tag{B.10}$$

The number of operations (B.10), i.e., $|\mathcal{L}(r_k^{e_1 0})||\Theta_k(e_1)| = |\mathcal{L}(r_k^{e_1 0})||\mathcal{E}(r_k^{e_1 0})|$, can be significantly smaller than that of operations (B.9b) (i.e., $|\mathcal{L}(r_k^{e_1 0})||\mathcal{E}\setminus\mathcal{E}(r_k^{e_1 0})|$) in large-scale STG2 instances, since we often have $|\mathcal{E}| \gg |\mathcal{E}(r_k^{e_1 0})|$ in these instances. Therefore, when assigning route $r_k^{e_1 0}$ to demand $k \in \mathcal{K}'$ in a level-1 scenario $(e_1, 0) \in \Phi_k$ (lines 7–10 of Algorithm 5), the query of bandwidth consumption $C_l^{e_1 0}$ is given by $C_l^{00} + \Delta_l^{e_1 0}$ for $l \in \mathcal{L}'$, while bandwidth consumption update is performed by operations (B.9a) and (B.10).

At the beginning of planning a level-2 scenario $(e_1, e_2) \in \Omega^2$, we initialize bandwidth consumption $C_l^{e_1 e_2}$ for $l \in \mathcal{L}'$ with operation (B.11). When assigning route $r_k^{e_1 e_2}$ to demand $k \in \mathcal{K}'$ in scenario $(e_1, e_2) \in \Psi_k \cup \Theta_k$, the bandwidth consumption is updated according to operations (B.8e). In Algorithm 5, level-2 scenarios are planned sequentially (lines 12–14), and planning a level-2 scenario imposes no constraint on another level-2 scenario. Therefore, at the end of planning scenario $(e_1, e_2) \in \Omega^2$, we release the memory for storing bandwidth consumption $C_l^{e_1 e_2}, \forall l \in \mathcal{L}'$.

$$C_l^{e_1 e_2} \leftarrow C_l^{00} + \Delta_l^{e_1 0} - \delta_l^{e_1 e_2}. \tag{B.11}$$

Compared with the direct update approach, the incremental update approach requires fewer operations. Specifically, when assigning the working route r_k^{00} to demand k, the number of update operations decreases from $|\mathcal{L}(r_k^{00})|(1 + |\mathcal{E}|^2)$ to $|\mathcal{L}(r_k^{00})|$, and when assigning the level-1 route $r_k^{e_1 0}$ to demand k, the number of update operations decreases from $|\mathcal{L}(r_k^{e_1 0})|(1+|\mathcal{E}\setminus\mathcal{E}(r_k^{e_1 0})|)$ to $|\mathcal{L}(r_k^{e_1 0})|(1+|\mathcal{E}(r_k^{e_1 0})|)$, where $|\mathcal{E} \setminus \mathcal{E}(r_k^{e_1 0})|$ is often much larger than $|\mathcal{E}(r_k^{e_1 0})|$ in large-scale STG2 instances.

Furthermore, for each lightpath $l \in \mathcal{L}'$, the incremental update approach only maintains C_l^{00}, $\Delta_l^{e_1 0}, \forall (e_1, 0) \in \bigcup_{k\in\mathcal{K}'} \Phi_k$, and $\delta_l^{e_1 e_2}, \forall (e_1, e_2) \in \bigcup_{k\in\mathcal{K}'} \Theta_k$, along with an additional scalar $C_l^{e_1 e_2}$ when (e_1, e_2) is the level-2 scenario being currently planned. These data are often much sparser than that maintained in the direct

update approach (i.e., $C_l^{e_1e_2}, \forall(e_1, e_2) \in \Omega$), indicating that the incremental update approach also requires less memory.

