The Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes

Alberto Guastalla

October 14, 2025

UNIVERSITÀ
DI TORINO

Doctoral School — Natural Sciences area
PhD in Computer Science

PhD thesis

The Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes

Alberto Guastalla

Supervisors Roberto Aringhieri
Pierre Hosteins
University of Torino

Reviewers Claudia Archetti
University of Brescia

Leandro Coelho
Université Laval

October 14, 2025

Alberto Guastalla
The Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes
PhD thesis, October 14, 2025
Reviewers: Claudia Archetti and Leandro Coelho
Committee: Andrea Cesare Grosso, Claudia Archetti and Maddalena Nonato
Supervisors: Roberto Aringhieri and Pierre Hosteins

Università degli Studi di Torino
PhD in Computer Science
Doctoral School — Natural Sciences area
Department of Computer Science
Corso Svizzera, 185
10149 – Torino (Italy)

Contents

1 Introduction

Routing problems in operations research have the aim of finding the most efficient paths or schedules for routing resources, such as vehicles, goods or information, through a network. These problems aim to minimize operational costs in order to meet various constraints like capacity limits, time windows or specific operational requirements. Routing problems are inherently difficult and are typically classified as NP-hard which essentially means that searching an optimal solution for large-scale instances is computationally challenging. As a result, a range of solution methods has been developed that include exact algorithms, heuristics, and metaheuristics to balance solution quality and computational efficiency. These problems have wide applicability across industries including transportation, logistics, telecommunications and service management.

This chapter begins by exploring the evolution of the vehicle routing problems, a cornerstone in logistics and operations research that underpins many practical applications. After this general discussion, the chapter introduces two specific real-world applications: the Daily Swab Test Collection Problem (DSTCP) and the Ambulance Routing Problem (ARP) for post-disaster management. The DSTCP focuses on the optimisation of the collection of the diagnostic swab tests identified through digital contact tracing. The ARP addresses the routing of ambulances in order to transport the injured patients to hospitals during emergencies in order to balance critical factors like patient prioritisation and limited resources. Considering these applications as a starting point, the chapter introduces a new variant of the Team Orienteering Problem (TOP), named the Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (TOP-ST-MIN). This variant combines key elements from both the DSTCP and ARP.

1.1 Vehicle routing problems

The Vehicle Routing Problem (VRP) is a well-known optimisation problem with the aim to find a set of routes for a fleet of vehicles to visit a specific set of locations. A route represents an ordered sequence of locations visited by a single vehicle, starting from and ending at a depot. Essentially, the objective consists in minimising the total travel cost with the unique constraint that each location must be visited exactly one time. The VRP has a wide range of practical applications in areas like logistics, transportation, delivery services and public transit. A recent and complete survey for the VRPs is reported in Golden et al. [46].

Numerous VRP variants have been introduced to address specific logistical needs, in order to make the VRP family both versatile and diverse. The Traveling Salesman Problem introduced by Dantzig et al. [29] is a special case of the VRP that consider a single vehicle. Its objective consists in determining the shortest possible route that visits a given set of cities exactly once and returns to the starting point. Figure 1.1 illustrates an example of the problem, where node 1 denotes the depot and the remaining nodes represent the customers to be visited.

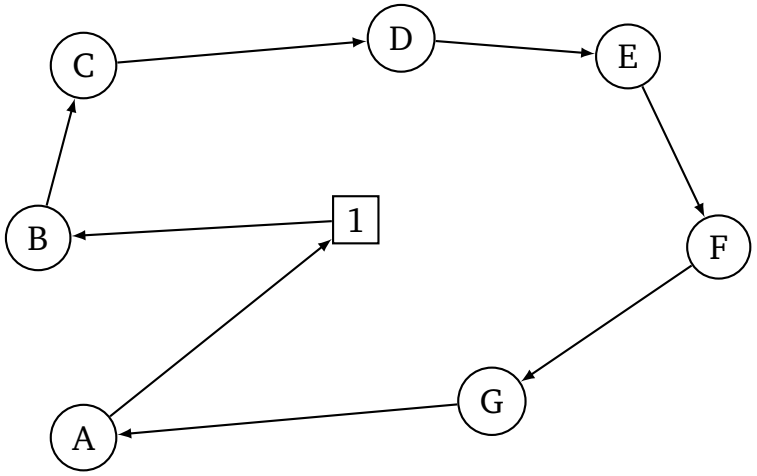


Fig. 1.1: A graphical representation of a feasible solution for the TSP. The square node indicates the depot (node 1), the circular nodes represent the customers to be served and each arrow is associated with a travel distance or time.

Another notable variant is the Capacitated Vehicle Routing Problem presented by Dantzig and Ramser [30] which considers vehicle capacity constraints ensuring balanced load distribution. The Vehicle Routing Problem with Time Windows proposed in Solomon [108] introduces temporal constraints the require that goods can be delivered only within specified time slots. This variant is particularly valuable for scenarios like e-commerce in which the customers time commitments are critical. Lastly, the Vehicle Routing Problem with Pickup and Delivery presented in Savelsbergh and Sol [104] adds complexity by incorporating paired pickup and drop-off locations. This problem is quite relevant in applications like ride-sharing and parcel collection networks. In this problem, goods can either be delivered or collected.

Recent VRP studies have also addressed stochastic and dynamic environments increasing the problem's applicability in uncertain and real-time scenarios. The Stochastic VRPs address uncertainty in parameters like customer demand, travel time and service time by modelling them as random variables. The approaches to solve these problems often rely on stochastic programming and decision Markov process techniques to find solutions that perform well under uncertainty. Dynamic VRPs, by contrast, model environments where customer demands, traffic conditions

or fleet availability can change in real time as commonly seen in emergency response or on-demand delivery services. These problems are typically solved with online algorithms or real-time re-optimisation techniques that allow for adjustments during route execution in order to enable responsive logistics management.

To tackle the VRP's inherent computational complexity, a broad array of solution approaches has emerged in the literature. Exact algorithms such as branch-and-bound, branch-and-cut and branch-and-price provide optimal solutions but are limited in scalability and only applicable to small or medium sized benchmark sets. For large-scale VRPs, metaheuristic methods such as genetic algorithms simulated annealing tabu search and ant colony optimisation are widely used due to their ability to produce high-quality solutions within reasonable running time. These metaheuristics are particularly effective when combined with problem-specific local search techniques to explore different parts of the solutions space. A more recent development in VRP research consists in the matheuristics that are algorithms that combine mathematical programming and metaheuristics. Matheuristic methods permit to balance the solution quality and the computational efficiency. VRPs are often adopted to numerous sectors, including e-commerce, postal and parcel delivery as practical applications.

Overall, the VRP and its variants constitute a rich field of study continuously evolving to incorporate new logistical challenges and technological advancements.

1.2 Vehicle routing problems with profits

Vehicle Routing Problems with Profits (VRPPs) extend the traditional VRP framework by integrating the concept of profit maximisation. This makes these models particularly interesting for applications in which visiting all customers is not required. Unlike classical VRPs, where the goal is typically to minimize costs across a fixed set of nodes, VRPPs are characterised by selective visits in order to allow the decision-maker to choose a subset of customers in order to maximise the overall profit. The aim is not only to identify routes that maximize the overall collected profit but also to take into account the cost of them.

Various VRPP variants have been proposed in the literature, each designed to handle specific practical problems. The Prize-Collecting Traveling Salesman Problem introduced in Balas [14] is a variant of the Travelling Salesman Problem that has the same objective but introduces an additional constraint that imposes the minimum profit collected must be greater or equals to a specific threshold. The Profitable Tour Problem presented by Dell'Amico et al. [31] has the objective to maximise the profit collected by visiting a subset of cities minimising the total travel time. In this problem, each customer has a profit and the objective is to determine which customers to serve in order to maximise the overall profit collected.

The Orienteering Problem (OP) introduced by Golden et al. [45] is one of the most widely studied VRPPs where the objective is focused on finding a single route that maximises the total profit without exceeding a specific time budget constraint. This problem is particularly relevant in applications like planning tourism itineraries [103]. Figure 1.2 depicts an example of the OP. In this case, a single route begins at depot 1, visits a subset of customers collecting the profits indicated inside the grey boxes and then returns to the depot. As evident from the example, not all customers have been visited.

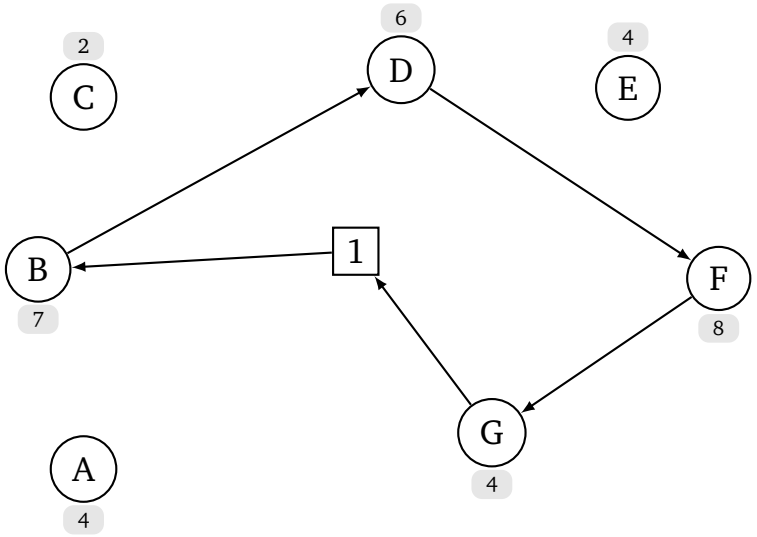


Fig. 1.2: A graphical representation of a feasible solution for the OP. The quantity shown inside the gray boxes represents the profit of the customers.

VRPPs are increasingly relevant in several applications about traditional logistics and transportation. For instance, in waste collection with selective pickups, the goal is to maximize service coverage or profit by strategically selecting neighbourhoods or regions based on need and available resources. In tourism, the VRPP models are applied to develop optimal multi-day itineraries where travellers can maximize the number of attractions based on personal preferences respecting the time budget. The rise of smart cities and urban mobility solutions has further expanded VRPP applications with VRPP models supporting the optimisation of last-mile delivery and autonomous vehicle operations in complex urban environments.

In conclusion, VRPPs provide a flexible and profit-oriented extension to classical VRPs offering robust tools in order to optimise service profitability. Incorporating selective visits constraints, VRPPs provide tailored solutions to different logistical challenges. Beyond theoretical advancements in combinatorial optimisation, VRPPs have substantial practical applications such as in logistics [92] and tourism [54].

The continuous evolution of VRPPs supported by advancements in computational methods ensures their applicability to increasingly complex problems.

1.3 Orienteering problems

The OPs represent a class of routing problems in which the goal is to design routes that maximize the total profit collected from a subset of nodes each associated with a specific profit. The route's cost must respect constraints like vehicle capacity and route duration. Originating from the classical OP, where a single vehicle navigates between a start and end point to accumulate the highest profit within a limited time, OPs are versatile and widely applied across fields where selective visits and profit maximisation are key objectives such as tourism route planning, mobile workforce management and emergency response logistics.

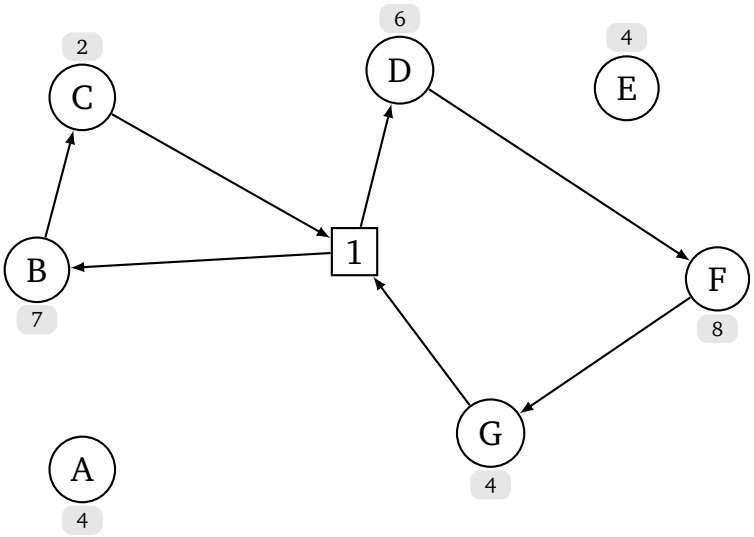


Fig. 1.3: A graphical representation of a feasible solution for the TOP. In this example two vehicles depart from the depot.

A significant extension of the OP, the Team Orienteering Problem (TOP) introduced in Butt and Cavalier [21], adapts the single-vehicle OP to a fleet of vehicles by optimally assigning locations to different routes. Figure 1.3 illustrates an example of the TOP with two vehicles.

The Orienteering Problem with Time Windows presented by Kantor and Rosenwein [59] incorporates time window constraints to request the visit of each selected node within predefined time intervals. The problem has proved especially useful for applications in which the time is crucial such as urban logistics where meeting delivery schedules is essential or in scenarios like mobile healthcare services that follow set appointment times.

Many OP variants also incorporate probabilistic elements in order to take into account for uncertainties commonly found in dynamic or stochastic environments. The Stochastic Orienteering Problem reported in İlhan et al. [134], considers random profit or service availability at each node reflecting real-life situations where customer presence, demand or service value may be unknown. This variant is especially suitable to demand-responsive services or marketing where expected customer interest may vary significantly across locations. Another stochastic variant, the Orienteering Problem with Stochastic Travel and Service Times presented in Campbell et al. [23] manages the uncertainty in travel and service times by capturing the impact of variable traffic conditions. This kind of applications include on-demand transport in congested urban areas or time-sensitive delivery services where traffic or customer engagement times are uncertain. Dynamic versions of the OP, such as the Time-Dependent Orienteering Problem introduced in Li et al. [71] further address scenarios where travel times can change throughout the day as in urban delivery contexts where peak traffic times must be considered into route planning. Variants of this problem are common in applications like tourism route planning where optimal routes may change based on opening hours or seasonal factors.

OPs have become increasingly relevant in a range of applications that benefit from selective routing. In tourism context, for example, OPs are often used to design multi-day itineraries that maximise the visitors preferencies through optimising the order of attractions. In a disaster relief, the OPs help prioritise routes for rescue teams by focusing on high-priority areas respecting time and resource constraints. In urban logistics, OP variants assist in the planning of last-mile deliveries by selecting profitable stops that can be feasibly reached within service hours.

A key application of the OP is the Tourist Trip Design Problem (TTDP) reported by Gavalas et al. [41]. The TTDP is a specialised optimisation problem in routing that focuses on designing customised itineraries for tourists. The goal is to plan routes that optimise tourist satisfaction by choosing a subset of Points of Interest (POIs) to visit respecting constraints such as available time, personal preferences and specified start and end points. The TTDP has practical relevance in areas like tourist management in which it is essential to balance the quality of the experience with logistical limitations. In the foundational TTDP, an itinerary is constructed by selecting and ordering POIs in such a way that the tourists can experience valuable locations within their time budget. A standard TTDP model aims to maximise the objective function which represents tourist satisfaction based on the interest level assigned to each POI. This base model is commonly expanded to embed constraints such as time windows for POIs, budget limits or accessibility requirements.

In summary, the OPs offer an adaptable framework that help to address routing challenges across different fields. Focusing on the visits selection within constraint-

driven environments, the OPs connect theoretical optimisation advancements with practical and impactful solutions in logistics and in public service.

1.4 Two healthcare applications

Orienteering problems have limited applications in healthcare literature despite to optimise team routing under time and resource constraints is often crucial. The central challenge in these healthcare settings lies in maximising the number of patients served respecting time and resource constraints. Two healthcare applications [7] are introduced here and reported in detail in Chapter 2.

1.4.1 The daily swab test collection problem

The efficiently collection and transportation of the swab tests is a crucial challenge in healthcare. The objective is to ensure the timely retrieval of the swab tests minimising resources consumption. Effectively addressing this logistical issue is essential for enhancing healthcare system efficiency, accelerating diagnostic processes, and improving public health outcomes.

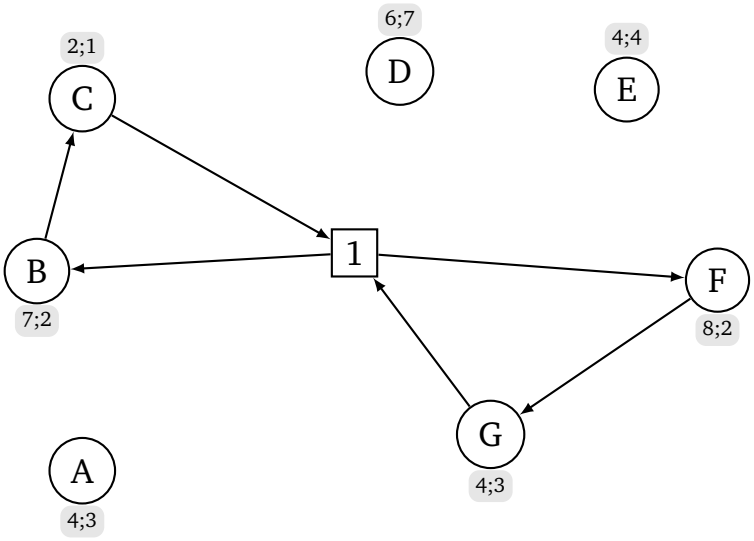


Fig. 1.4: A graphical representation of a feasible solution for the DSTCP. The profits and service times are both indicated inside the gray boxes (the first one is reported on the left, the second one on the right).

The Daily Swab Test Collection Problem (DSTCP) [5] can be formulated as a variant of the TOP considering service times at nodes to model the amount of time needed to collect the swab tests by the operator. The DSTCP specifically focuses on the swab test collection at home for individuals identified as close contacts of confirmed COVID-19 cases via a Digital Contact Tracing (DCT) application. The DCT is essentially a

smartphone application with the aim of tracking individuals' interactions and quickly identify those potentially exposed to the virus after a positive case is reported. This allows for rapid isolation and testing, which is critical to containing the spread of the disease. When a suspected case reports symptoms such as fever or cough, the system triggers an alert for home testing. However, delays in administering and processing these tests can significantly undermine the effectiveness of the DCT system by increasing the risk of further transmission. The picture 1.4 represents an instance of the DSTCP where the pair inside the grey boxes represent the profit and the service time associated to the node.

1.4.2 The ambulance routing problem

Similarly, efficient logistics are crucial in disaster management in which rapid responses can save lives. Disaster management typically involves pre-disaster and post-disaster phases. In the pre-disaster phase, planning is focused on risk mitigation and the development of emergency services. In the post-disaster phase, the focus changes to take into account the relief operations that can include the transportation of supplies, medical personnel and the critical injured patients to hospitals. Humanitarian logistics are important to this response by ensuring the efficient delivery of resources to the affected areas.

The Ambulance Routing Problem (ARP) presented in Aringhieri et al. [5] addresses the challenge of optimising ambulance routes during post-disaster situation. The problem focuses on the determination of the best routes for a fleet of ambulances to transport the most injured patients to hospitals in order to ensure immediate care to them. Two hierarchical objective functions are adopted and compared to drive the optimisation process. The first hierarchical objective function has a primary objective to maximise the score of the non-urgent patients served by the route (*efficiency*) while the secondary is to minimise the maximum waiting time for a urgent patient (*fairness*). On the contrary, the second objective function has the fairness has a primary objective while the efficiency as secondary.

The ARP can be formulated as a variant of the TOP with (i) service times at nodes to medicate the patients, (ii) mandatory nodes to model the urgent patients and (iii) incompatibilities between nodes, where travelling along certain arcs is forbidden, are referred to as physical incompatibilities from this point onward. In the ARP, once a critically injured patient is picked up by an ambulance, the ambulance is required to transport the patient directly to a hospital. This constraint implies that connections from the node representing the critically injured patient to any nodes other than hospitals are prohibited. Picture 1.5 represents a feasible solutions for the ARP.

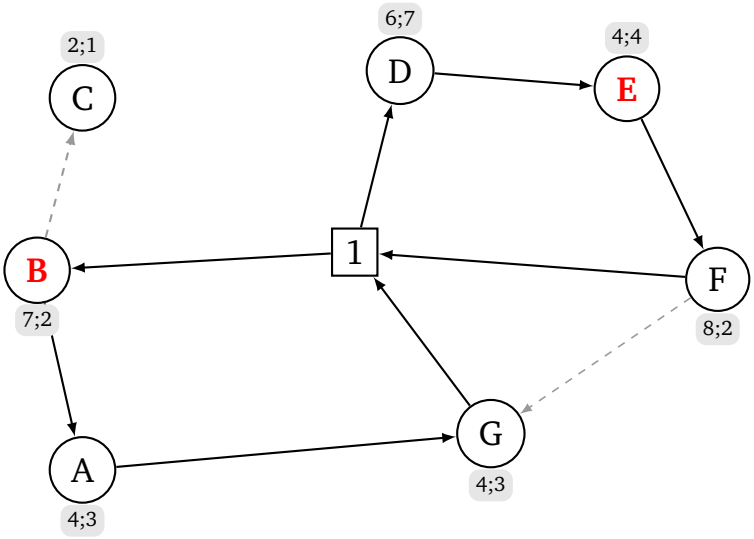


Fig. 1.5: A graphical representation of a feasible solution for the ARP. The mandatory nodes are highlighted in red and the physical incompatibilities are represented by dotted transparent arrows.

1.5 Towards a new variant of the team orienteering problem

Generalizing the two healthcare problems introduced, a new variant of the TOP that incorporates the three features outlined for these problems can be formulated. This new variant is referred to as the Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (TOP-ST-MIN). From the DSTCP, it inherits service times at nodes for tasks such as swab test collection, while from the ARP, it adopts service times for patient care, introduces mandatory nodes to represent critical injured patients that must be visited, and incorporates physical incompatibilities to model the mandatory transportation of these patients to hospitals. To complete the analysis, the TOP-ST-MIN also incorporates a different type of incompatibility between nodes. Specifically, two nodes are considered incompatible if they cannot be served by the same route. This situation can arise in healthcare settings, for example, when multiple ambulances must be assigned to patients with different viral diseases. In such cases, patients with different infectious diseases cannot share the same ambulance. This type of incompatibility is referred to as *logical incompatibility* and it was first introduced by Palomo-Martínez et al. [94] as exclusionary constraints.

In this thesis, two versions of the problem will be considered, each representing different operational contexts: the TOP-ST-MIN-P, which accounts only for Physical incompatibilities, and the TOP-ST-MIN-PL, which takes into account both Physical and Logical incompatibilities. The only work that incorporates all the introduced

features (excluding the physical incompatibilities) is reported in Vu et al. [124], which explores a variant of the TTDP that includes additional constraints such as logical incompatibilities and mandatory nodes. Figure 1.6 represents an instance of the TOP-ST-MIN-PL. The problem will be thoroughly detailed in Chapter 5.

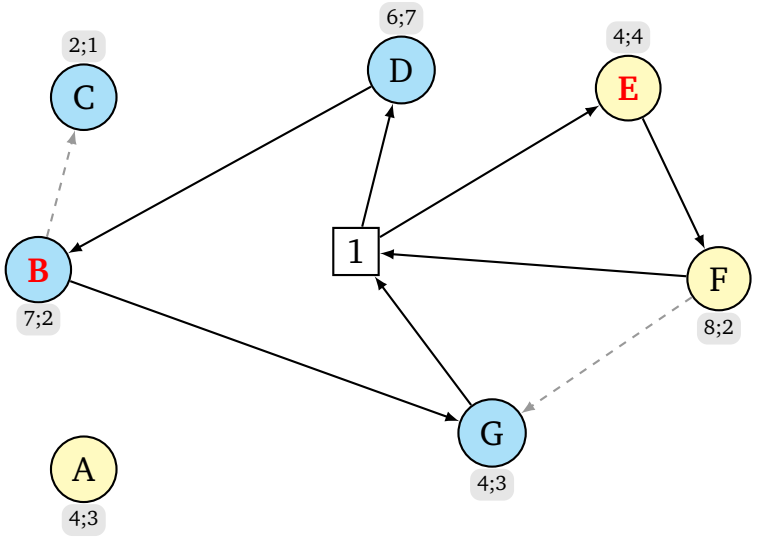


Fig. 1.6: A graphical representation of a feasible solution for the TOP-ST-MIN-PL. In this example, logical incompatibilities are illustrated using different colors representing distinct categories. Then, a vehicle is restricted to visit nodes that share the same color.

1.6 Research questions and thesis plan

The thesis aims to address several research questions in the study of the TOP-ST-MIN. The first question explores which approaches can be employed to analyse the problem in a deterministic point of view. This is done by focusing on the development of mathematical formulations and algorithmic solution methods. The second question investigates how these approaches can be evaluated. In this case, factors like computational efficiency, solution quality and scalability will be evaluated. A third area of inquiry explores how solution quality can be preserved while accounting for the stochastic and dynamic nature of the system, such as uncertain service times or travel durations, and new request arrivals. Finally, this thesis will examine the impact of a fairness perspective on the solution quality. The research questions are the following:

RQ1. Which approaches can be employed to analyse the problem from a deterministic point of view?