B.2 Details about Enhancement by Parallel Computing

B.2.1 Planning with Slave Threads

Multiple slave threads are used to plan level-2 scenarios in parallel without generating new lightpaths, while within a single thread scenarios are planned sequentially. The t-th thread executes the procedure shown in Algorithm 10 to plan level-2 scenarios Γ_t. The successfully planned scenarios are denoted as $\bar{\Gamma}_t$, while $\tilde{\Gamma}_t$ represents the set of blocked scenarios (failed to be planned due to insufficient lightpaths). Blocked scenarios $\tilde{\Gamma}_t$ are then planned in the master thread, where new lightpaths can be established.

Algorithm 10 proceeds as follows. The values to be returned are initialized in line 1. In line 2, the indicator to stop the thread $flag_{stop}$ and the number of routes generated in this thread n_R are initialized. At the beginning of planning a level-2 scenario (e_1, e_2) in line 5, bandwidth consumption on lightpaths $C_l^{e_1e_2}, l \in \mathcal{L}'$ is initialized with (B.11). For each unprotected demand in scenario (e_1, e_2), we attempt to find the level-2 route by evaluating generated routes (line 7) or using the labeling algorithm without establishing new lightpaths (line 9). If both are failed, the scenario should be planned in the master thread, and we record the bandwidth consumption on lightpaths in this scenario (line 18).

Stopping criteria are evaluated in lines 21, 25, and 27. The indicator of whether there is an idle slave thread $flag_{idle}$, the total number of scenarios that failed to be planned in all slave threads n_{fail}, the total number of routes newly

generated in all slave threads n_{route}, are modifiable data shared by all slave threads. Hyperparameters NF and NR represent thresholds on n_{fail} and n_{route}, respectively. The result of evaluating stopping criteria is *true* if and only if at least one of the following conditions is satisfied: (i) $flag_{idle} = true$; (ii) $n_{fail} \geq NF$; (iii) $n_{route} \geq NR$.

The motivation for stopping criterion (i) is to reduce thread idleness: When a thread has already stopped, stopping all slave threads earlier allows an earlier start of a new "master-slave" iteration of Algorithm 8 and an earlier reactivation of stopped threads. Stopping criteria (ii) and (iii) aim to control the number of "master-slave" iterations. Smaller NF and NR lead to more iterations, increasing the overhead of starting and stopping threads. Conversely, a larger NF results in fewer iterations, but tends to increase the number of scenarios being planned by the master thread, reducing parallelization. A larger NR results in less frequent synchronization of the route pool $\mathcal{R}'$ and more frequent calls to the labeling algorithm.

Lines 19, 22, and 26 determine when to evaluate stopping criteria. The hyperparameters CS and CR control this frequency. A higher frequency can lead to race conditions, i.e., multiple threads access and modify shared data $(flag_{idle}, n_{fail}, n_{route})$ concurrently. Whereas a lower frequency can cause idleness, as already stopped slave threads remain idle until all slave threads stop.

B.2.2 Planning with Master Thread

By applying the procedure shown in Algorithm 11, the master thread plans level-2 scenarios that slave threads cannot complete. The main differences between planning with the master thread and slave threads are: (i) The bandwidth consumption $C_l^{e_1 e_2}$ on lightpath $l \in \mathcal{L}'$ in scenario (e_1, e_2) is initialized with $\tilde{C}_l^{e_1 e_2}$ (line 2) instead of using (B.11); (ii) New lightpaths can be established when finding routes with

Algorithm 10 Slave Plan

Input: Γ_t, hyperparameters NF, NR, CS, CR, and modifiable data shared by all slave threads $flag_{idle}, n_{fail}, n_{route}$
Output: $\bar{\Gamma}_t, \tilde{\Gamma}_t, \tilde{\boldsymbol{C}}(t)$

$\bar{\Gamma}_t \leftarrow \emptyset, \tilde{\Gamma}_t \leftarrow \emptyset, \tilde{\boldsymbol{C}}(t) \leftarrow \emptyset, \mathcal{R}'(t) \leftarrow \emptyset$
$flag_{stop} \leftarrow false, n_R \leftarrow 0$
for all $(e_1, e_2) \in \Gamma_t$ **and** $flag_{stop} = false$ **do**
    $flag_{fail} \leftarrow false$ (Whether fails to plan some demand in the scenario)
    Initialize bandwidth consumption on lightpaths $C_l^{e_1e_2}, l \in \mathcal{L}'$ with (B.11)
    for all $k \in \mathcal{K}'$ **and** demand k is not protected in scenario (e_1, e_2) **do**
        if found a generated route $r \in \mathcal{R}'$ for k in (e_1, e_2) with Algorithm 7 **then**
            $r_k^{e_1e_2} \leftarrow r$; Update bandwidth consumption with (B.8e)
        else if found a new route r for k in (e_1, e_2) with Algorithm 6 **then**
            $r_k^{e_1e_2} \leftarrow r$; Update bandwidth consumption with (B.8e)
            $\mathcal{R}'(t) \leftarrow \mathcal{R}'(t) \cup \{r\}$
        else
            $flag_{fail} \leftarrow true$
    if $flag_{fail} = false$ **then**
        $\bar{\Gamma}_t \leftarrow \bar{\Gamma}_t \cup \{(e_1, e_2)\}$ (All demands are planned in the scenario)
    else
        $\tilde{\Gamma}_t \leftarrow \tilde{\Gamma}_t \cup \{(e_1, e_2)\}$ (The scenario is blocked)
        $\tilde{\boldsymbol{C}}(t) \leftarrow \tilde{\boldsymbol{C}}(t) \cup \{C_l^{e_1e_2} : l \in \mathcal{L}'\}$ (Store bandwidth consumption)
    if $flag_{fail} = true$ **then**
        $n_{fail} \leftarrow n_{fail} + 1$
        $flag_{stop} \leftarrow$ StopCriteria$(NF, NR, flag_{idle}, n_{fail}, n_{route})$
    else if $|\mathcal{R}'(t)| - n_R \geq CR$ **then**
        $n_{route} \leftarrow n_{route} + |\mathcal{R}'(t)| - n_R$
        $n_R \leftarrow |\mathcal{R}'(t)|$
        $flag_{stop} \leftarrow$ StopCriteria$(NF, NR, flag_{idle}, n_{fail}, n_{route})$
    else if $(|\bar{\Gamma}_t| + |\tilde{\Gamma}_t|)\%CS = 0$ **then**
        $flag_{stop} \leftarrow$ StopCriteria$(NF, NR, flag_{idle}, n_{fail}, n_{route})$
$flag_{idle} \leftarrow true$
return $\bar{\Gamma}_t, \tilde{\Gamma}_t, \tilde{\boldsymbol{C}}(t)$

the labeling algorithm (line 6); (iii) If the master thread cannot find a level-2 route to protect a demand in a scenario, the hierarchical constructive heuristic fails to find a feasible solution to the STG2 (line 10).

Algorithm 11 Master Plan

Input: blocked scenarios $\tilde{\Gamma}$, bandwidth consumption in blocked scenarios $\tilde{\boldsymbol{C}}$
Output: Routes $r_k^{e_1e_2}, k \in \mathcal{K}, (e_1, e_2) \in \tilde{\Gamma}$

1: **for all** $(e_1, e_2) \in \tilde{\Gamma}$ **do**
2: Initialize bandwidth consumption on lightpaths $C_l^{e_1e_2} = \tilde{C}_l^{e_1e_2}, l \in \mathcal{L}'$
3: **for all** $k \in \mathcal{K}'$ **and** demand k is not protected in scenario (e_1, e_2) **do**
4: **if** found a generated route $r \in \mathcal{R}'$ for k in (e_1, e_2) with Algorithm 7 **then**
5: $r_k^{e_1e_2} \leftarrow r$; Update bandwidth consumption with (B.8e)
6: **else if** found a new route r for k in (e_1, e_2) with Algorithm 6 **then**
7: $r_k^{e_1e_2} \leftarrow r$; Update bandwidth consumption with (B.8e)
8: Insert new lightpaths into $\mathcal{L}'$, if any.
9: **else**
10: Fails to find a feasible solution; Terminate.