RQ2. How can these approaches be thoroughly evaluated?

RQ3. How can a certain level of solution quality be ensured when accounting for stochasticity and dynamicity of the system?

RQ4. How to measure the impact of a fairness perspective in the solutions quality?

These research questions are addressed throughout the chapters of the thesis.

A comprehensive overview of the DSTCP and the ARP that served as inspiration for the development of the TOP-ST-MIN is presented in Chapter 2. This chapter explores these two problems providing valuable insights and methodologies. This chapter demonstrates that the TOP-ST-MIN reflects a problem with practical real-world relevance. Chapter 3 reviews the literature for the problems related to the TOP-ST-MIN covering deterministic and uncertainty points of view reporting mathematical models, heuristic and exact solution approaches. Finally, the chapter also reports the most relevant works inherent to the concept of fairness and their application to optimisation problems. This chapter highlights that the TOP-ST-MIN has not been previously studied in the existing literature. Therefore, Chapters 2 and 3 justify that the TOP-ST-MIN deserves to be studied.

Chapter 4 covers the generation of instances. It presents deterministic methods for creating a new set of instances in order to analyse the impact of the three features characterising the TOP-ST-MIN on the computational effort required to solve the instances. The chapter outlines two generation methods for each of the mandatory nodes, physical incompatibilities, and logical incompatibilities, along with one for service times. Consequently, this chapter addresses the second research question.

The TOP-ST-MIN is formally presented in Chapter 5, which provides an answer to the first research question. This chapter introduces two mathematical models for the TOP-ST-MIN, an exact cutting-plane algorithm for optimal solutions and two heuristic methods for larger instances. A thorough evaluation is conducted in order to measure the performance of these approaches on the newly generated instances, along with a comparative analysis against the state-of-the-art TOP algorithms using standard benchmark datasets.

The TOP-ST-MIN becomes more challenging under uncertainty and dynamic changes. To deal with this operative setting, Chapter 6 introduces a stochastic variant of the TOP, which incorporates uncertainty in travel times. This is referred to as the Stochastic Team Orienteering Problem (STOP). The chapter presents two exact solution methods based on a two-stage with recourse programming approach. Computational experiments validate the proposed methods using the TOP reference dataset. Then, in Chapter 7, an adapted version of the STOP including uncertain service time is used to deal with the TOP-ST-MIN under uncertainty and dynamic changes. This chapter introduces four algorithms based on the Branch & Regret paradigm, a scenario-driven approach using Discrete Event Simulation (DES) and optimisation.

Extensive computational tests validate the proposed methods. Together, these two chapters address the third research question.

Chapter 8 adopts an alternative perspective to evaluate the solution quality of the TOP-ST-MIN by incorporating the concept of fairness in order to address the final research question. This study is inspired by the fairness insights reported in Aringhieri et al. [5]. The chapter introduces three fairness functions and several indicators to measure the impact of the fairness on the solution quality. Fairness is examined from two perspectives. The first one, derived from the literature, focuses on balancing team profits, while the second, based on a more practical context, aims to balance the cumulative waiting times experienced by the customers. The concept of fairness is further extended in order to balance the cumulative waiting times over an extended period, adopting the notion of fairness over time for the TOP-ST-MIN. The chapter concludes with a detailed computational analysis.

Chapter 9 wraps up the thesis by highlighting the relevance of the work carried out on the TOP-ST-MIN and by providing a overview of future research directions.

Two healthcare applications

2

Effective resource allocation in healthcare is crucial in scenarios as swab test collection during pandemics and ambulance routing in post-disaster management. Both situations need solid plans in order to prioritise critical tasks optimising the use of available resources. Swab test collection aims to optimise the collection of swab tests. Ambulance routing instead, focuses on transporting patients to hospitals. This chapter delves into two critical real-application problems in healthcare. The first one is the Daily Swab Test Collection Problem (DSTCP) in which medical teams must visit multiple locations collecting swabs respecting a specified time limit. The problem also requires to prioritise individuals based on factors such as vulnerability and transmission risk. The second one, the Ambulance Routing Problem (ARP), focuses on optimising ambulance routes in post-disaster scenarios to transport the critical injured patients to hospitals.

Both problems, introduced in Sections 1.4.1 and 1.4.2, have been formally analysed in Aringhieri et al. [7] and Aringhieri et al. [6]. These problems represent two operative applications that motivate the study of the TOP-ST-MIN that will be analysed from a methodological perspective in Chapter 5.

2.1 The team orienteering problem

Both problems can be modelled using the Team Orienteering Problem (TOP) as basic framework. The TOP is a well-known optimisation problem in operations research defined by using a complete graph $G = (N, A)$ where the set N is the set of nodes and A, the set of arcs. Each node k has an associated score p_k, each arc (i, j) has an associated cost t_{ij} and the goal is to find a set $R = \{1, \ldots, m\}$ routes inside G in such a way to maximize the total profit collected. Each route must start with the source node 1 and must end with the destination node n and cannot exceed a time budget constraint $T_{\max}$. The TOP serves as a versatile framework for modelling a wide range of logistical and planning challenges, making it particularly suitable for complex real-world scenarios.

The TOP can be represented as an integer programming problem using the following decision variables: $x_{ijr} = 1$ if node i is followed by node j in the route r, 0 otherwise; $y_{kr} = 1$ if node k is visited by the route r, and 0 otherwise. The variables

u_{kr} denote the position of the node k in the route r. Then, the classical mathematical formulation of the TOP is the following reported in Vansteenwegen et al. [121]:

$$\max \quad \sum_{r=1}^{m} \sum_{k=2}^{n-1} p_k \, y_{kr} \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{r=1}^{m} \sum_{j=2}^{n} x_{1jr} = \sum_{r=1}^{m} \sum_{i=1}^{n-1} x_{inr} = m, \tag{2.1b}$$

$$\sum_{i=1}^{n-1} x_{ikr} = \sum_{j=2}^{n} x_{kjr} = y_{kr}, \quad \forall\, k = 2, \ldots, n-1,\ r \in R, \tag{2.1c}$$

$$\sum_{i=1}^{n-1} \sum_{j=2}^{n} t_{ij} \, x_{ijr} \leq T_{\max}, \quad \forall\, r \in R, \tag{2.1d}$$

$$\sum_{r=1}^{m} y_{kr} \leq 1, \quad \forall\, k = 2, \ldots, n-1, \tag{2.1e}$$

$$2 \leq u_{ir} \leq n, \quad \forall\, i = 2, \ldots, n,\ r = 1, \ldots, m, \tag{2.1f}$$

$$u_{ir} - u_{jr} + 1 \leq (n-1)(1 - x_{ijr}), \quad \forall\, i, j = 2, \ldots, n,\ r \in R, \tag{2.1g}$$

$$x_{ijr} \in \{0, 1\}, \quad \forall\, i, j = 1, \ldots, n,\ r \in R, \tag{2.1h}$$

$$y_{kr} \in \{0, 1\}, \quad \forall\, k = 2, \ldots, n-1,\ r \in R \tag{2.1i}$$

The objective function (2.1a) is to maximise the total collected score. Constraints (2.1b) guarantee that each route starts from the source node and ends to the destination node. Constraints (2.1e) ensure that every node is visited at most once. Constraints (2.1c) guarantee the connectivity of each route. Constraints (2.1d) ensure the limited time budget for each route. Constraints (2.1f) and 2.1g are necessary to prevent subtours. Constraints (2.1h) and (2.1i) are variable definition constraints.

In healthcare applications, the TOP can be adapted to address operational problems where prioritisation, efficiency and resource allocation are essential. The TOP serves as the foundational framework for addressing two significant real-world healthcare challenges: optimising the collection of diagnostic swabs during a pandemic and efficiently routing ambulances to deliver emergency medical services in the aftermath of a disaster. Both problems share some features with the TOP such as the need to prioritise service based on urgency and the requirement to operate under time constraints. Exploiting the TOP framework, these healthcare challenges are modelled to enable the development of efficient, practical solutions that support critical medical operations.

2.2 The daily swab test collection problem

Contact tracing is a tool to mitigate the outbreak of pandemic diseases. However, as the rate of disease transmission accelerates, manual contact tracing becomes

increasingly inefficient. Digital Contact Tracing (DCT) has emerged as an effective alternative during the spread of SARS-CoV-2 (COVID-19) as demonstrated by Ferretti et al. [37]. In their work, the authors highlighted that the DCT could successfully control the epidemic. DCT serves as a critical element in the "Triple T" strategy: Test, Trace, and Treat. This approach consists in the use of a smartphone application to automatically trace individual contacts in order to enable quick responses as illustrated in Figure 2.1. When an individual reports symptoms (e.g., fever, cough), he begins isolation and requests a home test. Prompt collection and processing of this test are essential to quickly notify their recent contacts. Any delay in testing, however underminine the effectiveness of the DCT system. Ferretti et al. [37] revealed that attaining around 60% adoption of the DCT app within a population could effectively stop the epidemic. Several countries worldwide, particularly in Europe

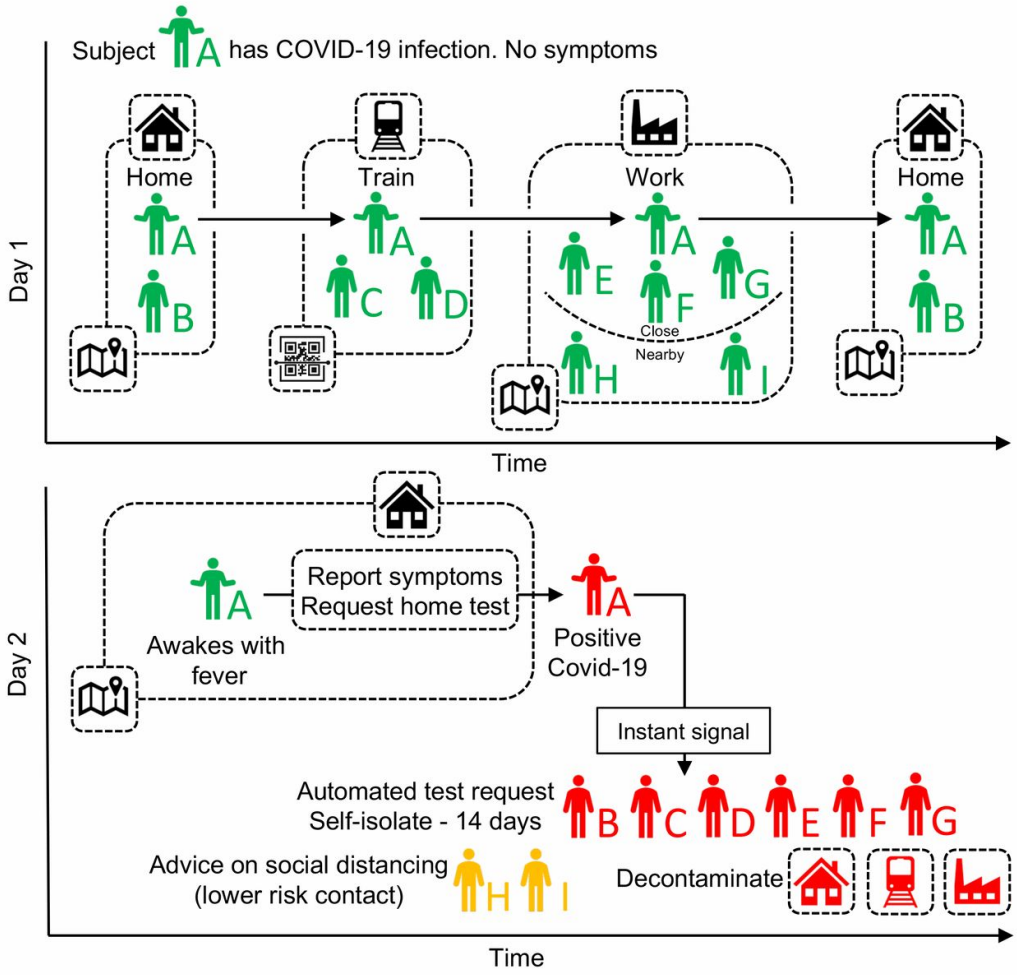


Fig. 2.1: General description of digital contact tracing (from Ferretti et al. [37]).

have implemented DCT. To improve their utility, the European Union has developed an EU-wide system ensuring interoperability among these apps. This initiative aims

to maximise the potential of DCT in breaking the chains of COVID-19 transmission across borders saving lives.

Despite the global rollout of DCT apps in many countries, insufficient attention has been paid to the healthcare logistics systems that support tracing activities. This study appears to be the first to investigate the healthcare logistics supporting DCT from an Operations Research perspective. The introduction of DCT presents significant challenges for local health systems, particularly regarding the volume of daily tests required during periods of high viral transmission. The effectiveness of DCT is directly connected to the ability of local health systems to test the majority of individuals notified by the application as close contacts of positive cases. This raises a critical research question: can the underlying logistics system be optimised to ensure timely testing for individuals identified through DCT?

The Daily Swab Test Collection Problem (DSTCP) focuses on optimising daily routes in order to collect swab tests directly at home of those people identified as contacts of positive cases detected in previous days. Medical teams are responsible for navigating the city to carry out these collections. The priority levels allocated to individuals are based on health conditions and social interactions. These priorities indicate the urgency to be tested fostering individuals who are likely to spread the virus or who belong to high-risk groups. This prioritisation is essential to select which swab tests to collect. Time constraints play a crucial role in this problem because it takes into account travel times between locations and the time required to collect swabs at each site. Efficient route planning is crucial to maximize the number of high-priority tests completed within the teams working hours.

Let $P = \{2, \dots n-1\}$ be a set of nodes where a number b_p ($p \in P$) of swab tests should be collected. The collection of the swab tests follows an integer priority r_p: the greater the priority is, the greater the importance of collecting such a swab test is. When the number of required tests are larger than the daily capacity, the priority represents the need of testing some people before other since they could become a spreader of the virus and/or they belong to a more risky class of people (e.g., elderly and/or frail people). In the operative context, the priority is provided by the health authority since the DCT data are private. Further remarks are discussed in the conclusions.

For each node $p \in P$, the estimated times s_p^+ and s_p^- represent the durations to dress and to undress the *personal protective equipment*, the time t_p^h to reach the house, and the time s_p^s to collect a single swab test. Therefore, the overall time s_p needed to perform all the operations needed to collect a swab test at the node $p \in P$ is equal to $s_p = s_p^+ + 2\, s_p^h + (b_p\, s_p^s) + s_p^-$. Let $K = \{1, \dots, m\}$ be a set of medical teams in charge of collecting swab tests during their work-shift whose maximum duration is equal to $T_{\max}$. The teams start their work-shift from a *depot* 1 ending at the laboratory n.

Depot and laboratory could be the same node. Considering $P^+ = P \cup \{1, n\}$, let t_{pq} be the travelling time to reach $q \in P^+$ from $p \in P^+$.

The DSTCP is modelled as a variant of the TOP with service time at nodes by using the following set of decision variables:

- $x_{pqk} = 1$ if the team $k \in K$ visits the node $q \in P^+$ immediately after visiting the node $p \in P^+$, 0 otherwise.
- $y_{pk} = 1$ if the node $p \in P^+$ is visited by team $k \in K$, 0 otherwise.
- u_{pk} is an integer representing the position of the node $p \in P^+$ in the path of the team $k \in K$.

The objective function seeks to maximise the overall priority of the swab tests collected:

$$\max z = \sum_{k=1}^{m} \sum_{p \in P} r_p \, b_p \, y_{pk}. \tag{2.2}$$

The rationale behind the idea of multiplying the priority for the number of requests on a node is that the collection of a greater number of swab tests is to be preferred. This is under the assumption that all the requests on a node are satisfied when it is visited, and is consistent with the objective of maximising the sum of priority scores associated with the collection of single swab tests. Constraint 2.3 ensures that each team starts its work-shift from the depot and ends it at the laboratory:

$$\sum_{k=1}^{m} \sum_{q \in P} x_{1qk} = \sum_{k=1}^{m} \sum_{p \in P} x_{pnk} = m, \tag{2.3}$$

where k is the number of teams. Constraints 2.4 ensure that every node is visited at most once:

$$\sum_{k=1}^{m} y_{pk} \leq 1, \quad \forall \, p \in P. \tag{2.4}$$

Constraints 2.5 guarantee the connectivity of the work-shift of each medical team:

$$\sum_{q \in P \cup \{1\}} x_{qpk} = \sum_{q \in P \cup \{n\}} x_{pqk} = y_{pk}, \quad \forall \, p \in P, \, k \in K. \tag{2.5}$$

Constraints 2.6 ensure that the duration of each work-shift is less than or equal to the maximum duration:

$$\sum_{p \in P \cup \{0\}} \sum_{q \in P \cup \{n+1\}} t_{pq} x_{pqk} + \sum_{p \in P} s_p y_{pk} \leq T_{\max}, \quad \forall \, k \in K. \tag{2.6}$$

Finally, the constraints 2.7 and 2.8 are necessary to prevent subtours in accordance with the Miller-Tucker-Zemlin formulation for the TSP Miller et al. [84]:

$$2 \leq u_{pk} \leq n, \quad \forall \, p \in P \cup \{n\}, \, k \in K. \tag{2.7}$$

$$u_{pk} - u_{qk} + 1 \leq (n-1)(1 - x_{pqk}), \quad \forall\, p, q \in P \cup \{n\},\ k \in K. \tag{2.8}$$

2.3 Fairness in ambulance routing for post disaster management

The organisation responsible for disaster response changes significantly across countries (see, e.g., Boin et al. [19]). For example, the Italian civil protection system operates under the principle of subsidiarity [22] in which larger administrative bodies are progressively involved as the scale and intensity of the disaster increase. Conversely, other European countries often employ more centralised systems. These differences result in distinct organisational frameworks at both tactical and operational levels. The problem addressed in this section draws inspiration from the formulation presented in Talarico et al. [113]. When a disaster strikes, initial data on damages and injuries is collected rapidly. Dispatchers classify patient needs based on severity and location and ambulances are deployed to provide on-site treatment and transport critical injured patients to hospitals. Patients in the affected area are categorised into two groups based on a triage system: red patients and green patients. The former require immediate transport to hospitals due to severe injuries while the latter only need first aid directly on-site.

For green patients, a priority score is adopted to rank their urgency levels, guiding the selection of those who should receive prompt attention. These scores act as a priority in the optimisation process with the goal of maximising the total profit of the served green patients. However, due to time constraints associated with ambulance work shifts (imposed for safety reasons), not all green patients may be attended to, necessitating prioritisation of those with the highest scores. In contrast, red patients, suffering from critical injuries, must be transported to hospitals promptly. The objective here is to minimize the maximum waiting time for red patients, ensuring equitable service delivery.

Ambulance operations begin at a designated depot (e.g., a hospital or medical center), from which they visit patients in affected areas and return to a hospital. Green patients receive treatment on-site in order to allow the ambulance to continue to the next green patient. Red patients, instead, are transported to hospitals in order to receive ad-hoc treatments. In this context, the study assumes that hospitals have sufficient capacity to treat all patients, following a common practice of stabilising patients and then transferring them to other facilities as needed to optimise resource utilisation and free up space for new arrivals.

The Ambulance Routing Problem (ARP) can be formulated as a variant of the TOP with (i) service time at nodes, (ii) mandatory nodes representing the red patients and (iii) incompatibilities among nodes which reflect the requirement to transport

red patients to hospitals. The goal is to find the optimal route for ambulances to deliver the services to patients after a disaster. Note that in the TOP each node can be visited once except for the source node and the destination node. The problem is modelled on a complete graph $E = (N, A)$, where N is the set of nodes representing locations over the considered area and $A = \{(i, j) : i, j \in N\}$ is the set of arcs indicating the connections between each pair of such locations. Three main types of nodes are considered in this problem: (i) green patients (set G), (ii) red patients (set R), and (iii) actual hospitals (set O). To align with the TOP formulation, additional dummy nodes are introduced: (iv) the dummy source and destination depot nodes 1 and n, and (v) several dummy hospitals (set D), which are replications of the actual hospital nodes (i.e. they have same coordinates of the original nodes of the set O) that allows to visit two or more times the same actual hospital visiting two different nodes. The number of replications for each actual hospital node has been set to a value that allows the hospital to serve as both the starting point for each route and the destination node for each red patient taking into account for the worst-case scenario where this hospital exclusively accommodates all tours and red patients. Furthermore, the set of patients is denoted as $P = G \cup R$ and the set of hospitals (actual and dummy) with $H = O \cup D$. Therefore, $N = P \cup H \cup \{1, n\}$. Nodes are enumerated from 1 to n with $n = |N|$. The priority score of green patients is denoted by s_i, while the set of available ambulances is indicated with $K = \{1, \ldots, m\}$. An

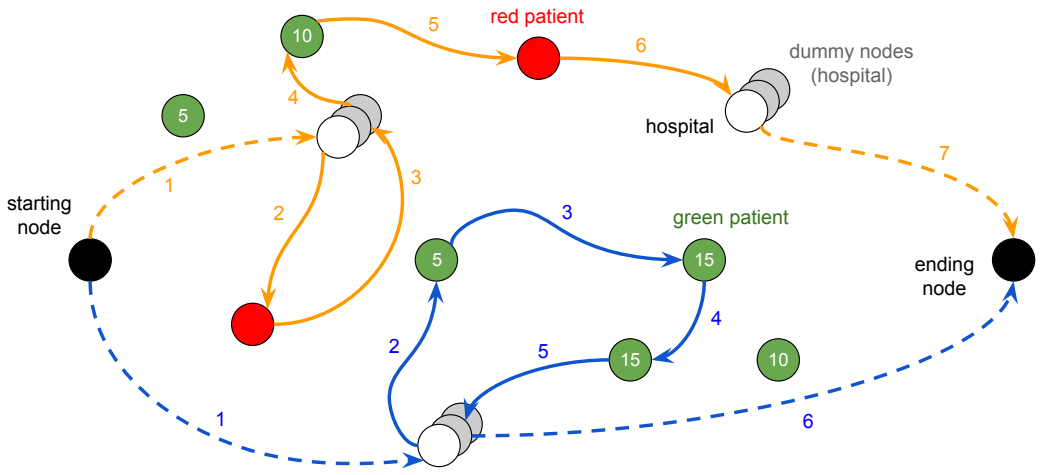


Fig. 2.2: Example: solution of a TOP formulation representing the ambulance tours after a disaster. Each route starts from the starting dummy node and ends in the ending dummy node, and hospitals are always visited in accordance with the graphic overlay. Numbers on arcs indicate the order in which they are visited in the route.

example of the introduced graph and the related solution is explained in Figure 2.2. Nodes of sets G, R and O are coloured in green, red and white, respectively. Labels on green nodes indicate the scores s_i, for each $i \in G$. In correspondence of each node of O, several replications are provided in order to suit the problem to the TOP framework; such nodes, in grey, are the elements of D. Black nodes represent the

dummy depots $\{1, n\}$. Two ambulance tours are represented by the sequence of coloured arcs (orange and blue) from the source depot to the destination depot: the actual movement of the ambulance between two physical places represented by the nodes, while dashed arcs are only used to connect the dummy depots to the actual hospital where the ambulance is located at the beginning and at the end. The connections of the node 1 are set in accordance with the instance of the problem: the first arc visited by the route of an ambulance is that between 1 and one of the nodes related to the hospital in which it actually is located at the beginning. Such an initial location is given by the parameter l_{hk}, which is equal to 1 if the ambulance $k \in K$ is initially parked at the hospital h. The connections of the node n have the same meaning, but they are part of the solution because there is no constraint about the final destination of the ambulances (except that it must be a hospital node). Furthermore, it is possible to observe that each intermediate node is visited at most once: if a hospital needs to be visited twice or more, then a different node belonging to H, and related to that hospital, is visited every time; for instance, the orange route visits the actual hospital node firstly and a dummy node for the second visit. Finally, red nodes must be visited once, while green, white and grey nodes could be not visited.

Since the locations of patients and hospitals are identified, the travel time between a pair of nodes $i, j \in N$ is defined as t_{ij}, where:

- $t_{ij} = t_{ji}$ for each $i, j \in N$.
- $t_{ij} = 0$ if $i \in \{1, n\}, j \in H$.
- $t_{ij} = 0$ if $i, j \in H$ are replications of the same actual hospital.

The time spent by ambulances at nodes is referred to as the service time f_i for each $i \in N \setminus \{1, n\}$. Such time represents: (i) the on-place treatment duration when $i \in G$, (ii) the preparation time spent on the place when $i \in R$, and (iii) the time spent to release a red patient to the hospital when $i \in H$. Hospitals are assumed to accept red patients without capacity limitations, and each ambulance is restricted to carrying only one red patient at a time, without attending to other patients during this period. A maximum time T_{max} is set for each ambulance to complete its assigned tasks, including treating patients and finishing its route at a hospital. The notation used in the model is summarised in Table 2.1.

The linear programming model of the problem is based on the TOP framework. Constraint (2.9) ensures that all ambulances start their visit from the dummy depot node 1 and end on the dummy depot node n.

$$\sum_{k \in K} \sum_{j \in P} x_{1jk} = \sum_{k \in K} \sum_{i \in P} x_{ink} = |K| \tag{2.9}$$

Tab. 2.1: Notation.

Sets		**Parameters**	
N	all nodes	t_{ij}	travelling time between i and j
G	green patients	f_i	service time in the node i
R	red patients	T_{max}	maximum route duration
P	all patients	s_i	priority score of the green patient i
O	actual hospitals	l_{hk}	initial location of k (1 if it is h, 0 otherwise)
D	dummy hospitals		
H	all hospitals		
K	ambulances		
	Decision variables		
x_{ijk}	takes value 1 if ambulance k visits node i directly before node j, 0 otherwise		
y_{ik}	takes value 1 if ambulance k visits patient i, 0 otherwise		
z_{hk}	takes value 1 if ambulance k visits hospital h, 0 otherwise		
m_{ih}	takes value 1 if red patient i is transported to hospital h, 0 otherwise		
u_{ik}	position of the patient i in the route of the ambulance k		
w_i	visit time of the node i (waiting time if i is a patient)		
C_{max}	maximum completion time of all red patients		

Constraints (2.10) impose that at the beginning each ambulance $k \in K$ is located to the hospital node $h \in H$, in accordance with the value of the parameter l_{hk}.

$$x_{1hk} = l_{hk}, \quad \forall\, h \in H, k \in K \tag{2.10}$$

Constraints (2.11)-(2.12) guarantee that each green patient or hospital is visited at most once and each red patient must be visited once, respectively.

$$\sum_{k \in K} y_{ik} \leq 1, \quad \forall\, i \in G \cup H \tag{2.11}$$

$$\sum_{k \in K} y_{ik} = 1, \quad \forall\, i \in R \tag{2.12}$$

Constraints (2.13) imply that each ambulance should go to a hospital after visiting a red patient node.

$$\sum_{j \in P} \sum_{k \in K} x_{ijk} = 0, \quad \forall\, i \in R \tag{2.13}$$

Constraints (2.14) enforce the route connectivity: when an ambulance visits a patient, it also has to leave that location. Furthermore, they allow the consistency of the decision variables x_{ijk} and the corresponding y_{ik}.

$$\sum_{j \in N \setminus \{n\}} x_{jik} = \sum_{j \in N \setminus \{1\}} x_{ijk} = y_{ik}, \quad \forall\, i \in P, k \in K \tag{2.14}$$

Constraints 2.15 ensure the respect of the time limit T_{max} for each ambulance route.

$$\sum_{i \in P \cup H} \sum_{j \in P \cup H} t_{ijk} x_{ijk} + \sum_{i \in P} f_i y_{ik} + \sum_{h \in H} f_h z_{hk} \leq T_{max}, \quad \forall\, k \in K \tag{2.15}$$

Constraints 2.16 fix z_{hk} to 1 when the hospital h is visited by the ambulance k.

$$x_{ihk} \leq z_{hk}, \quad \forall\, i \in R, h \in H, k \in K \tag{2.16}$$

Constraints (2.17)-(2.19) are introduced to properly compute the visit time of the nodes, that is the waiting time for patient nodes and the arrival time to the hospital for the (actual and dummy) hospital nodes. The waiting time of each patient visited directly after a green patient is fixed by the constraints (2.17), which take into account the waiting time and the service time of that patient, adding the travelling time between the two nodes. Similarly, constraints (2.18) set the waiting time of a patient visited directly after a hospital. In this case the service time at the hospital is considered only if that node is not the starting location of the ambulance. The same rationale is used for the constraints (2.19), which allows the computation of the arrival time at the hospital after visiting a patient node (that can be after a red patient node or after the last green patient of the ambulance route). Big-M is used in these three constraints.

$$w_i + f_i + t_{ij} \leq w_j + (1 - x_{ijk})M, \quad \forall i \in G, j \in P, k \in K \tag{2.17}$$

$$w_h + (1 - l_{hk})f_h + t_{hj} \leq w_j + (1 - x_{ijk})M, \quad \forall h \in H, j \in P, k \in K \tag{2.18}$$

$$w_i + f_i + t_{ih} \leq w_h + (1 - x_{ihk})M, \quad \forall i \in P, h \in H, k \in K \tag{2.19}$$

Figure 2.3 describes describes how constraints 2.17, 2.18 and 2.19 work. To simplify

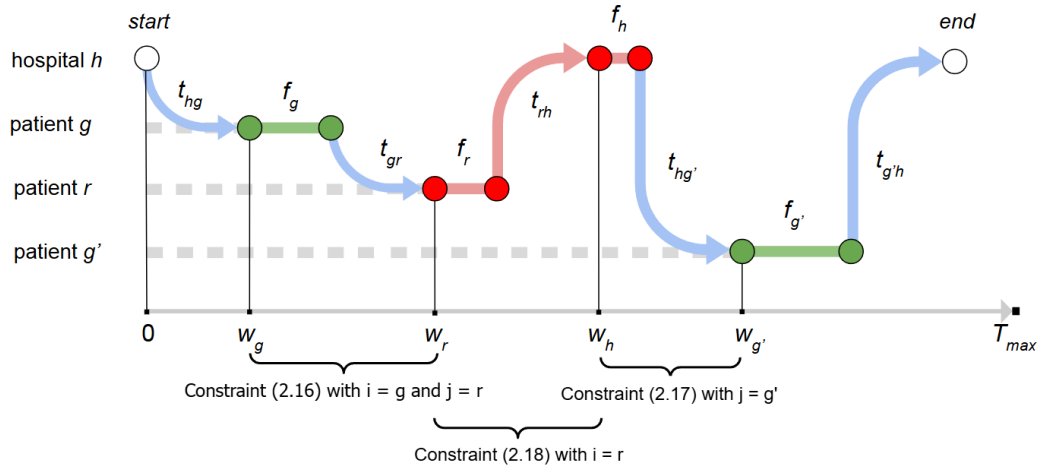


Fig. 2.3: Example: how the constraints (2.17)-(2.19) work.

the description, the assumption can be made that there is only one actual hospital h, with all its dummy nodes represented using the same notation.

In the example depicted in the figure, a single route is considered since the waiting times for patients visited on different routes are independent and have no impact on each other. Then, the ambulance starts from the hospital h, it moves to the green patient g, which requires a certain service time and a travelling time to reach the next node, that is the patient r. After visiting r on her/his location with a duration equal to the patient's service time f_r, the patient is transported to the hospital h (i.e., on a node which is one of its replications), from which the ambulance travels to the the green patient g' after a service time f_h. When the service time of this patient is elapsed, the ambulance goes back to the hospital to finish the route. Constraints (2.17) relate the waiting times of the patients g and r, then constraints (2.19) bind the visit time of h on the basis of w_r, and finally the waiting time of g' is related to the visit time of r through constraints (2.18).

Constraints (2.20) are used to compute the maximum completion time of the red patients, that will be minimised within the proposed objective functions. The completion time is defined as the time elapsed from the beginning of the ambulance route up to the release of the patient at the hospital. Constraints (2.21) are used to compute the value of the binary variables m_{ih}, which indicate in which hospital a red patient is transported and is used in constraints (2.20).

$$C_{max} \geq w_i + f_i + m_{ih}(t_{ih} + f_h), \quad \forall i \in R, h \in H \tag{2.20}$$

$$m_{ih} \geq \sum_{k \in K} x_{ihk}, \quad \forall i \in R, h \in H \tag{2.21}$$

Finally, constraints (2.22)-(2.23) are necessary to prevent sub tours.

$$2 \leq u_{ik} \leq n, \quad \forall i \in N \setminus \{1\}, k \in K \tag{2.22}$$

$$u_{ik} - u_{jk} + 1 \leq (n-1)(1 - x_{ijk}), \quad \forall i, j \in N \setminus \{1\}, k \in K \tag{2.23}$$

As discussed at the beginning of this section, two objectives emerge from the analysis of the problem: the fairness in reducing the red patient waiting times, and the efficiency in collecting the green patient scores. The former is the minimisation of C_{max}, while the latter is the maximisation of the value σ_G equal to the sum of the scores of all the green patients visited. Addressing these conflicting objectives simultaneously requires employing a hierarchical objective function modelling approach. This method prioritises optimising the higher-level objective first and subsequently enhances the lower-level objective as much as possible, ensuring that the higher-level objective remains unaffected.

Such objective functions are subject to the constraints (2.9)-(2.23) and consist of two levels corresponding to two different goals. The former z_f (2.24) aims to

minimise C_{max} at the higher level and maximising of σ_G at the lower level. This objective function guarantees to obtain a fair solution where the maximum waiting time experienced by the urgent patients is minimised. The latter z_e (2.25) reverses the optimisation order and guarantees that the overall profit collected is maximised. The constants α_1 and α_2 have to be fixed in such a way to ensure the hierarchy of the two components from a numerical point of view. A possible setting can be: $\alpha_1 = \sum_{i \in G} s_i + 1$ and $\alpha_2 = T_{max} + 1$.

$$\min \; z_f \; = \; \alpha_1 C_{max} - \sum_{k \in K} \sum_{i \in G} s_i y_{ik} \tag{2.24}$$

$$\max \; z_e \; = \; \alpha_2 \sum_{k \in K} \sum_{i \in G} s_i y_{ik} - C_{max} \tag{2.25}$$

As discussed in Anaya-Arenas et al. [2], the min-max approach is just one of the possible ways to represent and model the fairness. Alternative approaches include deprivation-like models or survival functions (see, e.g., Knight et al. [64]). The approach follows Talarico et al. [113], which inspired this work and is justified by the fact that the min-max approach seems more suitable to evaluate the price of fairness [90] through the hierarchical objective functions (2.24) and (2.25). To explain better how the two objective functions z_f (2.24) and z_e (2.25) work,

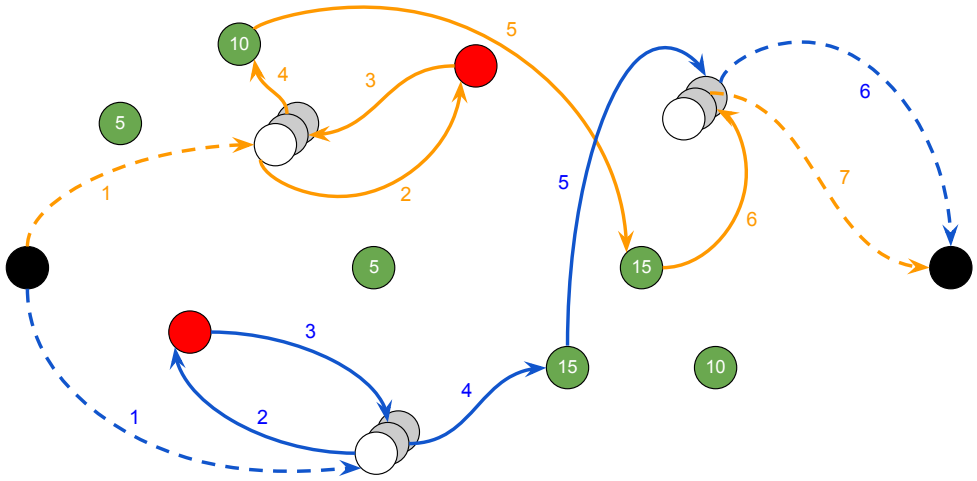


Fig. 2.4: Example: fairness solution of a TOP formulation representing the ambulance tours after a disaster. Each route starts from the starting dummy node and ends in the ending dummy node, and hospitals are always visited in accordance with the graphic overlay. Numbers on arcs indicate the order in which they are visited in the route.

consider the example depicted in Figure 2.2, which could represent the best solution for the proposed model using z_e. It is possible to observe that in the orange route a red patient is visited after a green patient. Such a route can not be an optimal solution using z_f: as a matter of fact, a change of the visit order of those patients can minimise the value of C_{max}, which means that red patients have a lower maximum completion time. When the fairness component is prioritised at the higher level of

the hierarchical objective function, red patients are typically expected to be visited at the start of the $|K|$ tours, as illustrated in Figure 2.4. After minimising the value of C_{max}, the objective function (2.24) maximises the scores collected in the green nodes without having an impact on the red patients. Conversely, the tours in Figure 2.2 maximise the collected scores, but they do not take into account how soon the red patients are visited, unless it is possible to decrease their completion time without decreasing the value of σ_G.

2.4 Solution approaches

Given that both the DSTCP and ARP involve large-scale optimisation they require the use of heuristic methods for practical requirements. These problems involve multiple variables and constraints, making exact solutions computationally expensive or infeasible within reasonable time limit.

Firstly, this section will discuss an initial solution approach based on spectral clustering [89]. Spectral clustering is a machine learning technique for partitioning data points into groups exploiting the eigenvalues and eigenvectors of a similarity matrix. This matrix captures the pairwise similarities between data points. The method maps the data into a lower-dimensional space in which the clusters become easily to separate. This process typically consists in constructing a graph where each data point is a node and edges between nodes is a kind of measure of similarity between points. These eigenvectors are used to embed the data into a new space, where standard clustering algorithms like k-means [81] can be applied to identify the clusters. Since the graph's structure can assume arbitrary shapes, the spectral clustering algorithm was selected for its superior performance compared to classical algorithms like k-means which are effective only for globular clusters.

Secondly, it will be explained a neighbourhood search algorithm that improves the initial solution. The neighbourhood search is an optimisation technique that explores the solution space by iteratively move the current solution to near solutions by doing small changes in the current one. The idea consists in starting from a solution and then generate a set of solutions. These solutions are generated by small perturbations of the current solution. After that, the algorithm evaluates these newly generated solutions and, if a better solution is found, it becomes the new current solution. The process continues until a stopping condition is met, for example after reaching a maximum number of iterations. neighbourhood search strategy is commonly used in local search algorithms and metaheuristics offering a balance between solution quality and efficiency.

2.4.1 Initial solution based on a machine learning approach

Both problems rely on a shared procedure to compute the initial solution, which adopts the spectral clustering algorithm to partition the initial set of nodes into $(c_1, \ldots, c_m)$ clusters. For each vehicle r, the corresponding cluster c_r serves as input for a procedure that generates the associated route v_r. Each route begins at the source node, visits a subset of nodes within the cluster and concludes at the destination node. The method used to compute this set of tours differs significantly and is tailored to the specific problem being addressed.

The daily swab test collection problem

For each vehicle k, a 0-1 knapsack problem [83] is solved in which the nodes belonging to the cluster c_k are the elements to be inserted in the knapsack whose capacity is set to $T_{\max}$, and each node $p \in c_k$ has weight equals to the service time s_p and profit equals to $r_b\, b_p$. This instance of the 0-1 knapsack is solved by applying the classic dynamic programming algorithm [82]. The subset of nodes selected by solving the 0-1 knapsack are then the input for a TSP procedure to compute the route r_k. It is worth noting that if the cost r_k exceeds the time budget $T_{\max}$, the procedure removes the node with the lowest profit and recalculates the route. This process iterates until the cost r_k is less than or equal to $T_{\max}$. In the implementation, such a route is computed using a TSP heuristic approach (both Christofides [25] and Lin and Kernighan [75] have been tested) starting from the source node. The resulting solution is then modified connecting the last node directly to the destination node, rather than returning to the source.

The ambulance routing problem

The clusters computed are then used as input for a customised single-team version of the linear program described in Section 2.3 to calculate the initial tour for the team assigned to each of the m clusters. When optimising the z_f objective function, the customised linear program ensures that all red nodes are visited before the first green node. In contrast, when optimising the z_e objective function, the same customised linear program is used, but with the goal of maximising the overall score.

2.4.2 Improving the initial solution

The initial solution is composed of m feasible routes that could be non-optimal. Such a solution is then improved by the neighbourhood search procedure. Neighbourhood search is an optimisation procedure that improves a solution iteratively by exploring a set of neighbouring solutions, typically defined by small and problem-specific changes to the current solution. It evaluates these solutions in order to identify improvements and iterates the process until no significant enhancements are possible.

This method is commonly employed in local search algorithms to efficiently explore the solution space. The algorithm utilises neighbourhoods specifically tailored to the problem at hand.

The daily swab test collection problem

Two algorithms have been developed. The first, the `ClusteringSearch` CS1 adopts two different schemes to intensify the search and a single diversification procedure. The algorithm exploits two different neighbourhoods to intensify the search and a single diversification method:

- The `intraClusterSearch` tries to improve the route of each cluster exchanging nodes in the route with nodes not yet visited but in the same cluster.
- The `interClusterSearch` modifies the nodes clusters by exchanging a pair of nodes from different clusters.
- Finally, after a number of non-improving iterations, a restart is performed to diversify the search. For each cluster c_k, the restart randomly relocates 25% of its nodes to the nearest cluster to c_k. After that, the tours for the modified clusters are re-computed in accordance with the same procedure used for computing the initial feasible solution.

The second algorithm, the `ClusteringSearch` CS2 adopts the same scheme of CS1 but using different neighbourhoods to explore the solution space. The algorithm exploits two different neighbourhoods to intensify the search and a single diversification method:

- The `increaseTour` seeks to improve the current route by adding a node not included in any route guaranteeing that the maximum duration is not exceeded.
- The `modifyTour` tries to improve the current route by swapping a node from a route with a node not included in any route.
- Finally, after a number of not improving iterations, a restart is performed to diversify the search. For each route r_k, the restart deletes a third of the nodes $p \in r_k$. The restart procedure computes a ratio

$$\frac{s_i}{t_{pi} + t_{iq}}$$

for each node i belonging to this set (the nodes p and q represent the predecessor and the successor of the node i inside the route r_k, respectively) The diversification is guaranteed by the fact that such nodes are inserted in a tabu list, which prevents from re-inserting the node in its route of origin for a specified number of iterations. As a consequence, such nodes are not considered during the next call to `increaseTour` application.

The algorithm exploits three distinct neighbourhood sets, each designed to address a specific set of nodes by intensifying or diversifying the search:

Neighbourhoods for red patients Three neighbourhoods are introduced to intensify or diversify the search on the current solution, specifically focusing on nodes within set R.

- The first neighbourhood N_1 involves swapping two red nodes within the same route to create a new feasible route, aiming to reduce the overall route duration.
- The second neighbourhood N_2 swaps two red nodes between different tours, generating two new feasible tours with the goal of minimising the duration of each route.
- The third neighbourhood N_3 shifts a red node (along with its corresponding hospital node) from one route to another, producing a feasible solution designed to decrease the duration of each route.

All neighbourhoods apply the best feasible move strategy (best improvement). However, N_2 and N_3 also consider moves that may worsen the current solution to explore alternative paths. To prevent cycles, a couple of tabu lists are adopted as described below.

Neighbourhoods for hospitals Two neighbourhoods are developed to intensify or diversify the exploration of the current solution, specifically considering nodes in H.

- The first neighbourhood N_4 swaps a hospital node within a route with another hospital node not currently visited aiming to reduce route duration while preserving feasibility.
- The second neighbourhood N_5 randomly selects one-third of the hospital nodes visited in a route and replaces each selected node with a hospital node not currently visited.

Both neighbourhoods apply a best improvement strategy by selecting the best feasible move, even if it temporarily worsens the current solution. To prevent cycling, tabu lists are employed as outlined below.

Neighbourhoods for green patients Three neighbourhoods are introduced to either intensify or diversify the search based on the current solution, focusing on nodes within G. The neighbourhoods N_6 and N_7 are aimed at improving the overall priority score.

- The neighbourhood N_6 swaps one green node belonging to a route with one green node not visited in such a way to generate a feasible solution inserting the new node to reduce, if possible, the route duration and, in the case of z_f, inserting the new green node after all the reds.
- The neighbourhood N_7 inserts one green node not visited into a route in such a way to generate a feasible solution inserting the new node to reduce, if possible, the route duration and, in the case of z_f, inserting the new green node after all the reds.

All the neighbourhoods select the best feasible move (best improvement) even if it worsens the current one. To prevent cycles, a couple of tabu lists have been introduced: one prohibiting the insertion of a node into the solution and the other forbidding the removal of a node from the solution.

Finally, after a fixed number of not-improving iterations, the neighbourhood N_8 is applied in order to remove one-third of the nodes belonging to G and visited by one route. These node are the ones having the minimal value of the ratio

$$\frac{s_i}{t_{hi} + f_i + t_{ij}}$$

where h and j are respectively the node that precedes and that follows the node i. The rationale is to remove from the solution those patients whose marginal score is minimum to free up time in such a way to allow the visit of more profitable patients. Note that this neighbourhood is adopted as a restart procedure.

To avoid cycles during the exploration of the above neighbourhoods, two types of tabu lists are used. The first one avoids a node to come back to its starting route for a number of iterations after a move while the second one blocks one node to be moved from its destination route for a number of iterations after a move. All four tabu lists are implemented using tabu tags [44].

The pseudocode outlining the algorithmic framework shared by both approaches is presented in Algorithm 1 that makes use of the following procedures:

(i) `resetTabuLists`: resets the TLs, (ii) `SC`: applies the Spectral Clustering algorithm, (iii) `computeRoute`: computes the route from a specified cluster of nodes accordingly to the initial solution procedure, (iv) `selectNeighbourhood`: selects the move to apply according to the current solution and TLs configuration, (v) `applyMove`: applies the chosen move to the current solution, (vi) `updateTabuLists`: updates the

TLs according to the selected move and (vii) `restartSearch`: restarts the solution search accordingly to the specific algorithm.

Algorithm 1: `DSTCP and ARP algorithmic framework`

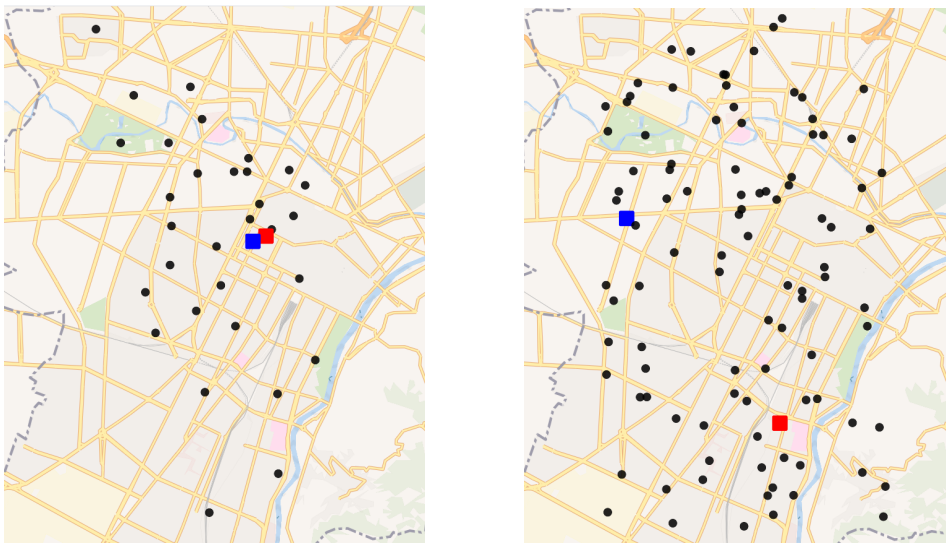
Data: m, n
Result: Solution S^*
`/* Initial solution */`
$(c_1, \ldots, c_m) \leftarrow$ `SC` (m, n); $S, S^* \leftarrow \emptyset$; $i, j, z^* \leftarrow 0$; TLs $\leftarrow$ `resetTabuList()`;
for $r \leftarrow 1$ **to** m **do**
    $v_r \leftarrow$ `computeRoute` (c_r);
    $S \leftarrow S \cup \{v_r\}$;
`/* Neighbourhood search */`
while *not* stoppingCondition(i) **do**
    move $\leftarrow$ `selectselectNeighbourhood` (S, neighbourhood, TLs);
    $S \leftarrow$ `applyMove` (S, move, TLs);
    TLs $\leftarrow$ `updateTabuLists` (move, TLs);
    if $j = \xi_{it}$ **then** $S \leftarrow$ `restartSearch` (S); $j = 0$;
    else $j \leftarrow j + 1$;
    if $z^* < z\,(S)$ **then** $S^* \leftarrow S$; $z^* \leftarrow z\,(S)$;
    $i \leftarrow i + 1$;
return S^*;

2.5 Quantitative analysis

This section presents a quantitative analysis of both problems conducted on a set of realistic instances to assess (i) the possibility of solving the problems in a practical operational context and (ii) the effectiveness and efficiency of the proposed algorithms. Firstly, a set of realistic instances has been generated for both problems in order to simulate real-world scenarios. These instances are designed to reflect practical conditions and constraints, providing a basis for evaluating the performance of the proposed algorithms. The instances take into account various factors relevant to each problem, ensuring that the analysis is grounded in realistic operational settings. Secondly, the chapter offers an in-depth evaluation of the performance of various solution methods on the newly generated instances. With this regard, a summary is provided of the analysis conducted using a general-purpose solver to solve both problems including key metrics such as computational time and solution accuracy. Following this, the focus shifts to the efficiency and quality of the solutions obtained by the proposed algorithms, with a comparison to the results from the general-purpose solver. This discussion addresses the trade-off between solution quality and computational cost, emphasising the strengths and potential limitations of each approach.