B.3 Illustration of Optimal Solution for Example in Figure 4.2

Table B.3 presents the optimal solution for the example depicted in Figure 4.2. This example consists of 5 nodes, 8 links, and 2 demands: k_1 = (A, C, 50 Gbps) and k_2 = (A, E, 50 Gbps). In Table B.3, $(0, 0)$ represents the working scenario, $(e_1, 0)$ represents the scenario where link e_1 fails, and (e_1, e_2) represents the scenario where links e_1 and e_2 fail successively. Additionally, routes r_1 through r_7 are defined in Figure 4.2. Notably, the table demonstrates that even for a relatively small network with 8 links, a total of 65 scenarios must be accounted for. This highlights the significant challenge posed by the quadratic growth in the number of scenarios as the size of STG2 instances increases.

Table B.3: Optimal routing for the example instance in Figure 4.2

Scenario	Route for k_1	Route for k_2	Scenario	Route for k_1	Route for k_2
(0, 0)	r_1	r_2	(BE, 0)	r_1	r_2
(AB, 0)	r_1	r_2	(BE, AB)	r_1	r_2
(AB, AC)	r_3	r_4	(BE, AC)	r_3	r_4
(AB, AD)	r_1	r_2	(BE, AD)	r_1	r_2
(AB, BC)	r_1	r_2	(BE, BC)	r_1	r_2
(AB, BE)	r_1	r_2	(BE, CD)	r_1	r_2
(AB, CD)	r_1	r_2	(BE, CE)	r_1	r_7
(AB, CE)	r_1	r_7	(BE, DE)	r_1	r_2
(AB, DE)	r_1	r_2	(CD, 0)	r_1	r_2
(AC, 0)	r_3	r_4	(CD, AB)	r_1	r_2
(AC, AB)	r_3	r_4	(CD, AC)	r_6	r_5
(AC, AD)	r_6	r_5	(CD, AD)	r_1	r_2
(AC, BC)	r_3	r_4	(CD, BC)	r_1	r_2
(AC, BE)	r_3	r_4	(CD, BE)	r_1	r_2
(AC, CD)	r_6	r_5	(CD, CE)	r_1	r_5
(AC, CE)	r_3	r_5	(CD, DE)	r_1	r_2
(AC, DE)	r_3	r_4	(CE, 0)	r_1	r_5
(AD, 0)	r_1	r_2	(CE, AB)	r_1	r_7
(AD, AB)	r_1	r_2	(CE, AC)	r_3	r_5
(AD, AC)	r_6	r_5	(CE, AD)	r_1	r_5
(AD, BC)	r_1	r_2	(CE, BC)	r_1	r_5
(AD, BE)	r_1	r_2	(CE, BE)	r_1	r_7
(AD, CD)	r_1	r_2	(CE, CD)	r_1	r_5
(AD, CE)	r_1	r_5	(CE, DE)	r_1	r_5
(AD, DE)	r_1	r_2	(DE, 0)	r_1	r_2
(BC, 0)	r_1	r_2	(DE, AB)	r_1	r_2
(BC, AB)	r_1	r_2	(DE, AC)	r_3	r_4
(BC, AC)	r_3	r_4	(DE, AD)	r_1	r_2
(BC, AD)	r_1	r_2	(DE, BC)	r_1	r_2
(BC, BE)	r_1	r_2	(DE, BE)	r_1	r_2
(BC, CD)	r_1	r_2	(DE, CD)	r_1	r_2
(BC, CE)	r_1	r_5	(DE, CE)	r_1	r_5
(BC, DE)	r_1	r_2			

References

Chadi Assi, Wei Huo, and Abdallah Shami. Multiple link failures survivability of optical networks with traffic grooming capability. *Computer Communications*, 29(18):3900–3912, 2006.

Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 178(3):755–766, 2007.

Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763, 2010.

Abderraouf Bahri and Steven Chamberland. A global approach for designing reliable WDM networks and grooming the traffic. *Computers & Operations Research*, 35(12):3822–3833, 2008.

Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

Tolga Bektas and Osman Oğuz. On separating cover inequalities for the multidimensional knapsack problem. *Computers & Operations Research*, 34(6):1771–1776, 2007.

Hamza Ben Ticha, Nabil Absi, Dominique Feillet, and Alain Quilliot. Empirical analysis for the VRPTW with a multigraph representation for the road network. *Computers & Operations Research*, 88:103–116, 2017.

Nicola Bianchessi, Timo Gschwind, and Stefan Irnich. Resource-window reduction by reduced costs in path-based formulations for routing and scheduling problems. *INFORMS Journal on Computing*, 36(1):224–244, 2024.

Víctor Blanco, Elena Fernández, and Yolanda Hinojosa. Hub location with protection under interhub link failures. *INFORMS Journal on Computing*, 35(5): 966–985, 2023.

Christina Büsing, Markus Leitner, and Sophia Wrede. Cover-based inequalities for the single-source capacitated facility location problem with customer preferences. *Computers & Operations Research*, 182:107082, 2025.

François Cartier, Brunilde Sansò, and André Girard. An efficient heuristic to dimension large-scale hybrid optoelectronic networks. *Computers & Operations Research*, 33(6):1741–1759, 2006.

Diego Cattaruzza, Nabil Absi, and Dominique Feillet. The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, 50(2): 676–693, 2016.

Diego Cattaruzza, Nabil Absi, and Dominique Feillet. Vehicle routing problems with multiple trips. *Annals of Operations Research*, 271(1):127–159, 2018.

Bensong Chen, George N Rouskas, and Rudra Dutta. Clustering methods for hierarchical traffic grooming in large-scale mesh WDM networks. *Journal of Optical Communications and Networking*, 2(8):502–514, 2010.

Chun Cheng, Yossiri Adulyasak, and Louis-Martin Rousseau. Drone routing with energy function: Formulation and exact algorithm. *Transportation Research Part B: Methodological*, 139:364–387, 2020.

Cisco. Cisco annual reports (2024). Technical report, 2024. URL `https://www.cisco.com/c/en/us/about/annual-reports.html`.

Geoff Clarke and John W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

Teodor Gabriel Crainic, Bernard Gendron, and Mohammad Rahim Akhavan Kazemzadeh. A taxonomy of multilayer network design and a survey of transportation and telecommunication applications. *European Journal of Operational Research*, 303(1):1–13, 2022.

Kevin Dalmeijer and Remy Spliet. A branch-and-cut algorithm for the time window assignment vehicle routing problem. *Computers & Operations Research*, 89: 140–152, 2018.

George B Dantzig and John H Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

Maryam Daryalal and Merve Bodur. Stochastic RWA and lightpath rerouting in WDM networks. *INFORMS Journal on Computing*, 34(5):2700–2719, 2022.

Milind Dawande, Rakesh Gupta, Sanjeewa Naranpanawe, and Chelliah Sriskandarajah. A traffic-grooming algorithm for wavelength-routed optical networks. *INFORMS Journal on Computing*, 19(4):565–574, 2007.

Guy Desaulniers, Timo Gschwind, and Stefan Irnich. Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models. *Transportation Science*, 54(5):1170–1188, 2020.

Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

Taiming Feng, Long Long, Ahmed E Kamal, and Lu Ruan. Two-link failure protection in WDM mesh networks with p-cycles. *Computer Networks*, 54(17): 3068–3080, 2010.

Hermann Gehring and Jörg Homberger. A parallel two-phase metaheuristic for routing problems with time-windows. *Asia Pacific Journal of Operational Research*, 18(1):35–48, 2001.

Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

Ahmed Hadjar, Odile Marcotte, and François Soumis. A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. *Operations Research*, 54(1): 130–149, 2006.