The daily swab test collection problem

A set of 54 instances has been randomly generated in order to test the impact of the optimisation on realistic scenarios based on the city of Turin, Italy, which has a surface of $130km^2$ and a population of $887,000$ inhabitants. Each instance is defined starting from (i) a set of nodes (representing the locations of the depot, the laboratory and the places), (ii) a probability distribution that assigns an integer score in $[1, 100]$ to the places, and (iii) a fixed number of teams m.

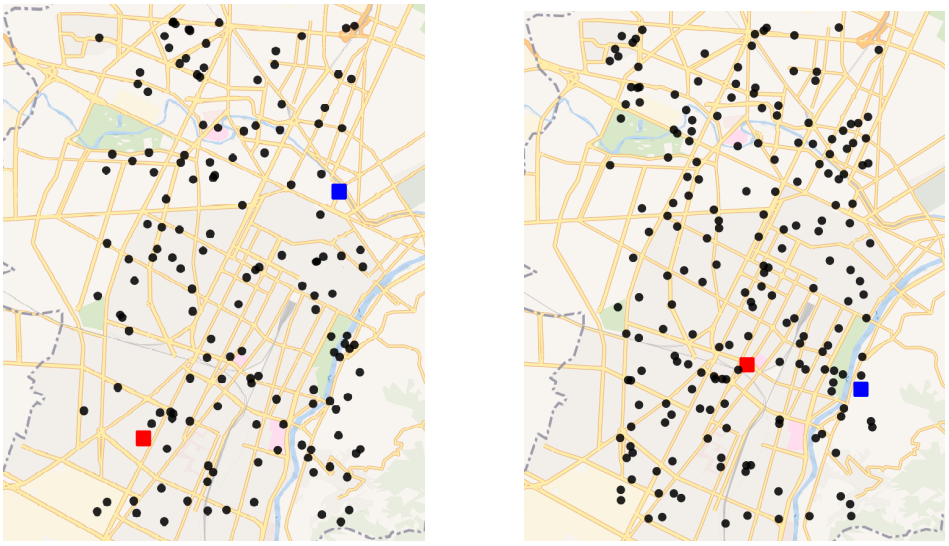


(a) Set_3 (b) Set_4

Fig. 2.5: Graphical representation of the set of nodes on the city of Turin.

Five different set of nodes Set_1–Set_5 are taken from the test instances for the TOP provided in Tsiligirides [118] and in Chao et al. [24], which are downloadable on the KU Leuven website: `www.mech.kuleuven.be/en/cib/op`. Since these sets have cardinalities ranging from 21 to 102, two additional sets of nodes, Set_6 and Set_7, with cardinalities of 150 and 200, respectively, were generated using a uniform distribution to create medium-sized instances. Finally, two further sets of nodes Set_8 and Set_9, with cardinality 907 and 2149 respectively, have been generated in order to have bigger and more challenging instances. The number of nodes of instances Set_8 and Set_9 have been computed taking into account the actual daily swab tests collected at the beginning of the second wave of the COVID-19 pandemic, when the DCT was available in Italy. The average (Set_8) and maximum (Set_9) number of regional swab tests during the period from 22nd September to 21st October 2021 were considered, multiplied by 0.2, which approximates the fraction of the inhabitants of Turin relative to the total population of the Piedmont region. Finally, the number of nodes was set to ensure that the expected number of required swab tests was half of the estimate for swab tests conducted in Turin, representing a scenario where 50% of them need to be collected at home.

All set of nodes are scaled on the area of Turin as depicted in Figures 2.5 and 2.6 in which the red and the blue squares represent the depot (source node) and the laboratory (destination node), respectively.



(a) Set_6 (b) Set_7

Fig. 2.6: Graphical representation of the set of nodes on the city of Turin.

Travelling distances are derived computing the 1-norm between each pair of nodes and considering an average speed of 20km/h. Distances were computed using the 1-norm due to the checkerboard structure of most of the city, which provides a better approximation than the classic Euclidean distance. Furthermore, for each node representing a point $p \in P$, the number of swabs b_p to be performed is generated using the distribution of households based on data from the Statistics Office of the City of Turin: the families with 1, 2, 3, 4, 5, and 6+ members are respectively the 46.99%, 26.84%, 14.12%, 9.10%, 2.13%, and 0.82%. The rationale of this choice is that each node can correspond to a family, and if a person needs a test, then all the family members could have a contact with her/him. For each node $p \in P$, a priority r_p of visiting a node has been generated using two different distributions in [1, 100]: a discretised uniform and a discretised cumulative exponential. The maximum duration $T_{\max}$ and the number of the teams/tours m have been set in such a way to guarantee that the available resources are not sufficient to collect all the swabs, fixing $t_p^s = 3$ minutes for each swab test plus $t_p^+ + 2\,t_p^h + t_p^- = 5$ minutes for the additional time spent on the node $p \in P$. For each set of nodes, the 1-tree bound [120] for the TSP was used to compute a lower bound for the travelling time required to visit all the nodes with only one team. The total service time t_p was added to the 1-tree bound value to obtain a lower bound L of the total time. Then, the value of $T_{\max}$ is fixed in such a way that it results as a multiple of 30 minutes, less than or equal to 9 hours and $m\,T_{\max} < L$. The characteristics of the 54 instances are

reported in Table 2.2 in which the letters U and E stand for uniform and exponential distributions used to generate the priority.

Tab. 2.2: Summary of the 54 instances used in the quantitative analysis.

starting set	$\|P^+\|$	$T_{\max}$	m	id	starting set	$\|P^+\|$	$T_{\max}$	m	id
Set_1	32	150	1	N1U1	Set_5	102	420	1	N5U1
			2	N1U2				2	N5U2
			3	N1U3				3	N5U3
			1	N1E1				1	N5E1
			2	N1E2				2	N5E2
			3	N1E3				3	N5E3
Set_2	21	90	1	N2U1	Set_6	150	540	1	N6U1
			2	N2U2				2	N6U2
			3	N2U3				3	N6U3
			1	N2E1				1	N6E1
			2	N2E2				2	N6E2
			3	N2E3				3	N6E3
Set_3	33	120	1	N3U1	Set_7	200	540	2	N7U2
			2	N3U2				3	N7U3
			3	N3U3				4	N7U4
			1	N3E1				2	N7E2
			2	N3E2				3	N7E3
			3	N3E3				4	N7E4
Set_4	100	360	1	N4U1	Set_4	100	360	1	N4E1
			2	N4U2				2	N4E2
			3	N4U3				3	N4E3
Set_8	907	540	10	N8U1	Set_9	2149	540	23	N9U1
			15	N8U2				35	N9U2
			20	N8U3				47	N9U3
			10	N8E1				23	N9E1
			15	N8E2				35	N9E2
			20	N8E3				47	N9E3

The analysis begins with a summary of the results obtained using a general-purpose solver to solve the DSTCP. This is followed by a discussion on the efficiency and quality of the solutions generated by the proposed algorithms. The computational tests have been performed on a standard desktop computer equipped with a Intel Core i7-8700 3.20GHz with 12 cores, and 16 GB of memory.

The integer linear program (2.2)–(2.8) has been implemented in Python adopting the Pyomo optimisation library [50] and CPLEX 12.9 as general purpose solver. With regard the computational results, the main settings of CPLEX are the default ones except for the relative mip gap tolerance which has been set to 3% and 1%. The rationale is to evaluate the running time required by a general-purpose solver to compute a high-quality solution for the DSTCP. These results are then compared with the solutions generated by the proposed algorithms, with CPLEX serving as a benchmark for comparison.

Table 2.3 reports the computational results of the general purpose solver with different relative mip gap tolerance on the first 42 instances belonging to the sets Set_1–Set_7: the columns best and secs report the value of the best integer solution

Tab. 2.3: Summary of the results obtained with CPLEX as general purpose solver.

	cplex 3%		cplex 1%			cplex 3%		cplex 1%	
id	best	secs	best	secs	id	best	secs	best	secs
N1U1	1301	0.2	1311	0.2	N4E1	3850	11.5	3850	10.2
N1U2	2207	52.0	2207	992.6	N4E2	6676	1998.9	6676	1826.4
N1U3	2782	29.1	2784	3754.8	N4E3	8755	1187.7	8747	1019.1
N1E1	1772	0.2	1772	0.2	N5U1	3793	14.1	3853	39.0
N1E2	3344	440.8	3345	5207.0	N5U2	6103	1498.3	6181	3671.9
N1E3	3668	2699.4	3668	3836.3	N5U3	7459	1185.7	7425	1176.3
N2U1	675	0.0	675	0.0	N5E1	4273	1.5	4309	9.5
N2U2	1123	0.2	1123	2.2	N5E2	7388	403.8	7445	2997.7
N2U3	1465	144.4	1465	52.5	N5E3	9198	1158.2	9273	1026.7
N2E1	837	0.0	837	0.0	N6U1	5201	33.6	5264	36.0
N2E2	1270	0.5	1270	0.5	N6U2	8632	698.8	8651	1647.6
N2E3	1680	3119.4	1680	3051.7	N6U3	10230	1920.0	10689	2093.2
N3U1	850	0.1	850	0.0	N6E1	3806	34.1	3821	287.2
N3U2	1921	0.8	1921	4.0	N6E2	6278	2650.3	6465	8281.3
N3U3	2360	4654.9	2360	4625.6	N6E3	7874	1812.2	8276	7148.1
N3E1	1124	0.2	1124	0.1	N7U2	10831	8023.3	10937	5587.6
N3E2	1513	4.8	1513	12.5	N7U3	12653	2244.5	12653	2239.5
N3E3	2586	294.8	2586	5006.3	N7U4	14408	3613.3	14135	3317.9
N4U1	3714	5.7	3714	23.8	N7E2	6834	1381.8	6801	1383.4
N4U2	5955	729.9	6046	469.5	N7E3	8810	2093.0	7866	1908.5
N4U3	7623	822.7	7644	2717.9	N7E4	10359	9867.8	10152	8225.6
	avg. time cplex 3%: 1305.5 secs					**avg. time cplex 1%: 1992.6 secs**			

computed and the running time required in seconds, respectively. The results demonstrate the increasing complexity of the problem as the number of teams m and/or the number of places n increases. For this reason, the instance with $m = 1$ is excluded from the following comparisons, as it can be easily solved. It is worth noting that the general-purpose solver encountered an "out of memory" error and stopped its computation before reaching the requested gap for an unexpected number of medium-size instances. This issue became more frequent when the mip gap was set to 1%. For the sake of completeness, the average running times of 621.6 and 825.8 are reported for the same test reported in Table 2.3 but with mip gap set to 10% and 5%, respectively.

Table 2.4 reports the comparisons between the algorithms and the general purpose solver on the instances belonging to the sets Set_1–Set_7: the columns best, secs report respectively the values of the best solution and the running time in seconds for each considered algorithm; the columns gap cplex 3% and gap cplex 1% report the relative gap of the two proposed algorithms with respect to the solution computed by CPLEX with different relative mip gap tolerance, respectively. Finally, the spectral clustering algorithm was used in the computational tests.

The results reported in Table 2.4 prove the capability of the proposed algorithms to compute good quality solutions saving a large amount of running time ranging between 88% and 93%. The gaps with respect to the objective function value of the

Tab. 2.4: Comparing the results of Clustering Search algorithms with CPLEX with a given mip gap on the instances with $m > 1$.

	init	**CS1**		**CS2**		**gap cplex 3%**		**gap cplex 1%**	
id	best	best	secs	best	secs	CS1	CS2	CS1	CS2
N1U2	2181	2181	49.6	2207	36.0	-1.2%	0.0%	-1.2%	0.0%
N1U3	2643	2709	38.4	2766	38.8	-2.6%	-0.6%	-2.7%	-0.6%
N1E2	3235	3344	58.3	3340	37.0	0.0%	-0.1%	0.0%	-0.1%
N1E3	3409	3525	52.6	3631	39.9	-3.9%	-1.0%	-3.9%	-1.0%
N2U2	951	1026	36.3	1123	15.6	-8.6%	0.0%	-8.6%	0.0%
N2U3	1175	1422	21.7	1464	20.4	-2.9%	-0.1%	-2.9%	-0.1%
N2E2	1071	1235	21.9	1270	16.1	-2.8%	0.0%	-2.8%	0.0%
N2E3	1481	1645	35.7	1680	19.4	-2.1%	0.0%	-2.1%	0.0%
N3U2	1681	1921	63.8	1921	36.9	0.0%	0.0%	0.0%	0.0%
N3U3	1995	2215	37.3	2360	44.7	-6.1%	0.0%	-6.1%	0.0%
N3E2	1234	1467	25.8	1513	33.8	-3.0%	0.0%	-3.0%	0.0%
N3E3	2336	2385	27.6	2514	44.7	-7.8%	-2.8%	-7.8%	-2.8%
N4U2	5571	5702	153.1	5939	139.3	-4.2%	-0.3%	-5.7%	-1.8%
N4U3	7258	7269	149.3	7417	150.2	-4.6%	-2.7%	-4.9%	-3.0%
N4E2	6301	6397	105.0	6660	134.9	-4.2%	-0.2%	-4.2%	-0.2%
N4E3	8382	8572	115.5	8591	153.0	-2.1%	-1.9%	-2.0%	-1.8%
N5U2	6140	6143	192.3	6140	145.2	0.7%	0.6%	-0.6%	-0.7%
N5U3	7684	7684	194.2	7684	143.2	3.0%	3.0%	3.5%	3.5%
N5E2	7074	7276	168.0	7309	145.8	-1.5%	-1.1%	-2.3%	-1.8%
N5E3	9518	9571	162.0	9592	147.4	4.1%	4.3%	3.2%	3.4%
N6U2	8348	8348	212.2	8434	224.5	-3.3%	-2.3%	-3.5%	-2.5%
N6U3	10939	10939	220.7	10939	225.3	6.9%	6.9%	2.3%	2.3%
N6E2	6132	6202	249.5	6264	213.4	-1.2%	-0.2%	-4.1%	-3.1%
N6E3	8135	8212	226.9	8238	208.8	4.3%	4.6%	-0.8%	-0.5%
N7U2	10695	10757	220.1	10821	347.6	-0.7%	-0.1%	-1.6%	-1.1%
N7U3	14057	14127	224.4	14217	345.2	11.6%	12.4%	11.6%	12.4%
N7U4	14041	15649	392.6	16581	396.8	8.6%	15.1%	10.7%	17.3%
N7E2	7094	7108	221.7	7239	312.1	4.0%	5.9%	4.5%	6.4%
N7E3	9469	9469	222.7	9476	342.0	7.5%	7.6%	20.4%	20.5%
N7E4	9481	10452	290.1	11028	353.2	0.9%	6.5%	3.0%	8.6%
			139.6		**150.4**	**-0.4%**	**1.8%**	**-0.4%**	**1.8%**

initial solution ranges between the 4.9% and the 7.2%, which prove their capability to improve the initial solution by escaping from local optima.

In terms of pure solution quality (discarding the running time), algorithm CS1 is less competitive on average with respect to the general purpose solver: while the solver is better on smaller instances, algorithm CS1 performs better on the larger ones. On the contrary, the algorithm CS2 computes on average better solution than the general purpose solver, especially on larger instances. Summing up, the gaps between the algorithms and the general purpose solver largely increases as soon as the complexity of the instances increases in terms of number of places and/or number of teams. For the sake of completeness, the comparison of the algorithms with the general purpose solver with mip gap set to 10% and 5% showed that the two algorithms compute better solutions than the solver: about 2.8% and 5.0% (mip gap set to 10%) and 0.7% and 2.9% (mip gap set to 5%). Although not explicitly

listed, it is notable to remark that the algorithm CS1 and CS2 perform very well also in the instances with $m = 1$.

What emerges from the proposed analysis is the fact that the instances whose scores are generated by a discretised uniform distribution are easier to solve than those with a score generated by a discretised cumulative exponential distribution. This fact can be explained considering that the former distribution generates scores that are more spread in the interval $[1, 100]$ with respect to the latter, which concentrates the scores in the values closest to 100. Accordingly, solutions of instances with uniform scores are more heterogeneous than the ones with cumulative exponential scores, which could lead to symmetry issues.

Table 2.5 reports the computational on the larger instances belonging to the sets Set_8 and Set_9: the columns best, secs report respectively the values of the best solution and the running time in seconds for each considered algorithm; the columns gap init and gap CS1 report the relative gap of the two proposed algorithms with respect to the solution computed by the initial solution procedure and the CS1 algorithm, respectively. It is important to note that CPLEX results are not reported, as it was unable to compute a good solution within a reasonable running time.

Tab. 2.5: Comparing the results of neighbourhood search algorithms on the larger instances.

	init	**CS1**		**CS2**		**CS1**	**CS2**	
id	best	best	secs	best	secs	gap init	gap init	gap CS1
N8E1	25937	58082	2378	65387	267	123.93%	152.10%	12.58%
N8U1	21660	47156	3741	49650	268	117.71%	129.22%	5.29%
N8E2	26202	74438	3024	81897	277	184.09%	212.56%	10.02%
N8U2	22073	58988	7376	63598	329	167.24%	188.13%	7.82%
N8E3	27777	83485	5457	97551	340	200.55%	251.19%	16.85%
N8U3	22567	65272	10312	74572	338	189.24%	230.45%	14.25%
N9E1	44785	129892	17285	148674	1549	190.03%	231.97%	14.46%
N9U1	38188	108222	18981	116239	1576	183.39%	204.39%	7.41%
N9E2	46899	142753	25912	189598	1628	204.38%	304.27%	32.82%
N9U2	39581	105981	24841	149814	2060	167.76%	278.50%	41.36%
N9E3	49968	156584	28459	222044	3095	213.37%	344.37%	41.81%
N9U3	41286	125468	31587	175937	2273	203.90%	326.14%	40.22%
			14946		**1167**	**178.80%**	**237.77%**	**20.41%**

The results reported in Table 2.5 prove the better efficiency and quality of algorithm CS2 with respect to CS1: as a matter of fact, CS2 is capable to compute, on average, better solutions (about 20.41%) than CS1 in less running time (CS2 is about ten times faster than CS1). Furthermore, the bigger gaps with the initial solution method confirm their capability to improve the initial solution by escaping from local optima.

The ambulance routing problem

A total of 24 benchmark instances have been generated exploiting the instance generator presented in Aringhieri et al. [8], which has been modified in such a way to partition the nodes into three different types (hospital, green patient and red patient) and to label: (i) the green patients with a score in 5, 10, 15 with uniform probability, (ii) all patients with a service time (in minutes) generated with a uniform distribution in [5, 35] for the green patients and in [2, 15] for the red patients, (iii) all hospitals with a fixed service time of 10 minutes. Coordinates have been scaled considering a square area of $900\,km^2$, then travelling times have been computed through the Euclidean distance between nodes and considering an average speed of 50 km/h.

Tab. 2.6: Characteristics of the realistic benchmark instances.

id	$\|N\|$	$\|G\|$	$\|R\|$	red distr.	$\|O\|$	$\|D\|$	$\|K\|$	$T_{\max}$ (min)
P10C3O1	18	7	3	clustered	1	5	1	248
P10S3O1	18	7	3	scattered	1	5	1	248
P10C4O1	19	6	4	clustered	1	6	1	315
P10S4O1	19	6	4	scattered	1	6	1	315
P10C3O2	26	7	3	clustered	2	12	2	135
P10S3O2	26	7	3	scattered	2	12	2	135
P10C4O2	28	6	4	clustered	2	14	2	135
P10S4O2	28	6	4	scattered	2	14	2	135
P25C6O2	54	17	8	clustered	2	25	3	203
P25S6O2	54	17	8	scattered	2	25	3	203
P25C8O2	58	15	10	clustered	2	29	3	225
P25S8O2	58	15	10	scattered	2	29	3	225
P25C6O3	70	17	8	clustered	3	40	4	167
P25S6O3	70	17	8	scattered	3	40	4	167
P25C8O3	76	15	10	clustered	3	46	4	169
P25S8O3	76	15	10	scattered	3	46	4	160
P50C12O3	120	35	15	clustered	3	65	5	203
P50S12O3	120	35	15	scattered	3	65	5	180
P50C16O3	135	30	20	clustered	3	80	5	248
P50S16O3	135	30	20	scattered	3	80	5	257
P50C12O4	146	35	15	clustered	4	90	6	158
P50S12O4	146	35	15	scattered	4	90	6	158
P50C16O4	166	30	20	clustered	4	110	6	225
P50S16O4	166	30	20	scattered	4	110	6	212

Eight instances were generated for three different numbers of patients: 10, 25, and 50. The size of the instances matches those tested in Talarico et al. [113]. Each instance varies based on the coordinates of the nodes, the distribution of red patients (scattered or clustered), the number of hospitals, and the number of ambulances. For each instance, the lower bound of the travelling time required to visit all the nodes with a single ambulance is calculated using the same procedure applied in the DSTCP.

All service times are then added to the 1-tree bound value to obtain a lower bound L for the total time. A value within the range $[0.8\,L, 1.2\,L]$ is then selected to ensure feasible but non-trivial solutions (e.g., the available time is sufficient to serve all the red patients but not large enough to easily visit all the green ones). The characteristics of the instances are summarised in Table 2.6, which highlights how the graph's size increases with the number of ambulances, hospitals, and red patients, due to the number of dummy hospital nodes, given by $|D| = |K| + |O|\,(|K| + |R|)$.

The analysis begins with a summary of the results obtained using a general-purpose solver to solve the mathematical model with the hierarchical objective functions z_f (2.24) and z_e (2.25). This is followed by a discussion on the efficiency and quality of the solutions generated by the proposed algorithm. Finally, a comparison is provided between the z_f and z_e solutions to assess the so-called price of fairness Nicosia et al. [90].

All the computational tests have been performed on a standard desktop computers equipped with a Intel Core i7-8700 3.20GHz with 12 cores, and 16 GB of memory. The integer linear program 2.9–2.16 resulting from the discussion reported in Section 2.3. has been implemented adopting the OPL language and solved with CPLEX 12.9 with default settings. It is important to remark that the default settings use all the cores available reducing the overall running time instead of using only one core. The algorithm has been implemented in Python.

Tab. 2.7: CPLEX: computational results over the benchmark instances with 10 patients.

	fairness					**efficiency**				
	z_f	$C_{\max}$	secs	z_e	σ_G	z_e	σ_G	secs	z_f	$C_{\max}$
P10C3O1	11987	182	7.5	6043	25	10959	45	0.4	16191	246
P10S3O1	7698	138	19.8	7332	30	10961	45	0.3	13619	244
P10C4O1	15509	219	96.7	12421	40	17077	55	0.2	21458	303
P10S4O1	14676	241	11.4	7659	25	12335	40	0.2	18565	305
P10C3O2	5783	88	294.9	3312	25	6679	50	15.9	7936	121
P10S3O2	4063	73	383.5	3327	25	5986	45	13.5	7459	134
P10C4O2	6218	88	667.2	3992	30	7355	55	12.9	8820	125
P10S4O2	6659	94	229.4	1946	15	7353	55	18.8	8962	127

Table 2.7 reports the computational results obtained with CPLEX on the first 8 instances, for which the optimal solution has been computed. The table contains two sets of columns. The first set (under the name *fairness*) reports the values z_f and $C_{\max}$ of the solution computed, and the running time in seconds; then the columns z_e and σ_G report the efficiency values computed on the fairness solution. On the contrary, the second set (under the name *efficiency*) reports the values z_e and σ_G of the solution computed, and the running time in seconds; then the columns z_f and $C_{\max}$ report the fairness values computed on the efficiency solution. The columns have the same meaning also in Table 2.8 and 2.9.

A consistent difference in running time is observed: computing a fairness solution requires ten times the running time needed to compute an efficient one. Similar running time differences are noted for the larger instances reported in Table 2.6. Furthermore, several hours are often required to find a sub-optimal solution for larger instances. Since the goal is to provide a prompt decision in post-disaster management situations, this highlights the need for the solution algorithm presented in Section 2.4.1.

Tab. 2.8: Algorithm $\mathcal{A}$: computational results over the benchmark instances with 10 patients.

	fairness					**efficiency**				
	z_f	$C_{\max}$	secs.	z_e	σ_G	z_e	σ_G	secs.	z_f	$C_{\max}$
P10C3O1	**11987**	182	3.8	6043	25	**10959**	45	4.0	16191	246
P10S3O1	**7698**	138	4.0	7332	30	**10961**	45	4.1	13619	244
P10C4O1	**15509**	219	4.7	12421	40	**17077**	55	4.9	21458	303
P10S4O1	**14676**	241	4.4	7659	25	**12335**	40	4.6	18565	305
P10C3O2	6235	95	6.0	4665	35	**6679**	50	5.1	7936	121
P10S3O2	**4063**	73	6.0	3327	25	**5986**	45	4.3	7459	134
P10C4O2	**6218**	88	5.0	3992	30	**7355**	55	4.2	8820	125
P10S4O2	7567	107	5.2	3973	30	**7353**	55	4.7	8962	127

Table 2.8 reports the computational results of the proposed solution algorithm where the numbers in bold highlight the optimal solutions with respect to the optimal values reported in Table 2.7. From a comparison with the values reported in Table 2.7, the results in Table 2.8 prove the quality of the solution computed by the algorithm $\mathcal{A}$: actually, the algorithm $\mathcal{A}$ replicates all the optimal values in the efficiency case, and 6 over 8 in the fairness case. Furthermore, the results prove the efficiency of the proposed algorithm, which is capable of obtaining such results in a few seconds instead of tens (efficiency) or hundreds (fairness) of seconds.

In order to evaluate the impact of each neighbourhood on the quality of the final solutions z_f and z_e, we solved the ten instances reported in Table 2.7 temporarily disabling one of the eight neighbourhoods resulting in 160 new solutions, which are then compared with the results reported in Table 2.8.

When dealing with the fairness, we observed significant variations in z_f (i.e., at least 0.5%) only for N_2 and N_6. We also consider the value of z_e computed on the fairness solution. Without N_2, the value of z_f worsens of the 6.02% while no significant variations are reported in the value of z_e. On the contrary, no significant variation of z_f are reported without N_6 while the value of z_e worsens of the 3.29%.

When dealing with the efficiency, we observed no significant variations in z_e (i.e., at least 0.5%) temporarily disabling one of the eight neighbourhoods. On the contrary we reported a significant worsening of the value of z_f computed on the efficiency solution ranging from 0.62% to 0.98%.

We performed a further test temporarily disabling all the neighbourhoods working on a specific component of the solution, that is (i) the neighbourhoods for red patients N_1, N_2 and N_3, (ii) the neighbourhoods for hospitals N_4 and N_5, and (iii) the neighbourhoods for green patients N_6, N_7 and N_8. The results clearly get worse in all the tests. We observed a peak of 6.31% for z_f when dealing with the fairness and disabling N_1, N_2 and N_3. On the contrary, we observed a peak of 10.58% for z_e when dealing with the efficiency and disabling N_6, N_7 and N_8.

Tab. 2.9: Algorithm $\mathcal{A}$: results over the benchmark instances with 25 and 50 patients.

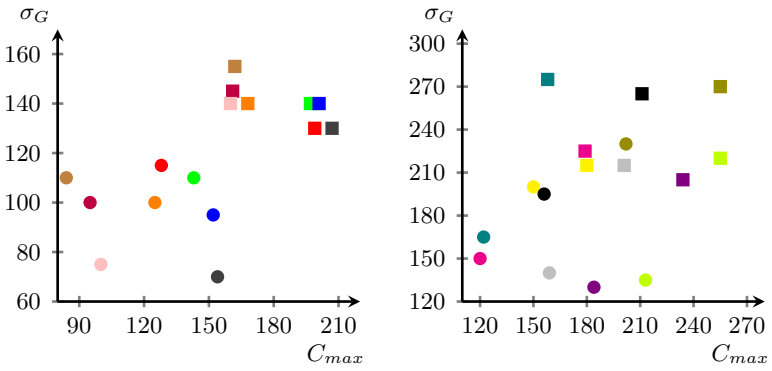
	fairness					**efficiency**				
	z_f	$C_{\max}$	secs	z_e	σ_G	z_e	σ_G	secs	z_f	$C_{\max}$
P25C6O2	22413	128	21.6	23217	115	26191	130	21.2	34894	199
P25S6O2	26488	143	23.7	22187	110	28223	140	25.7	36502	197
P25C8O2	25657	152	28.2	21852	95	28654	140	24.4	35435	201
P25S8O2	26264	154	32.3	14756	70	27483	130	25.1	35267	207
P25C6O3	16620	95	28.5	16205	100	23474	145	25.9	28191	161
P25S6O3	15514	84	26.9	17846	110	25103	155	26.6	29977	162
P25C8O3	20025	125	39.7	16875	100	23632	140	34.2	26908	168
P25S8O3	15025	100	37.7	11975	75	22380	140	29.3	24020	160
P50C12O3	56464	159	102.7	28261	140	43444	215	88.7	71341	201
P50S12O3	42570	120	103.3	27030	150	40546	225	88.4	63499	179
P50C16O3	55254	184	186.5	30366	130	47941	205	133.6	70229	234
P50S16O3	65043	213	181.4	34482	135	56285	220	146.6	77810	255
P50C12O4	53200	150	121.2	36050	200	38735	215	105.3	63865	180
P50S12O4	43267	122	138.0	26113	165	43567	275	108.5	55973	158
P50C16O4	61582	202	197.8	58908	230	69135	270	159.5	77760	255
P50S16O4	46761	156	214.9	41379	195	56234	265	176.7	63246	211

Table 2.9 reports the computational results of the algorithm over the remaining benchmark instances, that is those with 25 and 50 patients. The running time of the algorithm proves its capability of dealing with larger and more realistic instances. Fairness-oriented solutions seem harder to compute than efficiency-oriented solutions, at least in terms of running time.

A preliminary comparison between the two kinds of solutions can be done as follows. Moving from an efficient solution to a fair one, it is possible to measure the benefit in terms of fairness gained and efficiency lost. Vice versa, we can measure the benefit in terms of efficiency gained and fairness lost. In the former case, the average fairness gained is about 27.71% while the efficiency lost is about 29.87%. In the latter case, the average efficiency gained is about 46.10% while the fairness lost is about 40.63%. These results seem to prove that there is no dominance between the two kinds of solutions.

No significant difference emerges from dealing with instances with a different distribution of the red patients over the disaster area: similar completion times have been obtained for such patients on the clustered and scattered version of the same instances. For example, the instance P25C6O2 has a slightly lower value of C_{max}

with respect to P25S6O2, but the opposite occurs comparing the instances P25C6O3 and P25S6O3. In order to better appreciate the trade off between C_{max} and σ_G provided by the two different objective functions z_f and z_e, we plot the solution obtained for the instances with 25 and 50 patients in Figure 2.7.



(a) 25 patients (b) 50 patients

Fig. 2.7: Comparing the solution provided by the algorithm $\mathcal{A}$ using z_f (circles) and z_e (squares) as objective functions (same color is used for the two solutions of the same instance).

Both fairness and efficiency solutions provided for the same instances are far from dominating each other. Furthermore, smaller instances show a clear separation between fairness and efficiency solutions, regardless of how patients are distributed over the disaster area. Such a separation vanishes as the number of patients to be visited increases, but without loosening the strong trade off between the two objectives.

From the reported analysis that can be seen as a sort of Pareto analysis, it is evident that no clear dominance between the fairness and efficiency solutions has been identified. From an application perspective, this empirical result makes clearer the meaning of *price of fairness*: actually, to improve the efficiency of a fairness solution, it is necessary to allocate more time resources to serve the same green patients served by the efficient solution.

2.6 Conclusions

Digital contact tracing applications are important for managing infectious diseases like COVID-19 but their success rely on robust logistical support. Timely testing of identified contacts become critical during high transmission periods. The DSTCP highlights the role of operations research by combining tracing technologies with effective mathematical optimisation tools. Similarly, disaster response systems emphasize the importance of tailored strategies. Ambulance optimisation in post-disaster scenarios, as explored in Talarico et al. [113], seeks to balance efficiency

and equity by prioritising first aid for moderate cases while minimising delays for critical patients. These insights contribute to the design of systems that ensure fair and effective emergency care during crises.

From the generalisation of the healthcare applications presented in this chapter, the TOP-ST-MIN can be formulated as a new variant of the TOP that integrates all the features previously introduced. To the best of current knowledge, these features have never been considered together in a single optimisation problem as reported in Chapter 3. As discussed in Chapter 5, the TOP-ST-MIN results to be more challenging than the TOP because the problem of finding a feasible solution has been proven to be NP-Complete. The study reported for the ARP highlights the importance of considering a fairness perspective when evaluating the quality of a solution. For this reason, Chapter 8 provides an in-depth analysis from a fairness perspective.

Literature review

3

The literature review in this chapter provides an overview of the existing studies relevant to the TOP-ST-MIN. This review explores key contributions from both deterministic and uncertain perspectives, examining a range of approaches and methodologies. The deterministic approach emphasises mathematical models with both heuristic and exact solution methods, taking into account the three distinctive features of the problem: service times, mandatory nodes and incompatibilities. Dynamic and stochastic considerations, including the effect of travel or service time uncertainty, are also explored. Finally, this literature review also addresses the notion of fairness defined as the fair resources allocation among the agents working in a shared environment.

3.1 The deterministic perspective

Over the past decades, the field of orienteering problems has known significant growth, driven by their appeal to the research community and their practical applications. This section reviews the literature on the problems related to the TOP-ST-MIN by focusing on the deterministic aspects of the problem. The review explores several studies that present mathematical formulations and solution methods considering these features.

3.1.1 The three features of the problem

This part of the literature review examines three key features of the TOP-ST-MIN exploring recent advancements and methodologies. Table 3.1 summarises the key studies identified highlighting their general characteristics (denoted by 1 to 13) and the features specific to the TOP-ST-MIN (denoted by A to E).

In the context of the TOP-ST-MIN, the most prevalent feature identified is the service times (both fixed and variable) at nodes followed by mandatory nodes and logical incompatibilities. Many of the examined studies focus on the classic Tourist Trip Design Problem (TTDP), where a tourist must visit a subset of points of interest (POIs). Each POI is characterised by a profit, a service time and specific time windows for visitation. Regarding solution methods, the literature shows a predominance of metaheuristic techniques, with fewer applications of exact algorithms. Additionally, occasional use of matheuristic and specialised heuristic approaches has been reported. The sole work that groups all the features introduced (except the physical incompatibilities) is Vu et al. [124], which addresses a variant of the TTDP that

considers additional constraints such as logical incompatibilities and mandatory nodes. Notably, papers addressing variable service times (excluding Gündling and Witzel [47]) or logical incompatibilities focus exclusively on variants of the OP.

Tab. 3.1: Summary of the most recent contributions showcasing the features shared with the TOP-ST-MIN.

		Service Times		Mandatory	Incompatibilities	
	Reference	**Fixed**	**Variable**	**Nodes**	**Physical**	**Logical**
OP	Palomo-Martínez et al. [94]			✓		✓
	Palomo-Martínez et al. [93]			✓		✓
	Liao and Zheng [72]	✓				
	Lu et al. [79]			✓		✓
	Taylor et al. [115]	✓		✓		
	Yu et al. [129]		✓			
	Dutta et al. [33]	✓				
	Sebastia and Marzal [106]		✓			
	Wallace et al. [125]		✓			
	Zheng et al. [132]	✓				
	Nodarse et al. [91]	✓				
	Vu et al. [124]	✓		✓		✓
	Morandi et al. [85]		✓			
TOP	Kotiloglu et al. [65]	✓		✓		
	Lin and Yu [76]	✓		✓		
	Reyes-Rubiano et al. [98]	✓				
	Assunção and Mateus [11]			✓		
	Jin and Thomas [57]	✓				
	Stavropoulou et al. [109]	✓		✓		
	Thompson and Galeazzi [117]	✓				
	Gündling and Witzel [47]		✓			
	Hanafi et al. [49]	✓				
	Wisittipanich and Boonya [127]	✓				
	Aringhieri et al. [6]	✓		✓	✓	
	Flushing et al. [39]	✓				
	Heris et al. [54]	✓				
	Ruiz-Meza et al. [101]	✓				
	Ruiz-Meza and Montoya-Torres [102]	✓				
	Yu et al. [128]	✓				
	Corrêa et al. [26]	✓		✓		
	Li et al. [70]	✓				
	Aringhieri et al. [5]	✓				

To complete the analysis, additional studies that examine various extensions of the TOP have been considered. A significant number of these works focus on the integration of time-windows, particularly to model specific real-world applications of the TTDP, as can be seen in studies such as Vu et al. [124] and Ruiz-Meza et al. [101]. These contributions highlight the importance of temporal constraints in route planning problems. In contrast, fewer works address less common features. For instance, time-dependency constraints, where travel times may vary depending on departure times at locations, are considered in Nodarse et al. [91], while precedence constraints, requiring certain nodes to be visited before others, are discussed in Sun et al. [110]. Finally, capacity constraints, which impose limits on the resources or load a vehicle can carry, are addressed in Wang et al. [126].

The service time at nodes

Originally, the basic OP and TOP framework does not consider any service time at each node. However, many problems in healthcare, tourism, and in general everything related to a routing problem, may need to consider a certain amount of waiting time at each node to model a specific service.

The first related work involving this feature has been proposed by Erdogan and Laporte [35], where the authors introduced the Orienteering Problem with Variable Profits. This problem is an extension of the classic orienteering problem in which the full profit from the customers can be collected either by spending a continuous amount of time at the customers locations or by making a discrete number of passes. The objective is to design a tour that maximises the profit without exceeding a specified travel time limit.

The TTDP has been studied intensively becoming the most famous application of the OP. In Lim et al. [73], the authors proposed the `PersTour` algorithm to recommend personalised routes based on the popularity of points of interest (POIs) and user interest preferences. Such information have been derived from real-world travel patterns exploiting geo-tagged photographs. The problem is formulated as an OP with time windows and variable service times. User interest is quantified through visit duration, allowing for personalised POI visit times tailored to this interest measure.

As reported in Yu et al. [129], the Orienteering Problem with Service Time Dependent Profits involves collecting profits at each vertex, where the profit is determined by a nonlinear function of service time. The objective consists in maximising the total profits by selecting a subset of vertices to visit and assigning an appropriate service time to each within a given time budget. To tackle this problem, the authors developed a mixed-integer nonlinear programming model and a two-phase matheuristic algorithm. The results indicate that the proposed matheuristic algorithm is highly effective in finding high-quality solutions.

The idea of the TTDP has been extended to account for multiple routes spanning over several days. Specifically, Zheng et al. [132] sought to create personalised itineraries with integrated hotel selection for multi-day urban tourists. They developed a linear model framing the problem as a variant of the TOP, aiming to identify the optimal sequence of daily trips. Each day's route begins at one hotel and concludes at another, linking multiple daily paths into a continuous, overarching itinerary.

Li et al. [70] investigated a variation of the TOP in which each vertex requires a service time and two visits. In this problem the profit collected from each node depends on the interval between these two visits. Two variants are considered: the periodic OP, which focuses on the number of days and the TOP with a combination of time windows. To address these variants, the authors developed a mixed-integer linear

programming model for each of them in combination with two exact Branch-Price-Cut algorithms designed to exploit their block diagonal structures. Two strategies are introduced in order to enhance the algorithms performance: a simplification strategy that streamlines the directed graph network by eliminating redundant vertices and arcs without affecting optimality and a matheuristic approach for obtaining integer solutions more quickly. Computational experiments demonstrate the effectiveness of the proposed algorithms when combined with the simplification strategy achieving a consistent average computational time reduction compared to the basic approach. The matheuristic algorithm further confirms its efficiency in generating high-quality integer solutions.

The mandatory nodes

In a real application context, it could be necessary to visit some critical or relevant subset of nodes. In a tourism context, all tourists may share some common interests, which the route planner may impose as mandatory POIs. During an emergency situation, the ambulances have to collect all the patients that suffer of severe pathologies or injuries from a serious accident. For this reason, in a mathematical model, these patients could be modelled as mandatory nodes in a graph.

The earliest work identified on mandatory nodes was proposed by Kotiloglu et al. [65], in which the authors addressed the complex task of generating personalised route recommendations for tourists. The objective consists in creating routes that include all mandatory POIs while maximising the total profit from optional points visited each day. The problem also considers factors such as daily availability, opening hours, tour length limits, time budgets, and other constraints.

Taylor et al. [115] formalised the concept of mandatory nodes introducing a variant of the TTDP that incorporates mandatory POIs, named the Tour Must See Problem. Essentially, this problem is a variant of the OP where a subset of mandatory nodes must be visited. The authors developed an integer linear model to address the problem. Furthermore, Assunção and Mateus [11] introduced a new variant of the TOP called the Steiner Team Orienteering Problem. This problem has the same objectives as the TOP, with the distinction that it includes a specified subset of mandatory nodes that must be visited. The authors proposed an innovative commodity-based formulation for the problem and solved it with a cutting-plane approach.

In the context of transportation applications, Stavropoulou et al. [109] introduced a new mathematical model known as the Consistent Vehicle Routing Problem with Profits. This problem features two types of customers: the frequent customers (classified as mandatory) and the non-frequent (or potential) customers associated with known and estimated profits. Both groups have predefined demands and service requirements over a multi-day planning horizon. The objective is to identify a set of

routes that maximise the profit without violating the constraints related to vehicle capacity, route duration, and consistency. To address this problem, the authors developed an adaptive tabu search algorithm that incorporates both short and long term memory structures to enhance the search process.

A new variant of the TOP was introduced by Lin and Yu [76] who examined the TOP with time windows and mandatory visits. In this variant, certain customers are designated as important and must be visited, while others are classified as optional customers. Each customer generates a positive profit. The objective is to identify a specified number of routes that maximise the total profit collected from the visited nodes respecting the mandatory visits and time window intervals. The authors formulated a mathematical linear model and developed a multi-start simulated annealing metaheuristic to solve the problem.

The incompatibilities between nodes

As far as we know, the concept of physical incompatibilities had not been addressed in the literature until the work by Aringhieri et al. [6]. As reported in the study, it is possible to model the requirement of transporting injured patients to hospitals by imposing physical incompatibilities between nodes. On the other hand, the logical incompatibilities can be used to separate several types of patients. For example, patients with different infectious diseases can not share the same ambulance.

The first identified study that addresses logical incompatibilities is Palomo-Martínez et al. [94]. The authors examined a variant of the OP that incorporates mandatory visits and incompatibilities among nodes. They developed a search algorithm that combines a greedy randomised adaptive procedure with a variable neighbourhood search approach. Additionally, they validated the performance of such an algorithm using several instances drawn from the literature on the traditional OP.

Lu et al. [79] introduced another variant of the OP known as the OP with Mandatory Visits and Exclusionary Constraints. The goal is to find a route that visits a specified set of mandatory nodes along with some optional nodes in such a way to respect some logical incompatibilities among nodes and a maximum total time budget constraint. To solve this problem, the authors developed a powerful memetic algorithm that incorporates (i) a specialised tabu search procedure capable of exploring both feasible and infeasible solutions, (ii) a backbone-based crossover method, and (iii) a randomised mutation procedure to mitigate premature convergence.

Another study that incorporates the concept of logical incompatibilities is Vu et al. [124]. The authors enriched the TTDP by introducing additional constraints: (i) the inclusion of mandatory locations, (ii) the categorisation of each POI, (iii) restrictions on the number of locations that can be visited from each category, and (iv) the sequence in which selected locations are visited, which introduces incompatibilities.

A Branch & Check algorithm is proposed for the solution of this problem. In their approach, the reduced problem identifies a subset of POIs that verifies all constraints except those related to time. Each subset of POIs constitutes a candidate solutions for which the sub-problem decides whether a feasible trip can be constructed using the specified locations.

3.1.2 Solution approaches

This part of the literature review focuses on the solution approaches for the orienteering problems that share at least one of the three features that characterise the TOP-ST-MIN, providing a detailed examination of recent developments in this area. The most recent of these works are listed in Table 3.2.

Tab. 3.2: Overview of the latest contributions on solution approaches, highlighting the features they share with the TOP-ST-MIN. The evaluated features are: (A) Fixed Service Times, (B) Variable Service Times, (C) Mandatory Nodes, (D) Physical Incompatibilities, and (E) Logical Incompatibilities.

	Reference	Heuristic	Metaheuristic	Matheuristic	Exact	(A)	(B)	(C)	(D)	(E)
OP	Palomo-Martínez et al. [94]		✓					✓		✓
	Wang et al. [126]	✓	✓			✓				
	Zheng et al. [133]		✓			✓				
	Taylor et al. [115]				✓	✓		✓		
	Liao and Zheng [72]		✓			✓				
	Lim et al. [73]	✓				✓				
	Lu et al. [80]		✓				✓		✓	
	Sun et al. [110]	✓				✓				
	Márquez et al. [87]		✓			✓				
	Yu et al. [129]			✓			✓			
	Dutta et al. [33]		✓			✓				
	Zheng et al. [132]		✓			✓				
	Vu et al. [124]				✓	✓		✓		✓
	Morandi et al. [85]				✓		✓			
TOP	Aghezzaf and Fahim [1]		✓			✓				
	Kotiloglu et al. [65]		✓			✓		✓		
	Lin and Yu [77]		✓			✓		✓		
	Sylejmani et al. [111]		✓			✓				
	Reyes-Rubiano et al. [98]		✓			✓				
	Jin and Thomas [57]		✓			✓				
	Assunção and Mateus [11]				✓			✓		
	Stavropoulou et al. [109]		✓			✓		✓		
	Thompson and Galeazzi [117]		✓			✓				
	Gündling and Witzel [47]		✓				✓			
	Hanafi et al. [49]			✓	✓	✓				
	Wisittipanich and Boonya [127]		✓			✓				
	Flushing et al. [39]			✓		✓				
	Heris et al. [54]		✓			✓				
	Ruiz-Meza et al. [101]		✓			✓				
	Yu et al. [128]				✓		✓			
	Aringhieri et al. [7]		✓			✓		✓		
	Aringhieri et al. [5]		✓			✓				
	Corrêa et al. [26]		✓			✓		✓		
	Yu et al. [130]		✓			✓				
	Li et al. [70]			✓	✓	✓				

The most frequent feature found consists in the presence of service times (fixed and variable) at each node, followed by the mandatory nodes and incompatibilities. Most solution methods rely on metaheuristic approaches with additional contributions from matheuristics, tailored heuristics and exact approaches.

The metaheuristic approaches

As it appears from Table 3.2, the majority of the OP and TOP solution methods considered are based on the metaheuristic framework because they are efficient and effective in exploring the search space.

The first study addressing metaheuristics was introduced by Labadie et al. [68]. The authors proposed a metaheuristic approach for the Capacitated Team Orienteering Problem with Time Windows. In this problem, a set of customers is defined, each with specific demand, profit, service time and time window. The goal consists in finding a set of vehicle routes that maximises the total profit collected while respecting to constraints related to time windows, vehicle capacity, and the maximum duration for each route. To tackle the problem, the authors developed a variable neighborhood search algorithm that primarily explores granular neighborhoods to enhance the algorithm's efficiency without sacrificing effectiveness.

The Mixed Team Orienteering Problem with Time Windows was introduced in Gavalas et al. [42]. This study essentially outlines a practical application of the TOP with time windows in the context of asymmetric graphs. This problem represents a variant of the TTDP with the objective of deriving near-optimal multi-day itineraries for tourists visiting various points of interest (POIs). The authors developed two metaheuristic approaches based on iterated local search and simulated annealing techniques.

Taking into account mandatory nodes and logical incompatibilities, Lu et al. [80] introduced a highly effective memetic algorithm for the Orienteering Problem with Mandatory Visits and Exclusionary Constraints. The objective of this problem is to visit a specified set of mandatory locations with some optional ones respecting the logical incompatibilities (exclusionary constraints) between nodes and the overall maximum time budget constraint. The algorithm integrates a specialised tabu search procedure that evaluates both feasible and infeasible solutions through constraint relaxation, alongside a backbone-based crossover and a randomised mutation procedure to avoid premature convergence.

Lastly, in Yu et al. [130], the authors present the Set Team Orienteering Problem with Time Windows, a new variant that combines elements from the well-known Team Orienteering Problem with Time Windows and the Set Orienteering Problem. In this problem, customers are organised into clusters, with each cluster associated with a profit that can be obtained by visiting any customer within its designated time window. To address this problem, a Simulated Annealing with Reinforcement Learning algorithm has been developed.

The matheuristic approaches

Another popular approach to tackle combinatiorial optimisation problems is the matheuristic framework, which sometimes results to be more effective yet slower

than metaheuristic approaches since it requires to manage mathematical optimisation models. As can be seen from Table 3.2, there are fewer matheuristic approaches compared to metaheuristic approaches.

The first study addressing matheuristics was proposed by Yu et al. [129]. The authors introduced the Orienteering Problem with Service Time-dependent Profits, where the profit earned at each vertex is defined by a non-linear function of the service time. The objective of this problem is to maximise total profits by selecting a subset of vertices to visit and assigning suitable service times to each, all within a specified time budget. The authors proposed a mixed-integer nonlinear programming model and developed a two-phase matheuristic approach that integrates a tabu search method with a nonlinear programming model in order to optimise service time allocation.