Florent Hernandez, Dominique Feillet, Rodolphe Giroudeau, and Olivier Naud. A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4OR*, 12(3):235–259, 2014.

Florent Hernandez, Dominique Feillet, Rodolphe Giroudeau, and Olivier Naud. Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, 249(2): 551–559, 2016.

John H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

Nan Huang, Hu Qin, Yuquan Du, and Li Wang. An exact algorithm for the multi-trip vehicle routing problem with time windows and multi-skilled manpower. *European Journal of Operational Research*, 319(1):31–49, 2024a.

Nan Huang, Hu Qin, Gangyan Xu, and Fang Wan. An enhanced exact algorithm for the multi-trip vehicle routing problem with time windows and capacitated unloading station. *Computers & Operations Research*, 168:106688, 2024b.

Stefan Irnich, Guy Desaulniers, Jacques Desrosiers, and Ahmed Hadjar. Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, 22(2):297–313, 2010.

Arunita Jaekel, Ataul Bari, and Subir Bandyopadhyay. Resilient traffic grooming for WDM networks. *Journal of Optical Networking*, 7(5):378–387, 2008.

Arunita Jaekel, Ataul Bari, Quazi Rahman, Ying Chen, Subir Bandyopadhyay, and Yash Aneja. Resource efficient network design and traffic grooming strategy with guaranteed survivability. *Optical Switching and Networking*, 9(4):271–285, 2012.

Aura Maria Jalal, Eli Angela Vitor Toso, and Reinaldo Morabito. Integrated approaches for logistics network planning: a systematic literature review. *International Journal of Production Research*, 60(18):5697–5725, 2022.

Brigitte Jaumard and Maryam Daryalal. Efficient spectrum utilization in large scale RWA problems. *IEEE/ACM Transactions on Networking*, 25(2):1263–1278, 2017.

Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008.

Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Grigorios D Konstantakopoulos, Sotiris P Gayialis, and Evripidis P Kechagias. Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational research*, 22(3):2033–2062, 2022.

Xiangyong Li and Y.P. Aneja. A branch-and-benders-cut approach for the fault tolerant regenerator location problem. *Computers & Operations Research*, 115: 104847, 2020.

Leonardo Lozano, Daniel Duque, and Andrés L Medaglia. An exact algorithm for the elementary shortest path problem with resource constraints. *Transportation Science*, 50(1):348–357, 2016.

Jens Lysgaard, Adam N Letchford, and Richard W Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

Zhipeng Lü, Yuan Fang, Zhouxing Su, Yang Wang, Xinyun Wu, and Fred Glover. Dual-neighborhood iterated local search for routing and wavelength assignment. *Computers & Operations Research*, 160:106396, 2023.

Alexandre X. Martins, Christophe Duhamel, Philippe Mahey, Rodney R. Saldanha, and Mauricio C. de Souza. Variable neighborhood descent with iterated local search for routing and wavelength assignment. *Computers & Operations Research*, 39(9):2133–2141, 2012.

Aristide Mingozzi, Roberto Roberti, and Paolo Toth. An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing*, 25(2): 193–207, 2013.

Jianwei Niu, Junyan Liu, Fan Zhang, Fabo Sun, Kerong Yan, Junqi Ma, Jie Sun, and Tieyong Zeng. Survivable traffic grooming with practical constraints in large-scale optical network. In *Proceedings of the 11th International Network Optimization Conference*, pages 10–15, 2024.

Rosario Paradiso, Roberto Roberti, Demetrio Laganá, and Wout Dullaert. An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research*, 68(1):180–198, 2020.

Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100, 2017.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183(1):483–523, 2020.

Marcos Roboredo, Ruslan Sadykov, and Eduardo Uchoa. Solving vehicle routing problems with intermediate stops using VRPSolver models. *Networks*, 81(3): 399–416, 2023.

Stefan Ropke and David Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006.