Hanafi et al. [49] presented a matheuristic approach to address a variant of the TOP known as the Multi-visits Team Orienteering Problem with Precedence Constraints. In this problem, each customer has a series of tasks to be performed in a specified order by a heterogeneous fleet of vehicles. If a customer is chosen, all tasks must be completed, although they do not have to be executed by the same vehicle. To solve this problem, the authors propose an enhanced version of the Kernel Search (KS) framework that employs various sorting strategies and compare its performance with a branch-and-cut algorithm that incorporates dynamic separation of different valid inequalities and utilises a simplified KS as a primal heuristic.

The most recent study found employing a matheuristic approach is Flushing et al. [39]. In this paper, the authors tackle the issue of mission planning for teams of mobile autonomous agents. They analysed spatially distributed tasks that provide rewards for both full and partial completion. The objective is to develop a system-level plan that assigns tasks to agents in order to maximise mission performance. The authors addressed the mission planning problem adopting a model that considers several sub-problems: task selection and allocation, task scheduling, task routing and managing agent proximity over time. They formalised this as a mixed-integer linear program. To enhance efficiency, the authors proposed a matheuristic approach that integrates a generic mathematical solver with a genetic algorithm.

The tailored heuristic approaches

Unlike the metaheuristic and matheuristic approaches, the tailored heuristic approaches are often very efficient but less effective because they do not reevaluate every decision they take. As can be seen from Table 3.2, there are only a few tailored heuristic methods.

A new multi-agent variant of the OP, referred to as the Multi-Agent Orienteering Problem with Capacity Constraints, was introduced by Wang et al. [126]. Unlike

existing OP variants, this problem permits a group of visitors to visit a node simultaneously while imposing a limit on the number of visitors that can be served at each node at the same time. Determining an optimal solution for this problem can become prohibitive as the number of vertices and agents increases, the authors proposed an efficient heuristic sequential algorithm to address this challenge.

Lim et al. [73] developed a heuristic algorithm for recommending personalised tours by leveraging POI popularity and user interest preferences, which are automatically derived from actual travel sequences based on geotagged photographs. The recommendation problem is addressed using a formulation of the OP and incorporates user trip constraints, such as time limits and the requirement to start and end at specific POIs. Additionally, the authors took into account varying levels of user interest based on visit durations.

Finally, Sun et al. [110] introduced the Time-Dependent Capacitated Profitable Tour Problem with Time Windows and Precedence Constraints. The goal of this problem is to determine a tour and its departure time from the depot that maximises the profit collected, subtracting the total travel cost (which is measured by total travel time). In order to account for road congestion, travel times are treated as time-dependent. The authors proposed a specialised labelling algorithm to identify the optimal tour using the dynamic programming paradigm. While this approach yields the optimal solution for their proposed problem, it is not sufficiently fast to solve larger instances within a reasonable computation time. Consequently, they also proposed a heuristic method that adopts a restricted variant of the exact approach.

The exact approaches

As evident for Table 3.2, the literature related to the exact approaches results to be relatively limited. In addition to the already described papers on the exact approaches on the other sections, Yu et al. [128] investigated a variant of the TOP, where both arrival and service times influence the amount of profit collected. The problem integrates time scheduling into the routing decisions making it particularly relevant for humanitarian search and rescue missions where survival rates decrease sharply over time. Rescue teams must assist trapped individuals across multiple affected locations with the number of lives saved depending not only on the visited sites but also on the time spent at each. Efficient allocation and scheduling of rescue teams are therefore critical in order to maximising survival rates. To address this problem, the authors formulated a mixed integer nonconcave programming model and develop a Benders Branch & Cut algorithm integrated by valid inequalities in order to strengthen the upper bounds. Computational experiments demonstrated that the valid inequalities significantly reduce the optimality gap and the proposed exact method outperforms the SCIP mixed integer nonlinear programming solver successfully finding optimal solutions where the solver fails.

Finally, Morandi et al. [85] extended the traditional OP framework in order to incorporate multiple drones collaborating with a truck to visit a subset of the given nodes. The authors referred to this variant as to the Orienteering Problem with Multiple Drones. In this setting, drones have limited battery endurance: they can either travel alongside the truck without consuming battery power or be launched to perform short flights between different customer locations. Each drone flight must start and end at separate customers and serves exactly one customer per flight. Additionally, both the truck and the drones are required to wait for one another at designated landing points. Each customer's prize can only be collected once, either by the truck or by a drone. As in the classical OP, the objective is to maximize the total collected prize, ensuring that both the truck and the drones return to the depot within a specified time limit. The authors proposed a mixed-integer linear programming formulation for this problem and developed a specialised Branch & Cut algorithm leveraging a novel problem decomposition. The method successfully solves instances with up to 50 nodes within one hour using a standard computational setup. Finally, the authors adapted the framework in order to address closely related problems from the literature and benchmarked its performance against existing methods.

3.2 The stochastic and dynamic perspective

The study of orienteering problems has gained attention due to their practical importance and theoretical complexity. When examined through a stochastic and dynamic point of view, the TOP-ST-MIN presents additional challenges including uncertainty in travel and service times, and incomplete or evolving information. These aspects influence the three key features of the problem. This review highlights developments in adaptive mathematical models and solution techniques designed to address these challenges focusing on the computational efficiency and practical applicability in stochastic and dynamic contexts.

3.2.1 The stochastic problem

Despite its practical relevance, literature about stochastic variants of OPs is still limited. The main existing solution approaches collected from the literature consist in scenario-based methods for stochastic programming, Markov decision process and sample average approximation techniques.

The earliest related study identified is Taylan and Daskin [114], which introduces the Orienteering Problem with Stochastic Profits. In this problem, the objective is to identify a tour that visits a subset of nodes within a predefined time budget maximising the likelihood of achieving a minimum target profit level. The profit of each node is represented as a normally distributed random variable. The authors

developed an exact solution method based on a parametric formulation of the problem and proposed a bi-objective genetic algorithm as a heuristic approach. They also provided a detailed analysis of the problem's characteristics and reported computational evaluations of the proposed algorithms.

Campbell et al. [23] introduced the Orienteering Problem with Stochastic Travel and Service Times, a variant of the OP in which travel and service times follow specified probability distributions. In this problem, a reward is earned if a commitment to a customer is fulfilled by the end of the day where, an unfulfilled commitment results in a penalty. This model addresses the operational challenge faced by companies that may have more customers than they can serve in a single day. The authors identified specific instances of the problem that are solvable with exact methods and demonstrated how existing variable neighborhood search heuristics can be applied to address more general cases.

In this work on the Orienteering Problem with Stochastic Weights, Evers et al. [36] addressed the inherent uncertainty of real-world applications by proposing a two-stage stochastic programming model with recourse. The model evaluates the impact of stochastic weights on the expected total profit in the first stage. To manage the non-linearity of expected profits, a linearisation method has been introduced in order to provide an approximation of the profit for a given route under various weight realisation scenarios. In order to solve the problem, the authors proposed a sample average approximation algorithm but the computation of the optimal solution for the two-stage model can be computationally expensive. For this reason, the authors implemented a heuristic approach for this problem. This method results to be effective in handling larger instances.

Another study addressing stochastic travel times is presented by Verbeeck et al. [122], introducing the Stochastic Time-Dependent Orienteering Problem with Time Windows. In this variant, travel time between locations is represented by random variable depending on the departure time from the initial location. The authors developed three fast and efficient algorithms to tackle this problem based on the metaheuristic approach. They conducted extensive experiments on realistic benchmark instances with diverse sizes and characteristics.

More recently, Angelelli et al. [3] introduced the Probabilistic Orienteering Problem, a variant of the OP defined on a directed graph where each arc has an associated cost and each node has a prize but is only available with a certain probability. A server departs from a fixed origin, has a set budget to visit a subset of nodes and must end at a designated destination. In the first stage, a subset of nodes is selected and a corresponding a priori path is determined ensuring the server can visit all nodes in the subset and reach the destination within budget. Once the availability of each node in the subset is revealed, the server follows the initial path omitting the nodes that are unavailable. The objective of this problem is to determine a first-stage

solution that maximises the expected profit of the second-stage path defined as the difference between the expected total prize and the expected total cost. The authors discussed the problem's relevance and formulated it as a linear integer stochastic programming model. They developed a branch-and-cut approach along with several matheuristic methods that employ different strategies to explore the search space.

The Stochastic Orienteering Problem is introduced by Zhang et al. [131] within a network of queues where the traveller must arrive at locations during specified time windows to collect rewards. However, the traveller faces stochastic waiting times at each location before service can start. To maximise the expected reward, the traveller needs to decide which locations to visit and how long to wait at each one. The problem is formally modelled as a Markov decision process with the objective of maximizing the expected reward. It is tackled using an approximate dynamic programming approach based on rollout algorithms. This method incorporates a two-stage heuristic estimation known as compound rollout. In the first stage, the algorithm determines whether to stay at the current location or move to another. If the decision is to leave, the next location is selected in the second stage. The effectiveness of the proposed model and solution methods is demonstrated by comparing dynamic policies with pre-determined route solutions involving recourse actions.

Reyes-Rubiano et al. [99] tackled the challenge of managing Unmanned Aerial Vehicles (UAVs) for data collection and product delivery in smart cities. UAVs offer a potential alternative to traditional road vehicles in congested urban environments where they can often complete tasks more quickly. However, UAVs are limited by their driving-range capacity. The study assumes that the reward each UAV earns from visiting a customer is a random variable with service time at each customer dependent on the reward collected. The objective is to determine the optimal set of customers for each UAV to visit respecting range limitations. To solve this, the authors developed a simheuristic algorithm which is validated through extensive computational experiments.

In contrast, Thayer and Carpin [116] introduced a variant of the Stochastic Orienteering Problem incorporating chance constraints with random edge traversal times. In this work, the path length is treated as a random variable and the model includes chance constraints to control the probability of the path length exceeding a specified limit. The authors formulated this problem as a constrained Markov decision process and applied a Lagrangian approach to solve it. By leveraging structural characteristics of this problem, they were able to efficiently compute the optimal Lagrange multiplier. This allows to compute a good feasible solution faster than the linear programming-based methods for this problem with chance constraints. Experimental results confirmed the efficiency of this approach.

3.2.2 The dynamic problem

Research on dynamic variants of the OPs, and VRPs in general, has gained increasing attention due to their practical relevance in real-time tasks. However, this area of study remains underexplored with a limited number of works compared to the deterministic area. Despite this, the field is gradually expanding as researchers recognize the importance of addressing dynamicity such as new customer requests which are common in real-world applications. To tackle the dynamic aspects of the problem, various strategies adopt online optimisation methods which adjust decisions based on newly integrated information.

The first considered work is Azi et al. [12]. The authors examined a vehicle routing problem where vehicles execute multiple delivery routes throughout their workday with new customer requests arriving dynamically. The approach proposed for solving this problem is based on an adaptive large neighbourhood search heuristic which was originally developed for the static version of the problem. In the dynamic setting, various potential scenarios are considered regarding future requests to determine whether a new request should be added to the current solution. It is important to note that the real-time decision involves the acceptance of a request, not its immediate service which can only occur in future routes once a delivery route is closed after a vehicle departs from the depot. The computational results include a comparison with a myopic approach which does not account for future request scenarios.

Ritzinger et al. [100] addressed a survey that summarises recent developments in the field by introducing a new classification based not only on the available stochastic data but also on the computational effort required at various decision points. The analysis compares the solution quality for the approaches that consider dynamic or stochastic aspects with respect to those ones that take into account both. Additionally, the survey examines the research intensity in different problem classes and evaluates their impact in recent years. Finally, the authors presented guidelines and suggests promising areas for future research.

The dynamic parcel services problem has been explored by Ulmer et al. [119]. The authors discussed parcel services in which vehicles are routed to pick up parcels within a service area with pickup requests arising dynamically throughout the day. The study highlights how such systems require flexible and adaptive approaches to handle the unpredictability of customer demands. Because of limited working hour, service vehicles have restricted time to manage these dynamic requests meaning that not all requests can be fulfilled. To maximise the number of confirmed requests dispatchers need to effectively manage their time by predicting future requests. A heuristic approach is proposed adopting approximate dynamic programming techniques to effectively address the problem. The algorithm simulates the problem's

evolution repeatedly to approximate values for each combination of time point and available time budget, helping to approximate an optimal decision policy. Given the large number of possible combinations, a dynamic lookup table is introduced as a method to partition the vector space adaptively for the approximation process.

A recent study by Angelelli et al. [4] introduced an online version of the OP where stochastic customer service requests arise dynamically over a specific time interval on the nodes of a graph. Each request must be accepted or rejected immediately upon arrival. Subsequently, within a second time interval, a vehicle is tasked on visiting all accepted customers. Accepted requests incur a prize specific to each customer and a service cost determined by routing decisions with the additional constraint that accepting a request reduces the time available for future ones. The objective is to maximise expected profit calculated as the difference between expected prizes and service costs. The problem is modelled as an Markov decision process with analytical derivations for transition probabilities and optimal policy. Due to the computational intractability of exact policy computation, several heuristic strategies are proposed and evaluated. These include static approximations, non-anticipatory greedy algorithms and sample average approximation techniques leveraging Monte Carlo sampling of potential future scenarios. Extensive computational experiments assess the performance of these algorithms offering insights into their advantages and limitations.

3.3 The fairness perspective

Ensuring a fair distribution of resources among agents can introduce further complexities into an optimisation problem potentially leading to a more balanced but less efficient outcome. The incorporation of fairness not only affects the allocation of resources but also influences how outcomes are perceived by participants making it a central issue in fields like humanitarian logistics, resource allocation and public service distribution. This section explores recent advancements in integrating fairness into optimisation problems focusing on the trade-off between fairness and efficiency, the mathematical models used to address these concerns, and the practical implications of these approaches in real-application settings. Lastly, this section concludes examining the literature concerning the concept of fairness over time.

The first work collected is reported by Schulz and Stier-Moses [105] where the authors analysed the route-guidance system in a fair perspective. The proposed approach calculates a traffic pattern that minimises total travel time respecting user constraints. These constraints ensure that the recommended routes are not significantly longer than the shortest paths under current network conditions. To configure the system, a measure called normal length is required. The authors demonstrated that when this length is set as the travel time at equilibrium, the resulting traffic

assignment is both efficient and relatively fair. Efficiency is evaluated by comparing the outcome of the best solution without guidance. Fairness is measured by comparing travel times across users showing that differences are minimal. Traffic assignments that are inefficient or unfair can lead to excessively long travel times or discourage users from using the system. This effect can undermine the effectiveness of a route-guidance system.

Bertsimas et al. [16] examined resource allocation problems involving multiple self-interested parties or players. It introduces the concept of the Price of Fairness (PoF) which quantifies the relative efficiency loss incurred adopting a fair allocation compared to a fully efficient allocation that maximises the sum of individual utilities. The analysis focuses on two widely accepted and axiomatically justified fairness criteria: proportional fairness and max-min fairness. For these notions, the study provides a detailed and tight characterisation of the PoF across a broad spectrum of resource allocation problems.

Bertsimas et al. [17] extended their work addressing a fundamental question: how to design an appropriate objective for resource allocation problems. The analysis examines how to balance these elements by exploring a framework that encompasses various resource allocation scenarios relevant to decision-makers. Specifically, it focuses on a well-studied family of objectives known for their fairness properties and investigates how to select the most suitable objective from this set. The study characterises the trade-off between efficiency and fairness that emerge choosing different objectives and offers practical managerial guidelines for making these decisions. The utility of the proposed framework is illustrated through a case study in air traffic management highlighting its practical application.

Karsu and Morton [61] addressed two concerns related to equity: equitability and balance. Three main modelling approaches address equitability concerns: the first leverages the Rawlsian principle emphasising the improvement of the least advantaged. The second adopts an explicit inequality index and the third one uses equitable aggregation functions to capture decision-makers preferences. The discussion covers two main approaches for managing balance. The first approach uses indicators to measure deviations from a reference balanced solution. The second approach converts balance concerns into equitability concerns allowing the application of methods designed for equitability. A summary of these approaches is finally provided with an exhaustive analysis of their advantages and disadvantages.

Naldi et al. [88] explored the multi-agent allocation problem where multiple departments compete for portions of a company's budget. Each department has its own set of projects, each with specific expected profits and costs that aims to select the most profitable subset of projects within its allocated budget. Even if the overall company profit is an important factor the decision maker must also consider fairness. In this context the work introduces an equity criterion based on maximin fairness that aims

to balance profit and fairness. A bi-criteria integer linear programming model is developed in which one objective maximises the total profit while the other one maximises the minimum budget assigned to any department. After an experimental analysis the study reveals a near-perfect linear anti-correlation between profit and fairness index values that indicates a trade-off between the two objectives.

Nicosia et al. [90] explored the challenge faced by decision makers for allocating a limited resource across multiple agents in order to maximise the overall utility. This issue is central to resource allocation problems where fairness and efficiency must often be balanced. An optimal solution may result unbalanced and potentially perceived as unfair. Conversely, balanced allocations may not correctly maximise utility. The focus of this study is to evaluate the quality of fair solutions by considering the system efficiency loss accordingly to a fair allocation. The latter is compared to one that maximises the total utility. The PoF is analysed under three fairness criteria: maximin fairness, Kalai–Smorodinski fairness and proportional fairness. The results are twofold. Firstly, a set of properties applicable to any general multi-agent problem is formalised without assuming specific details about agent utilities. Secondly, an allocation problem is introduced where each agent consumes a bounded resource in discrete quantities (items). The utility of each agent is defined to be the sum of weights of the allocated items. Two scenarios are distinguished. In the first one, the agents have disjoint sets of items. In the second one, they share a common set. The maximisation of total utility corresponds to a subset sum problem. For this problem, an upper and a lower bound on the PoF are given, which depend on an upper limit on the size of the items.

Challenges faced by non-governmental organisations in delivering humanitarian aid to vulnerable rural populations inspire many logistical studies. Anaya-Arenas et al. [2] analysed this scenario. A key concern for these organisations ensures that the distribution process is both impartial and transparent by emphasising the concept of fairness. The article explores how fairness can be defined and adopted in scenarios that require delivery of essential items. The approach incorporates performance metrics to address fairness. It offers valuable considerations for organisations responsible for the equitable resource allocation. Additionally, an empirical analysis of an academic case modelled on a rural aid distribution challenge illustrates how different mathematical models can help crisis managers by considering fairness and equity into their strategies.

Disaster management covers various stages including post-disaster situations where an immediate response can be considered crucial. Aringhieri et al. [6] explored humanitarian logistics that play a crucial role in ensuring the effectiveness of relief operations particularly during the response phase. One critical aspect is the ambulance routing problem in which the aim consists in the allocation and routing of ambulances to transport critical injured patients to hospitals efficiently. This process

must consider equity and fairness to ensure impartial delivery of services to individuals with different levels of need. Effective management in the immediate aftermath of disasters requires to address both efficiency and equitable service delivery. The research focuses on the development of the optimal ambulance routing strategies during disaster relief operations considering fairness and equity to balance services among patients. More details of this work can be found in Section 2.3.

Addressing road congestion is a critical challenge in modern cities due to its negative effects on time efficiency, pollution, industrial costs and extensive road maintenance requirements. Morandi [86] examined advances in intelligent transportation systems. These technologies facilitate the implementation of a coordination mechanism known as coordinated traffic assignment which aims to optimize route assignments for drivers in order to reduce congestion. This allows to minimize the total travel time within traffic networks. Among various congestion management strategies, coordinated traffic assignment is particularly effective because it does not require excessive investments in expanding the road infrastructure. Traffic assignments focus on two main concepts inspired by the Wardropian principles: (i) user equilibrium and (ii) system optimum. User equilibrium represents a self-interested approach in which each driver selects the most convenient path in order to ensure fairness by making travel times equal for all users with the same origin and destination. However, this approach results in a suboptimal system-wide travel time that can lead to the Price of Anarchy (PoA) concept. Conversely, the system optimum aims to minimize the total travel time across the network. However, this approach can lead to disproportionate travel times for certain user groups that potentially can reduce compliance and fairness. Over the past few decades, significant efforts have been made to integrate these two goals in traffic assignment. This survey examines the state-of-the-art approaches that explore this trade-off between fairness and efficiency in traffic management.

The concept of fairness can also be extended in order to balance the shared resources over time. With this regard, Bampis et al. [15] investigated the dynamic variant of the Max-Min Fair Allocation problem, where the time horizon is divided into multiple steps. At each time step, a set of demands and a set of available resources are given, potentially changing over time and defining a different instance at each step. The objective is to determine a sequence of solutions that remain near-optimal at each time step ensuring minimal changes between consecutive solutions to maintain stability. The influence of future knowledge on solution quality and stability is analysed under three scenarios: (i) the offline setting, where the entire sequence of instances is known in advance, (ii) the online setting, where no future information is available and (iii) the k-lookahead setting, where a limited number of future instances are known at each step. Initially, the authors addressed the case without restrictions, where the set of agents and resources remains constant over time and each resource can be allocated to any agent. In this context, the overtime

problem is shown to be significantly harder than its static counterpart, becoming computationally intractable even for simple families of instances. For the offline setting, an approximation algorithm is proposed, whose performance depends on the approximation quality of the static version. The authors also developed a competitive algorithm for the online setting, based on the same static approximation method. When restrictions are placed on resource allocations, it is shown that a polynomial-time algorithm with the same approximation guarantee is achievable in the offline setting. In contrast, for the online setting, no online algorithm can achieve a bounded competitive ratio. In the one-step lookahead setting, an approximation algorithm is presented, again building on the approximation algorithm used for the static problem.

Finally, Lodi et al. [78] presented a study that addresses problems involving multiple stakeholders, where a centralized decision maker is required to make fair decisions. Each decision results in different utility levels for the stakeholders. When decisions are made repeatedly, efficient mathematical programming formulations are provided in order to identify the highest level of fairness achievable and the decisions that progressively improve fairness, based on reasonable fairness metrics. The authors applied this framework to the ambulance allocation problem, where decisions in successive rounds are constrained. With this added complexity, structural results are established for identifying fair and feasible allocation policies, and a hybrid algorithm is proposed that combines column generation and constraint programming techniques in order to solve this class of problems. Computational experiments show that the proposed method significantly outperforms a naive approach, solving these problems much more quickly.

3.4 Conclusions

This chapter provided a literature review on the articles related to the TOP-ST-MIN for its deterministic version, for the stochastic variants of the OP and TOP and, more generally, dynamics aspects in routing problems. Furthermore, the literature of fairness in optimisation is briefly reported.

The review examined deterministic approaches, which rely on mathematical models and optimization techniques, as well as methods that address the dynamic and stochastic aspects of the problem. Additionally, fairness considerations in resource allocation within shared environments were explored in order to highlight the increasing recognition of equity in solving such problems. This literature review has outlined potential research directions for the TOP-ST-MIN.

This chapter together with Chapter 2 provides a strong justification for studying the TOP-ST-MIN: Chapter 2 shows its relevance to real-world applications, while this chapter proves that the problem never been addressed in the existing literature.

4 Instances generation

The generation of the instances is a fundamental aspect of algorithmic evaluation in optimisation. By creating a proper and diverse set of problem instances, researchers can rigorously test the performance, efficiency, and robustness of algorithms under a variety of conditions. This chapter focuses on the methods and strategies used to generate instances for the TOP-ST-MIN, highlighting the computational impact introduced by the three new features. It explores the key principles behind instance design, the factors that influence instance characteristics. To evaluate the computational impact of the new features, this chapter explores different logics for generating the new instances.

4.1 Generation methods

The section describes two generation methods for the mandatory nodes, two for the physical incompatibilities, two for the logical incompatibilities and a single method for generating service times. All the following methods, except for the service times generation, were developed in a fully-deterministic mode to control the number and selection of mandatory nodes and to specify the number and topology of arcs in the graph. The rationale behind this choice is to evaluate the impact of the considered features on the problem's resolution. In fact, all the feature generation methods were intentionally developed in contrasting ways in order to highlight the specific characteristics of each feature.

In order to reduce the risk of creating infeasible instances, the generation procedure guarantees the existence of routes capable of visiting each mandatory node individually, without violating incompatibility constraints and respecting the time budget. The parameters used in each method were determined through preliminary tests.

As a starting point, the TOP benchmark sets available in the literature were used, which can be accessed online at `https://www.mech.kuleuven.be/en/cib/op`. This benchmark set, proposed by Chao et al. [24], is composed of 387 instances divided into 7 sets in accordance with the number of nodes ranging from 21 to 102. Each set differs from the number of teams (vehicles, from 2 to 4) available to serve the customers and for the value of the maximum time duration of a route. The instances are composed by nodes represented though two-dimensional coordinates in a Euclidean plane.