Munise Kübra Şahin and Hande Yaman. A branch and price algorithm for the heterogeneous fleet multi-depot multi-trip vehicle routing problem with time windows. *Transportation Science*, 2022.

Pruk Sasithong, Le Quang Quynh, Poompat Saengudomlert, Pisit Vanichchanunt, Nguyen Hoang Hai, and Lunchakorn Wuttisittikulkij. Maximizing double-link

failure recovery of over-dimensioned optical mesh networks. *Optical Switching and Networking*, 36:100541, 2020.

Guido Schryen. Parallel computational optimization in operations research: A new integrative framework, literature review and research directions. *European Journal of Operational Research*, 287(1):1–18, 2020.

Samir Sebbah and Brigitte Jaumard. An efficient column generation design method of p-cycle-based protected working capacity envelope. *Photonic Network Communications*, 24:167–176, 2012.

Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

Arun K Somani. Survivable traffic grooming. In *Traffic Grooming for Optical Networks: Foundations, Techniques, and Frontiers*, pages 137–157. 2008.

Benoit Vignac, François Vanderbeck, and Brigitte Jaumard. Reformulation and decomposition approaches for traffic routing in optical networks. *Networks*, 67 (4):277–298, 2016.

Yong Wang and Qian-Ping Gu. On the complexity and algorithm of grooming regular traffic in WDM optical networks. *Journal of Parallel and Distributed Computing*, 68(6):877–886, 2008.

Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

Xinyun Wu, Tao Ye, Qi Guo, and Zhipeng Lü. GRASP for traffic grooming and routing with simple path constraints in WDM mesh networks. *Computer Networks*, 86:27–39, 2015.

Xinyun Wu, Zhipeng Lü, and Fred Glover. A matheuristic for a telecommunication network design problem with traffic grooming. *Omega*, 90:102003, 2020.

Fei Yang. Design and simulated implementation of double-link failures protection algorithms in traffic grooming WDM mesh networks. Master's thesis, Northeastern University, 2009.

Yu Yang. An exact price-cut-and-enumerate method for the capacitated multitrip vehicle routing problem with time windows. *Transportation Science*, 57(1): 230–251, 2023.

Yu Yang. Deluxing: Deep lagrangian underestimate fixing for column-generation-based exact methods. *Operations Research*, 73(3):1184–1207, 2025.

Wang Yao and B Ramamurthy. Survivable traffic grooming with path protection at the connection level in WDM mesh networks. *Journal of Lightwave Technology*, 23(10):2846–2853, 2005a.

Wang Yao and Byrav Ramamurthy. A link bundled auxiliary graph model for constrained dynamic traffic grooming in WDM mesh networks. *IEEE Journal on Selected Areas in Communications*, 23(8):1542–1555, 2005b.

Wang Yao and Byrav Ramamurthy. Survivable traffic grooming in WDM mesh networks under SRLG constraints. In *IEEE International Conference on Communications*, volume 3, pages 1751–1755, 2005c.

Barış Yıldız and Oya Ekin Karaşan. Regenerator location problem in flexible optical networks. *Operations Research*, 65(3):595–620, 2017.

Barış Yıldız, Oya Ekin Karaşan, and Hande Yaman. Branch-and-price approaches for the network design problem with relays. *Computers & Operations Research*, 92:155–169, 2018.

Xiaojin Zheng, Meixia Yin, and Yanxia Zhang. Integrated optimization of location,

inventory and routing in supply chain network design. *Transportation Research Part B: Methodological*, 121:1–20, 2019.

Hongyue Zhu, Hui Zang, Keyao Zhu, and Biswanath Mukherjee. A novel generic graph model for traffic grooming in heterogeneous WDM mesh networks. *IEEE/ACM Transactions On Networking*, 11(2):285–299, 2003.

Keyao Zhu and Biswanath Mukherjee. Traffic grooming in an optical WDM mesh network. *IEEE Journal on Selected Areas in Communications*, 20(1):122–133, 2002.