4.1.1 Mandatory nodes

Two distinct methods have been identified to generate the mandatory nodes. The first method, the Scattered Mandatory (SMN) approach, selects the mandatory nodes maximising the sum of distances between all pairs of selected nodes. As depicted in Figure 4.1, this approach ensures that the mandatory nodes are distributed far from each other potentially increasing the complexity of these instances. On the other

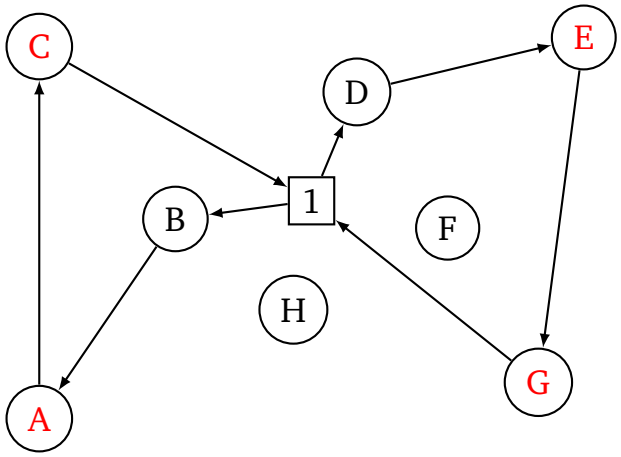


Fig. 4.1: A graphical representation of a feasible solution composed by two vehicles for an instance characterised by mandatory scattered nodes.

hand, the second method, the Clustered Mandatory (CMN) approach, creates an aggregate of mandatory nodes close to each other as illustrated in Figure 4.2. This approach leads to solutions where the mandatory nodes are more concentrated in specific areas, making it easier for a vehicle to visit them.

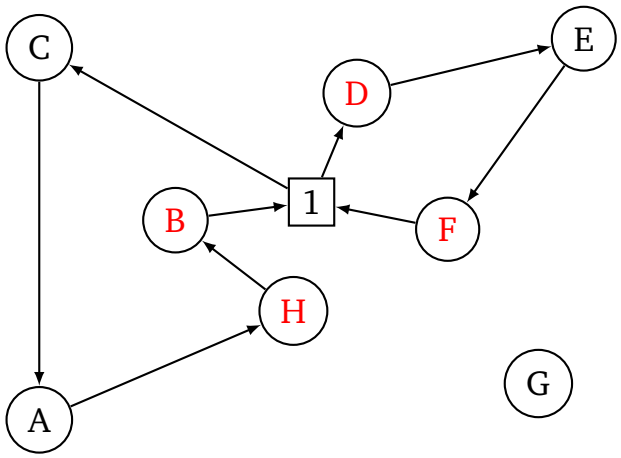


Fig. 4.2: A graphical representation of a feasible solution composed by two vehicles for an instance characterised by mandatory clustered nodes.

These two methods provide contrasting strategies for structuring the mandatory nodes, each offering unique challenges for optimisation algorithms and allowing for a more comprehensive evaluation of their performance under different conditions. To achieve this, the algorithms recently reported in Aringhieri et al. [9] for the Maximum Diversity Problem [67] were adopted. This problem aims to select a subset of elements that maximises the overall diversity, usually measured by the sum of pairwise distances. Finally, following Assunção and Mateus [11] for the Steiner TOP, 5% of the total number of customers are set as mandatory.

4.1.2 Physical incompatibilities

We adopt two strategies to define the topology of the arcs composing the graph representing the instances. The first one, the Clusters-based Physical Incompatibilities (CPI) method, tends to create a graph composed of subgraphs in which some of them are completely or partially disconnected from each other. On the contrary, the second, the Degree-based Physical Incompatibilities (DPI) method, tends to generate a graph in which the degree of each node is similar to each other. To avoid generating overly simple or infeasible instances, 20% of the total number of arcs are removed (practically, 20% of physical incompatibilities are imposed using one of the proposed approaches). Finally, a bidirectional crossing of the arcs is also guaranteed, i.e. If the arc (i, j) is included in the set of arcs, the arc (j, i) will also be included, and vice versa.

The clusters-based approach

The basic idea is to build a graph by partitioning the n nodes and reducing the connections among nodes belonging to different partitions. We start by running the K-Means algorithm [81] to partition all the nodes (except the source and destination ones) in $c = 3$ different clusters. It is worth noting that such an algorithm may not create balanced partitions. An incompatibility is then imposed between pairs of clusters (e.g., clusters k and w) through a uniform distribution, meaning that all the nodes belonging to cluster k could end up being incompatible with the ones belonging to cluster w. The final graph is built by solving an integer linear program which selects a given amount of arcs in such a way to minimise the number of selected arcs among pairs of incompatible clusters. As a secondary objective, the model tries to balance the number of arcs inside a given cluster and towards other (compatible) clusters. The idea is to foster the generation of instances with a larger number of feasible solutions. The mathematical model follows.

$$\min \quad (\gamma + 1) \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{c} \sum_{w=1}^{c} \lambda_{ik}\, \lambda_{jw}\, \phi_{kw}\, x_{ij} \right) + \sigma \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{c} \sum_{w=1}^{c} \lambda_{ik}\, \lambda_{jw}\, x_{ij} = \gamma, \tag{4.1b}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_{ik}\,\lambda_{jk}\,x_{ij} \leq \sigma \quad \forall\, k = 1,\ldots,c, \tag{4.1c}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{w=1}^{c}\lambda_{ik}\,\lambda_{jw}\,x_{ij} \leq \sigma \quad \forall\, k = 1,\ldots,c, \tag{4.1d}$$

$$x_{ij} = x_{ji}, \quad \forall\, i,j = 1,\ldots,n, \tag{4.1e}$$

$$x_{ij} \in \{0,1\}, \quad \forall\, i,j = 1,\ldots,n, \tag{4.1f}$$

$$\sigma \in \mathbb{N}^{+}. \tag{4.1g}$$

The parameters λ_{ik} is equal to 1 if and only if the node i belongs to the cluster k, 0 otherwise, and ϕ_{kw} is equal to 1 if and only if the cluster k is incompatible with the cluster w, 0 otherwise. The variable x_{ij} is 1 if the arc (i,j) has been chosen, 0 otherwise. The variable σ counts the maximal number of arcs intra and inter clusters. The hierarchical objective function (4.1a) minimises the arcs that violate the physical incompatibilities as first goal balancing the intra and inter arcs between clusters as a second goal. The constraint (4.1b) guarantees that exactly γ (γ is equal to 80% of the total number of arcs in a complete graph of n nodes) arcs will be chosen. The constraints (4.1c) and (4.1d) calculate the maximal number of arcs intra and inter clusters, respectively. The constraints (4.1e) guarantee the bidirectional crossing of the arcs. Finally, the constraints (4.1f) and (4.1g) are the variable definition constraints. An example of a graph characterised by clusters-based physical incompatibilities is reported in Figure 4.3 where four distinct clusters, each represented by a different colour, are highlighted.

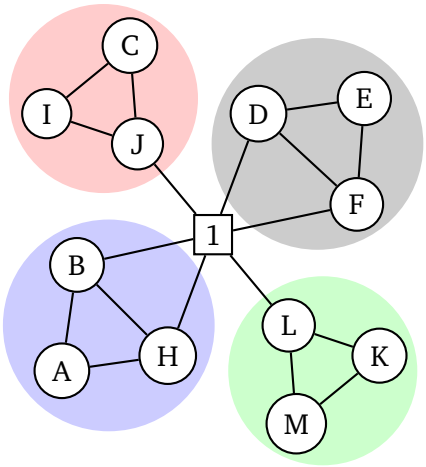


Fig. 4.3: A graphical representation of a portion of a graph designed to emphasize physical incompatibilities, strategically arranged in order to create clusters within the graph.

The degree-based approach

Compared to the previous approach, the aim is now to select the arcs in such a way that the degree of the nodes is similar in such a way to balance the incompatibilities among the nodes. To create a graph along these lines, the following integer linear program was developed:

$$\min \quad \alpha \tag{4.2a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = \gamma, \tag{4.2b}$$

$$\sum_{k=1}^{n} x_{ik} \leq \alpha \quad \forall\, i = 1, \ldots, n, \tag{4.2c}$$

$$x_{ij} = x_{ji}, \quad \forall\, i, j = 1, \ldots, n, \tag{4.2d}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \ldots, n, \tag{4.2e}$$

$$\alpha, \beta \in \mathbb{N}^{+}. \tag{4.2f}$$

The objective function (4.2a) minimises the maximum degree between all vertices. Together with the constraint (4.2b), which guarantees that exactly γ arcs will be chosen, and the constraints (4.2c), which links variables α and x_{ij}, the objective function fosters the balancing of the degree among the nodes. The constraints (4.2d) guarantee the bidirectional crossing of the arcs.

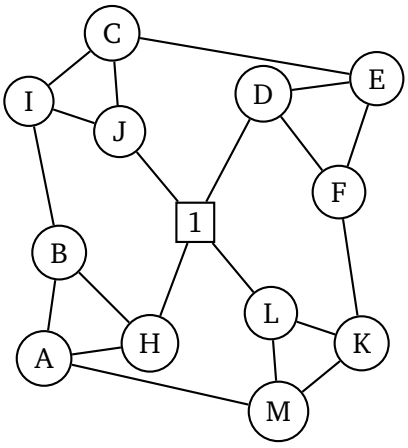


Fig. 4.4: A graphical representation of a portion of a graph designed to emphasize physical incompatibilities, strategically arranged in order to equally distribute the degree of the nodes (except for the source node).

Finally, the constraints (4.2e) and (4.2f) are the variable definition constraints. Figure 4.4 depicts an example of a graph characterised by degree-based physical incompatibilities where, the degree of each node is balanced.

4.1.3 Logical incompatibilities

Two distinct methods have been designed to generate logical incompatibilities. The first one, the Nearest Logical Incompatibilities (NLI) approach, establishes a logical incompatibility between the node i and all nodes inside the set of nearest nodes from i. This method ensures that the incompatibility is applied to nodes that are spatially or operationally close to i by creating a challenging scenario for route planning and resource allocation. The structure of these incompatibilities are depicted in Figure 4.5, where nodes filled with different colours indicate logical incompatibility. In contrast, the second method, referred to as Farthest Logical Incompatibilities

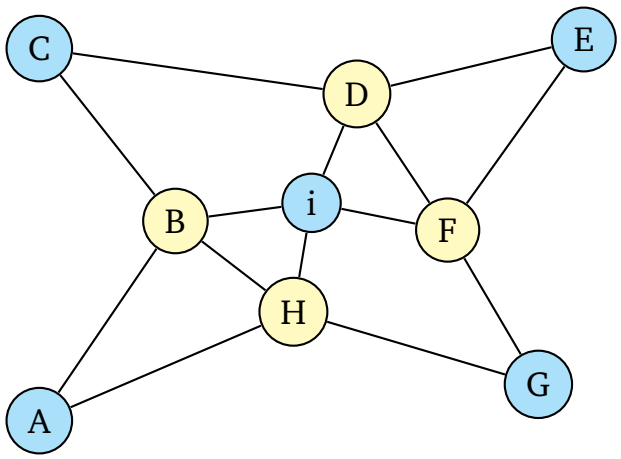


Fig. 4.5: A graphical representation of a portion of a graph designed to emphasise logical incompatibilities, systematically structured to account for logical incompatibilities between node i and the set of nodes nearest from i. Different node colours means different node categories. Two nodes belonging to two different categories are considered to be logical incompatible.

(FLI), introduces logical incompatibilities between a node i and all nodes contained in the set of the farthest nodes to i. This approach ensures that nodes far from i are incompatible as illustrated in Figure 4.6. The set of farthest and nearest nodes are calculated by taking 5% of the farthest and nearest nodes to i, respectively. The former method favours routes where the customers tend to be close to each other while the latter favours routes where the customers are distant from one another.

4.1.4 Service times

Finally, the last feature to consider is the service times of the customers, which are generated using a uniform distribution. A global amount of service time $m\,T_{\max}\,/\,2$ is randomly distributed among the nodes where, m is the number of routes and $T_{\max}$ the time budget for each of them. Furthermore, to create challenging instances, the time limit for each route is increased by $T_{\max}\,/\,2$, obtaining a new time budget of $3\,T_{\max}\,/\,2$.

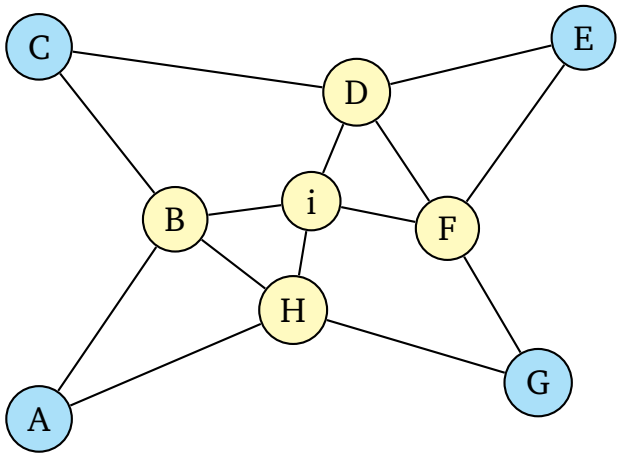


Fig. 4.6: A graphical representation of a portion of a graph designed to emphasise logical incompatibilities, systematically structured to account for logical incompatibilities between node i and the set of nodes farthest from i. Different node colours means different node categories. Two nodes belonging to two different categories are considered to be logical incompatible.

4.2 The instance generation scheme

Combining all the presented methods, 4 and 8 different generation schemes were developed respectively for the TOP-ST-MIN-P and TOP-ST-MIN-PL instances, as reported in Figure 4.7: the root node represents a generic initial TOP instance while each level of the tree consists in the generation of a specific new feature. Assuming that the root of this picture represents the level 0, a path from the root to a node at the level 2 (dark grey area) describes a generation scheme for a TOP-ST-MIN-P instance while a path from the root to a leaf (light grey area) describes a generation scheme for a TOP-ST-MIN-PL instance.

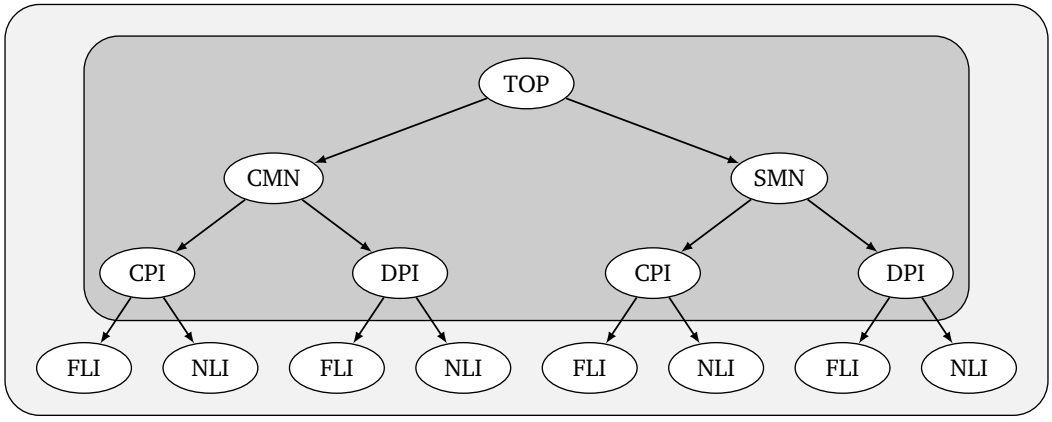


Fig. 4.7: Illustration of the instances generation schemes.

Table 4.1 summarises the results of the generation process. Since the number of TOP-ST-MIN instances would have been too large, a subset of the instances from the

initial TOP benchmark set was considered. The columns TOP report the information about the TOP instances (Chao et al. [24]), that is the number of instances taken into account for the generation process and the number of nodes ($|N|$) and arcs ($|A|$) for each instance in the initial benchmark set. The columns TOP-ST-MIN-P report the number of new instances generated, and the number of mandatory nodes ($|M|$) and physical incompatibilities ($|I|$) while the columns TOP-ST-MIN-PL add the number of logical incompatibilities ($|C|$). Note that all the initial instances are generated on the two-dimensional euclidean plane and, as a consequence, the classical triangle inequality property holds for each new generated instance.

Finally, the newly generated instances were grouped into three benchmark sets based on the number of nodes (*small*, *medium* and *large*) as reported in Table 4.1.

Tab. 4.1: A summary of the newly generated instances.

		TOP			**TOP-ST-MIN-P**			**TOP-ST-MIN-PL**			
	Set	#	$\|N\|$	$\|A\|$	#	$\|M\|$	$\|I\|$	#	$\|M\|$	$\|I\|$	$\|C\|$
SMALL	1	15	32	992	60	2	198	120	2	198	30
	2	9	21	420	36	1	84	72	1	84	10
	3	15	33	1056	60	2	211	120	2	211	30
	ALL	**39**	-	-	**156**	-	-	**312**	-	-	-
MEDIUM	5	15	66	4290	60	3	858	120	3	858	86
	6	15	64	4032	60	3	806	120	3	806	100
	ALL	**30**	-	-	**120**	-	-	**240**	-	-	-
LARGE	4	15	100	9900	60	5	1980	120	5	1980	230
	7	15	102	10302	60	5	2060	120	5	2060	240
	ALL	**30**	-	-	**120**	-	-	**240**	-	-	-

4.3 Conclusions

The generation of the problem instances is a critical step in the evaluation of algorithms for the TOP-ST-MIN. This chapter outlined the methodologies for instance generation in order to emphasise the integration of the introduced features. It was possible to maintain precise control over the instance characteristics guaranteeing feasibility through the design of routes that respect all constraints. The original TOP benchmark sets provided a robust starting point and were modified to consider all the additional features of the TOP-ST-MIN.

This structured approach ensures that the generated instances remain both challenging offering valuable insights into the computational implications of the problem's novel features. The instance generation procedure focused on the three key features that define the TOP-ST-MIN, aiming to assess their impact based on the difficulty these features pose to the algorithms. This approach provides deeper insight into

why algorithms may perform differently across instances, allowing for a more distinct comparison between methods.

Chapter 5 will adopt these newly generated instances as a benchmark set for testing the performance of several solution methods for the TOP-ST-MIN while chapters 7 and 8 will describe how to adapt them in order to account uncertainty.

5 Solutions for the deterministic TOP-ST-MIN

Drawing inspiration from the two real-world problems discussed in Chapter 2, the Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (TOP-ST-MIN) emerges as a new variant of the Team Orienteering Problem (TOP). From the DSTCP, it inherits service times at nodes reflecting tasks like diagnostic swab collection. From the ARP, it incorporates service times for patient care, mandatory nodes to represent critical patients requiring immediate attention and physical incompatibilities in order to address the transportation constraints of these patients. Furthermore, in alignment with existing literature (e.g., Palomo-Martínez et al. [94]), the TOP-ST-MIN extends its scope by including exclusionary constraints referred as logical incompatibilities.

This chapter presents two mathematical formulations for the problem (Sect. 5.3). It also introduces an exact cutting-plane algorithm for efficiently achieving optimal solutions (Sect. 5.4). Furthermore, two heuristic approaches are designed to handle larger problem instances (Sect. 5.6). An extensive quantitative analysis is then discussed in order to evaluate the performance of the proposed approaches across various benchmark sets examining the computational impact of key problem characteristics, respectively in Section 5.5 and 5.7.

5.1 Mathematical notation

Mathematically, the TOP-ST-MIN can be modelled by using a complete directed graph $G = (N, A)$ where N is the set of nodes denoted from 1 (the source or depot) to n (the destination), and A is the set of directed arcs across the nodes in N. To simplify the readability of the mathematical formulations, the set $\hat{N} = \{2, \ldots, n-1\}$ is introduced to represent the set of *customers* and the set $\hat{A} = \{(i, j) \in A : i \in \hat{N} \cup \{1\},\ j \in \hat{N} \cup \{n\},\ i \neq j\}$ to represent the set of *traversable* arcs. To each node $k \in \hat{N}$ is associated a non-negative amount of service time s_k and a profit p_k. Each arc has a non-negative travel time t_{ij} across the node i and j. Further, the following three sets are introduced: let $M \subset \hat{N}$, $I \subset \hat{A}$, $C \subset \hat{N} \times \hat{N}$ be the set of mandatory nodes, the set of physical incompatibilities, and the set of logical incompatibilities, respectively. The objective of the TOP-ST-MIN is to determine a set $R = \{1, \ldots, m\}$ of routes from the source node to the destination node that maximises the total profit collected, while respecting that the total time (including service and travel times) for each route does not exceed the maximum allowed time, $T_{\max}$. Additionally,

each mandatory node must be visited, and no route should violate the specified incompatibilities.

5.2 Complexity results

The proof that the TOP-ST-MIN is NP-hard is relatively simple and follows from a reduction of the classical TOP to the TOP-ST-MIN. Any instance of the TOP can be transformed into an instance of the TOP-ST-MIN by defining an empty set of mandatory nodes and incompatibilities, and setting all service times to zero. This shows that the TOP-ST-MIN is at least as hard as the TOP. Since the TOP has been proven to be NP-hard by Butt and Cavalier [21], it follows that the TOP-ST-MIN is also NP-hard.

Moreover, it is possible to demonstrate that the additional features introduced in the TOP-ST-MIN make the problem of finding a feasible solution NP-complete in certain cases, even for the single-path version of the problem (OP-ST-MIN). The Feasibility Problem (FP) for the OP-ST-MIN involves finding a single route from the source to the destination node that visits all mandatory nodes, satisfies the incompatibility constraints, and ensures that the total route time does not exceed a maximum threshold, $T_{\max}$. The following statement is then presented:

Theorem 5.2.1. *Finding a feasible solution for the OP-ST-MIN is NP-complete.*

Proof. In order to prove the above theorem, a reduction to the well-known NP-complete Hamiltonian Path Problem (HPP) [60] is provided. An HPP instance consists of an undirected graph $G = (V, E)$ to which the following question is associated: is there a path in G which contains all the nodes of V? Starting from a generic instance ψ of HPP, an instance ϕ of OP-ST-MIN is constructed as follows:

- introduce a graph $G'(V', E') = G$ and add two nodes s (source) and t (destination) to V' with an edge between s (respectively t) and any node of V;
- to any node of V', associate a service time of 0 and to each edge of E' associate a travel time of 0. The maximum time available is set at $T_{\max} = 0$;
- consider all nodes in V' as mandatory nodes, so that the tour of the single team must contain all nodes;
- For every edge absent in G, it is assumed that a physical incompatibility exists between the corresponding pair of nodes. No logical incompatibility is present.

Suppose first that a Hamiltonian path p exists in G, i.e., ψ is a *Yes* instance. An OP-ST-MIN tour can be built by connecting s to the first node in p and the last node in p to t. ϕ is therefore a *Yes* instance.

Conversely, suppose that ϕ is a *Yes* instance, i.e., there exists a tour from s to t. Removing nodes s and t from the tour, a path of G containing all the nodes of V is obtained, which is therefore a Hamiltonian path for G. Then, ψ is a *Yes* instance, which completes the proof. □

It is observable that the presence of mandatory nodes and physical incompatibilities makes the FP NP-complete. On the contrary, in a generic instance of the TOP (i.e., the graph is complete and no mandatory nodes are present), it is always possible to perform a tour directly from s to t without performing any additional stop. This proves that the TOP-ST-MIN is harder to tackle with respect to the TOP.

5.3 Mathematical formulations

Two distinct mathematical formulations for the TOP-ST-MIN are introduced: the route-based formulation and the flow-based formulation. Section 5.3.1 discusses the first model which addresses the TOP-ST-MIN considering only physical incompatibilities (referred to as TOP-ST-MIN-P). This model focuses on optimising the routes for a team of vehicles taking into account mandatory nodes that must be visited and avoiding conflicts that arise from physical constraints, such as the inability to visit certain nodes together due to capacity or other transportation limitations.

Section 5.3.2 extends this initial formulation by also considering logical incompatibilities as described in the introduction. This leads to the TOP-ST-MIN-PL. For example, this extension could be useful in case where different types of patients, such as those with infectious diseases, need to be transported separately due to health protocols.

Table 5.1 summarises the notation introduced for the formulations (5.1) and (5.3) and their extension to include the logical incompatibilities (5.2) and (5.4), respectively.

Tab. 5.1: Summary of the notation.

Sets	
N	nodes
$\hat{N}$	customers
A	arcs
$\hat{A}$	traversable arcs
M	mandatory nodes
I	physical incompatibilities
C	logical incompatibilities

Parameters	
n	number of nodes
m	number of routes
p_k	profit associated to the node k
t_{ij}	travelling time between the nodes i and j
s_k	service time of the node k
$T_{\max}$	time limit for each route

Decision variables (Mixed formulation)	
x_{ijr}	1 if the arc (i,j) is traversed by the route r
y_{kr}	1 if the node k is visited by the route r
z_{ijr}	flow traversing the arc (i,j) by the route r

Decision variables (Compact formulation)	
x_{ij}	1 if the arc (i,j) is traversed
y_k	1 if the node k is visited
z_{ij}	flow traversing the arc (i,j)
v_k	route identifier of the node k
u_{ij}	1 if $v_i - v_j \geq 1$

5.3.1 The route-based formulation

Combining the vehicle-based formulation for the TOP [121] and the flow-based formulation for the TOP [18], the route-based formulation for the TOP-ST-MIN-P is the following:

$$\max \quad \sum_{r=1}^{m} \sum_{k=2}^{n-1} p_k \, y_{kr} \tag{5.1a}$$

$$\text{s.t.} \quad \sum_{r=1}^{m} \sum_{j=2}^{n} x_{1jr} = \sum_{r=1}^{m} \sum_{i=1}^{n-2} x_{inr} = m, \tag{5.1b}$$

$$\sum_{(i,k)\in\hat{A}} x_{ikr} = \sum_{(k,j)\in\hat{A}} x_{kjr} = y_{kr}, \quad \forall\, k \in \hat{N},\, r \in R, \tag{5.1c}$$

$$\sum_{r=1}^{m} y_{kr} \leq 1, \quad \forall\, k \in \hat{N}, \tag{5.1d}$$

$$\sum_{r=1}^{m} y_{kr} = 1, \quad \forall\, k \in M, \tag{5.1e}$$

$$\sum_{r=1}^{m} x_{ijr} = 0, \quad \forall\, (i,j) \in I, \tag{5.1f}$$

$$\sum_{(k,j)\in\hat{A}} z_{kjr} - \sum_{(i,k)\in\hat{A}} z_{ikr} = \sum_{(k,j)\in\hat{A}} (t_{kj} + s_k)\, x_{kjr}, \quad \forall\, k \in \hat{N},\, r \in R, \tag{5.1g}$$

$$z_{ijr} \leq (T_{\max} - s_j - t_{jn})\, x_{ijr}, \quad \forall\, (i,j) \in \hat{A},\, r \in R, \tag{5.1h}$$

$$z_{ijr} \geq (t_{1i} + s_i + t_{ij})\, x_{ijr}, \quad \forall\, (i,j) \in \hat{A},\, r \in R, \tag{5.1i}$$

$$z_{1kr} = t_{1k}\, x_{1kr}, \quad \forall\, k \in \hat{N},\, r \in R, \tag{5.1j}$$

$$y_{kr} \in \{0,1\}, \quad \forall\, k \in \hat{N}, \quad r \in R, \tag{5.1k}$$

$$x_{ijr} \in \{0,1\}, \quad \forall\, (i,j) \in \hat{A} \setminus \{(1,n)\}, \quad r \in R, \tag{5.1l}$$

$$x_{1nr} \in \mathbb{N}^+, \quad \forall\, r \in R, \tag{5.1m}$$

$$z_{ijr} \in \mathbb{R}^+, \quad \forall\, (i,j) \in \hat{A}, \quad r \in R, \tag{5.1n}$$

The proposed model makes use of the following decision variables: the variable x_{ijr} is equal to 1 if and only if the arc (i,j) is traversed by the route r, 0 otherwise; the variable y_{kr} is equal to 1 if and only if the node k is visited by the route r, 0 otherwise; the variable z_{ijr} defines the arrival time at node j coming from the node i and can be thought as the amount of flow that passes through the arc (i,j) inside the route r. The objective function (5.1a) maximises the overall profit collected by each route. Constraint (5.1b) guarantees (with the optimisation sense of the objective function) that each route starts from the source node 1 and ends to the destination node n. Constraints (5.1c) guarantee the connectivity of each route while the constraints (5.1d) guarantee that a node can only be visited by at most one route. Constraints (5.1e) ensure that all the mandatory customers M will be visited. Constraints (5.1f) guarantee that all the physical incompatibilities are satisfied. Con-

straints (5.1g) and (5.1h) are the classical Gavish-Graves (GG) subtours elimination constraints adapted to the TOP [43]. Constraints (5.1h) and (5.1i) set the upper (lower) bound on the duration of the route r while the constraints (5.1j) bound the flow originating from the initial depot by the same route. Constraints (5.1k), (5.1l), (5.1m) and (5.1n) are the variable definition constraints. To incorporate logical incompatibilities into the model for TOP-ST-MIN-PL, the following constraints can be added:

$$|C_k|(1 - y_{kr}) \geq \sum_{i \in C_k} y_{ir}, \quad \forall\, k \in \hat{N}, \quad \forall\, r \in R\,, \tag{5.2}$$

to the formulation (5.1) where the set C_k represents the subset of nodes that are logical incompatible with k. The constraints (5.2) impose that only one set of nodes between $\{k\}$ and C_k can be visited by the route r.

5.3.2 The flow-based formulation

Starting from the TOP formulation reported in [18], the flow-based formulation for the TOP-ST-MIN-P is the following:

$$\max \quad \sum_{k=2}^{n-2} p_k\, y_k \tag{5.3a}$$

$$\text{s.t.} \quad \sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = m, \tag{5.3b}$$

$$\sum_{(i,k) \in \hat{A}} x_{ik} = \sum_{(k,j) \in \hat{A}} x_{kj} = y_k, \quad \forall\, k \in \hat{N}, \tag{5.3c}$$

$$\sum_{(k,j) \in \hat{A}} z_{kj} - \sum_{(i.k) \in \hat{A}} z_{ik} = \sum_{(k,j) \in \hat{A}} (t_{kj} + s_k)\, x_{kj}, \quad \forall\, k \in \hat{N}, \tag{5.3d}$$

$$z_{ij} \leq (T_{\max} - s_j - t_{jN})\, x_{ij}, \quad \forall\, (i,j) \in \hat{A}, \tag{5.3e}$$

$$z_{ij} \geq (t_{1i} + s_i + t_{ij})\, x_{ij}, \quad \forall\, (i,j) \in \hat{A}, \tag{5.3f}$$

$$z_{1k} = t_{1k}\, x_{1k}, \quad \forall\, k \in \hat{N}, \tag{5.3g}$$

$$y_k = 1, \quad \forall\, k \in M, \tag{5.3h}$$

$$x_{ij} = 0, \quad \forall\, (i,j) \in I, \tag{5.3i}$$

$$y_k \in \{0,1\}, \quad \forall\, k \in \hat{N}, \quad r \in R, \tag{5.3j}$$

$$x_{ij} \in \{0,1\}, \quad \forall\, (i,j) \in \hat{A} \setminus \{(1,n)\}, \quad r \in R, \tag{5.3k}$$

$$x_{1n} \in \mathbb{N}^+, \quad \forall\, r \in R, \tag{5.3l}$$

$$z_{ij} \in \mathbb{R}^+, \quad \forall\, (i,j) \in \hat{A}, \quad r \in R, \tag{5.3m}$$

The proposed model makes use of the following decision variables: the variable x_{ij} is equal to 1 if and only if the arc (i,j) is traversed, 0 otherwise; the variable y_k is equal to 1 if and only if the node k is visited, 0 otherwise; the variable z_{ij} defines

the arrival time at node j coming from the node i and can be thought as the amount of flow that passes through the arc (i, j).

The objective function (5.3a) maximises the overall profit collected by each route. Constraint (5.3b) guarantees (together with the sense of the optimisation of the objective function) that each route starts from the source node 1 and ends to the destination node n. Constraints (5.3c) guarantee the connectivity of each route. Constraints (5.3d) and (5.3e) are the GG subtour elimination constraints adapted for the TOP. Constraints (5.3e) and (5.3f) set the upper (lower) bound on the duration of each route. Constraints (5.3g) bound the flow originating from the initial depot. Constraints (5.3h) ensure that all the mandatory customers M are visited. Constraints (5.3i) guarantee that all the physical incompatibilities are satisfied. Constraints (5.3j), (5.3k), (5.3l) and (5.3m) are variable definition constraints. To incorporate logical incompatibilities into the TOP-ST-MIN-PL model, the following constraints are introduced into the formulation (5.3):

$$v_k \geq j \cdot x_{1k}, \quad \forall\, k \in \hat{N}, \tag{5.4a}$$
$$v_k \leq j \cdot x_{1k} - (n-2)\,(x_{1k} - 1), \quad \forall\, k \in \hat{N}, \tag{5.4b}$$
$$v_j \geq v_i + (n-2)\,(x_{ij} - 1), \quad \forall\, (i,j) \in \hat{A}, \tag{5.4c}$$
$$v_j \leq v_i + (n-2)\,(1 - x_{ij}), \quad \forall\, (i,j) \in \hat{A}, \tag{5.4d}$$
$$v_i \geq v_j + 1 - (n-2)\,(1 - u_{ij}), \quad \forall\, i,j \in C,\ i \neq j \tag{5.4e}$$
$$v_i \leq v_j - 1 + (n-2)\,u_{ij}, \quad \forall\, i,j \in C,\ i \neq j \tag{5.4f}$$
$$u_{ij} \in \{0,1\}, \quad \forall\, i,j \in C,\ i \neq j \tag{5.4g}$$
$$v_k \in \mathbb{R}^+, \quad \forall\, k \in \hat{N}\,. \tag{5.4h}$$

The variable v_k [40] represents the index of the first node of the route that visits node k (the route identifier of the node k). The variable u_{ij} is equal to 1 if and only if $v_i - v_j \geq 1$, 0 otherwise (this variable is introduced in order to linearise the not equals relationship expressed through the constraints (5.4e) and (5.4f)). Constraints (5.4a) and (5.4b) guarantee that the route identifier for the node k assumes the same index of the first visited node of the route. Constraints (5.4c) and (5.4d) guarantee that the index of the first visited node is forwarded to the next nodes in the route. Constraints (5.4e) and (5.4f) guarantee that all the logical incompatibilities are satisfied. Actually, for each pair of logical incompatible nodes i and j in C, the route identifier of i must be different from the route identifier of j (the not equals relationship $v_i \neq v_j$ was linearised).

5.4 An exact method

An exact method leveraging the cutting-plane methodology was developed to enhance the model by incorporating five distinct families of valid inequalities. This

cutting-plane algorithm is grounded in the flow-based formulation (5.1), which generally outperforms the route-based formulation (5.3) in terms of computational efficiency, as demonstrated in Section 5.5.1.

5.4.1 The valid inequalities

The directed graph $\overline{G} = (N, \overline{A})$, where $\overline{A} = \{(i,j) : \overline{x}_{ij} > 0\}$, is defined for every optimal relaxed solution $(\overline{x}, \overline{y}, \overline{z})$ of (5.3.2). Such a graph makes it easier to describe the valid inequalities. Throughout this section, $\overline{x}_{ij}$ represents the flow through the arc (i,j), and $\overline{y}_i$ denotes the flow passing inside the node i within the graph $\overline{G}$.

Route inequalities A relaxed solution of the flow-based formulation (5.3.2) allows the presence of infeasible routes whose travel time exceeds $T_{\max}$. The *Route Inequalities* (RIs) aims to cut off such routes from the search space:

$$\sum_{i=1}^{l-1} x_{r_i r_{i+1}} \leq \sum_{k=2}^{l-1} y_{r_k}, \tag{5.5}$$

where r_i represents the i^{th} node visited by r and $l = |r|$ is the cardinality of the route.

Set inequalities The *Set Inequalities* (SIs) could be considered a strong version of the RIs since they do not take into account the order of the nodes composing a route. A directed graph $\overline{G} = (N, \overline{A})$ is defined, with $\overline{A} = \{(i,j) : \overline{x}_{ij} > 0\}$, where $(\overline{x}, \overline{y}, \overline{z})$ represents the optimal relaxed solution of (5.3.2). For an infeasible route r, a directed subgraph $G(r) = (N(r), E(r))$ is constructed, where $N(r)$ denotes the nodes in route r, and $E(r)$ represents the arcs connecting these nodes.

To determine if a feasible route exists that visits all nodes in $N(r)$, the Helsgaun bound [53] is applied to $G(r)$, providing a lower bound δ_{LB} on the travel time of such a route. This bound, similar to the Held-Karp bound [52], is derived from the Lagrangian relaxation of the classical TSP formulation. The computation uses the subgradient method, employing a simple decreasing step size and adjusting the Lagrangian multipliers as outlined in Volgenant and Jonker [123]. The Helsgaun bound is utilised instead of the Held-Karp bound due to its lower computational cost, while maintaining the same level of solution quality.

If the lower bound δ_{LB} exceeds $T_{\max}$, no feasible route exists that can visit all nodes in $\hat{N}(r) = N(r) \setminus \{1, n\}$, which represents the set of customers served by route r. Under such circumstances, the following inequality can be applied:

$$\sum_{i \in \hat{N}(r)} \sum_{j \in \hat{N}(r)} x_{ij} \leq |\hat{N}(r)| - 2\,. \tag{5.6}$$

This cut imposes that the set of nodes in $\hat{N}(r)$ must be visited by, at least, two routes.

Subpath inequalities The path inequalities defined by Fischetti et al. [38] were adapted to leverage feasible subpaths, effectively eliminating infeasible routes. For a feasible subpath p of customers with cardinality l that belongs to an infeasible route r, a pair of Subpath Inequalities (SPIs) can be defined:

$$\sum_{i=1}^{l-1} x_{p_i p_{i+1}} - \sum_{k=2}^{l-1} y_{p_k} - \sum_{v \in L(p)} x_{v p_1} \leq 0, \tag{5.7}$$

$$\sum_{i=1}^{l-1} x_{p_i p_{i+1}} - \sum_{k=2}^{l-1} y_{p_k} - \sum_{v \in R(p)} x_{p_l v} \leq 0. \tag{5.8}$$

$L(p)$ and $R(p)$ represent the set of nodes that is possible to add at the beginning and at the end of the subpath p without making itself infeasible, respectively. Finally, p_i describes the i^{th} node visited by p.

These cuts imposes that the subpath p should be left-connected (right-connected) with a node belonging to $L(p)$ ($R(p)$). Their validity derives directly from the validity of the path inequalities.

Subtour elimination constraints The mathematical formulation outlined in Section 5.3.2 effectively prevents the formation of subtours and ensures route connectivity. However, additional *Subtour Elimination Constraints* (SECs) were incorporated to further strengthen the formulation:

$$\sum_{i \in U} \sum_{j \in U} x_{ij} \leq \sum_{i \in U} y_i - y_k, \quad \forall\, U \subseteq \{2, \ldots, n-1\},\ k \in U. \tag{5.9}$$

The additional SECs are exponential in the number of nodes, necessitating their dynamic separation during the branch-and-bound process. These constraints, an adaptation of the classical Dantzig-Fulkerson-Johnson SECs originally designed for the Traveling Salesman Problem (TSP), derive their validity from the foundational work presented in Dantzig et al. [29]. The following lemma is the formulated:

Lemma 5.4.1. *Any violated SEC* (5.9) *can only be identified in those sets associated with cycles in the graph $\overline{G}$. A subset of nodes U induces a graph in $\overline{G}$ with cycles if and only if the arcs involved with the nodes contained in U form one or more cycles in $\overline{G}$.*

Proof. To prove it, consider a subset of nodes U associated with no cycles in the graph $\overline{G}$. Now consider $\overline{G}_U$ as the subgraph of $\overline{G}$ induced by the set U. Until the end of this proof, $\overline{x}_{ij}$ and $\overline{y}_k$ as the flow passing through the arc (i, j) and through the node k, respectively. Thanks to the constraints (5.3c), it can be observed that for

each $k \in U$, the corresponding value $\overline{y}_k$ is at least large as the sum of the flow on the incoming arcs on the node k:

$$\sum_{i \in U} \sum_{j \in U} \overline{x}_{ij} \leq \sum_{i \in U} \overline{y}_i.$$

θ represents the flow corresponding to all arcs (i, j) with $i \notin U$ and $j \in U$:

$$\theta = \sum_{i \notin U} \sum_{j \in U} \overline{x}_{ij}.$$

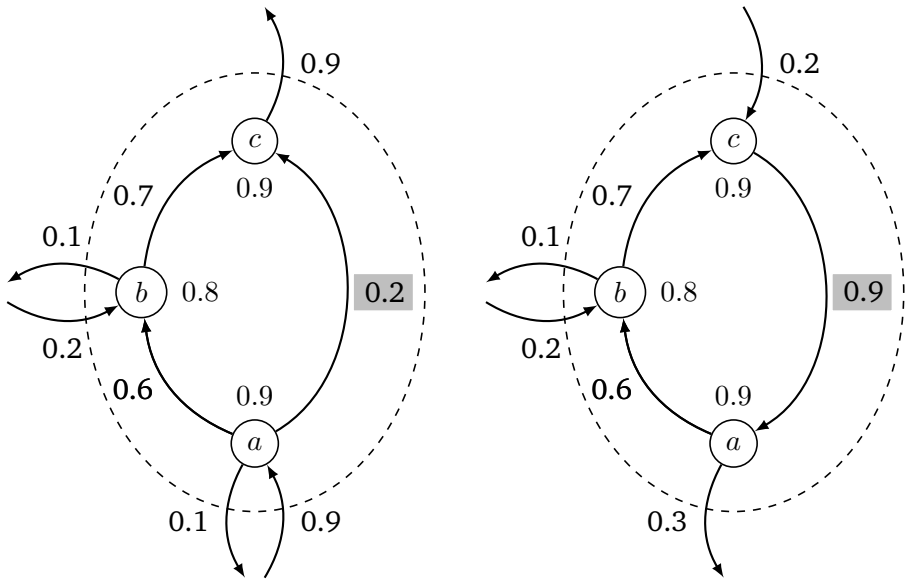
Hence, to balance the previous inequality, θ is added to the left part of it:

$$\sum_{i \in U} \sum_{j \in U} \overline{x}_{ij} + \theta = \sum_{i \in U} \overline{y}_i.$$

Since $\overline{G}_U$ does not contain any cycle, it can be deduced by simple flow conservation that $\theta \geq \overline{y}_k$ for each $k \in U$. Therefore, it can be stated that:

$$\sum_{i \in U} \sum_{j \in U} \overline{x}_{ij} = \sum_{i \in U} \overline{y}_i - \theta \leq \sum_{i \in U} \overline{y}_i - \overline{y}_k \quad k \in U,$$

which completes the proof. □



(a) Some violated SECs (b) No violated SECs

Fig. 5.1: Illustration of the lemma 5.4.1.

Figure 5.1 gives a graphic intuition of the Property 5.4.1. Figures 5.1(a) and 5.1(b) depict a portion of the graph $\overline{G}$ that only considers the nodes in U, which is composed of the three nodes a, b, c. The numbers close to the nodes and the arcs are the

(fractional) values of the flow. The dotted ellipses represent the boundary between U and the rest of the graph, which means that the ingoing flow through the dotted ellipse must be equal to the outgoing flow. This property is guaranteed by the constraints (5.3c). It is easy to verify that the lemma 5.4.1 holds in Figure 5.1(b) while not in Figure 5.1(a) due to the presence of a cycle. The difference between the flow values on the arcs (a, c) in Figure 5.1(a) and (c, a) in Figure 5.1(b) is highlighted, with the distinction marked by a dark grey rectangle. The application of the lemma 5.4.1 allows us to consider only the cycles inside $\overline{G}$ instead of taking into account the entire graph. The violated SECs can be separated efficiently using an algorithm for detecting elementary cycles, as described by Hawick and James [51]. Asymptotically, this separation method results to be faster than the classical one used in literature that runs the max-flow/min-cut algorithm (see, e.g., the algorithm inz Boykov and Kolmogorov [20]) for each node of the graph. Essentially, the SECs can be separated in $\mathcal{O}\left((|N| + |A|) \cdot \mu\right)$ instead of $\mathcal{O}\left(|N|^3 \cdot |A| \cdot \tau\right)$ where, μ represents the number of cycles inside $\overline{G}$ and τ, the cost of the maximum flow/minimum cut for the same graph.

Logical inequalities The mathematical formulation 5.3 allows the presence of routes that contain logical incompatibilities in a relaxed solution. The *Logical Inequalities* (LIs) have the same structure of the SIs and their aim is to cut off every route composed by a pair of logically incompatible nodes, at least. For a given route r, all subpaths p of customers within r are identified by checking whether the starting and ending nodes of p are logically incompatible. In such cases, the corresponding inequality can be considered:

$$\sum_{i \in \hat{N}(p)} \sum_{j \in \hat{N}(p)} x_{ij} \leq |\hat{N}(p)| - 2. \tag{5.10}$$

Even in this case, this cut imposes that the nodes contained in $\hat{N}(p)$ must be visited by two routes at least.

5.4.2 A cutting-plane algorithm

An exact algorithm for solving the TOP-ST-MIN is proposed, based on the cutting-plane methodology, which incorporates all the valid inequalities introduced earlier to enhance the linear programming formulation. This algorithm is referred to as CPA. The classic cutting-plane method works by iteratively refining the feasible region. At each iteration, valid linear inequalities are added, which are derived by solving the corresponding separation problem. These inequalities reduce the search space, and the process continues until an optimal integer solution is found. For each optimal relaxed solution, the graph $\overline{G}$ is built and the complete set of routes contained in it is represented by the set $\mathcal{R}$ that is computed by using a modified version of the Depth-First Search (DFS) algorithm.

Valid Inequalities First, the *Subpath inequalities* (5.7) are considered. To separate them, all infeasible routes are taken into account. $r \in R$ $(\delta_r > T_{\max})$ where, δ_r represents the sum of travel and service times of the arcs and nodes composing r. For each feasible subpath p of customers belonging to r, The associated subroute s is created by adding the source (or destination) node at the beginning (or end) of p. If the value of δ_s is less than or equal to $T_{\max}$, the sets $L(p)$ and $R(p)$ are constructed, and it is then checked whether the corresponding inequalities are violated. Afterward, the focus shifts to the *Set inequalities* (5.6) and the *Route inequalities* (5.5). To separate the *Set Inequalities*, the Helsgaun lower bound δ_{LB} is calculated on the subgraph $G(r)$, composed of the nodes in r. If δ_{LB} exceeds $T_{\max}$, the first inequality is checked for violation. Otherwise, the second inequality is examined for violation. Additionally, the *Logical inequalities* (5.7) are checked for violation when solving the TOP-ST-MIN-PL model. Finally, all the elementary cycles within $\overline{G}$ are examined, and the associated *Subtour Elimination Constraints* (5.9) are checked for violations.

Preprocessing Some instances may contain nodes and arcs that are unreachable due to the restricted time budget and incompatibilities. A node or arc is considered unreachable if including it in the route would render it infeasible. Apart from the physical incompatibilities already accounted for in the formulation, the sets U_n and U_e represent the unreachable set of nodes and arcs, respectively. These sets are populated by considering both the total route time and logical incompatibilities. A node or arc is deemed unreachable if adding it to a route causes the total travel time to exceed $T_{\max}$, or if the nodes involved in an arc are logically incompatible. If these sets are non-empty, the corresponding variables can be fixed accordingly:

$$y_k \leq 0, \quad \forall\, k \in U_n \quad \text{and} \quad x_{ij} \leq 0, \quad \forall\, (i,j) \in U_e.$$

Implementation The pseudocode of the CPA callback is reported in Algorithm 2 and makes use of the following procedures: (i) `buildGraph` builds the graph $\overline{G}$ from the relaxed solution; (ii) `buildRoutes` builds the set R of routes from the graph $\overline{G}$ with a variant of the DFS algorithm; (iii) `buildSubRoute` builds the corresponding subroute from the subpath p; (iv) `buildSubGraph` builds the corresponding the sub-graph $G(r)$ from the nodes contained in the route r; (v) `calculateTravelTimeBound` calculate the Helsgaun lower bound for the sub-graph $G(r)$; (vi) `buildSets` builds the corresponding sets $L(p)$ and $R(p)$ from the subpath p. The overall CPA callback has been developed as an embedded procedure inside the CPLEX environment, which is called for every branching node with an optimal fractional solution.

Algorithm 2: The CPA callback.

Data: N, $T_{\max}$, t_{ij}, with $i, j \in \{1, \ldots, n\}$, s_k, with $k \in \{2, \ldots, n-1\}$, $\overline{x}$

```
1  G̅ ← buildGraph(x̅) ;
2  R ← buildRoutes(G̅, 1, n) ;
3  for route r ∈ R(x̅) do
4      if δ_r > T_max then
5          for subpath p ∈ r do
6              if 1 ∉ p and n ∉ p then
7                  s ← buildSubRoute(p) ;
8                  if δ_s ≤ T_max then
9                      L(p), R(p) ← buildSets(p, N, T_max) ;
10                     check inequality (5.7) associated with p and L(p) ;
11                     check inequality (5.8) associated with p and R(p) ;
12         G(r) ← buildSubGraph(r, t_ij, s_k) ;
13         δ_LB ← calculateTravelTimeBound(G(r)) ;
14         if δ_LB > T_max then
15             check inequality (5.6) associated with N̂(r) ;
16         else
17             check inequality (5.5) associated with r ;
18     for subpath p ∈ r do
19         if 1 ∉ p and n ∉ p then check inequality (5.10) associated with p ;
20 for cycle c ∈ G̅ do
21     check inequalities (5.9) associated with c ;
```

5.5 Quantitative analysis

This section presents an extensive quantitative analysis to validate the proposed approach and offers insights into the structures of the newly generated instances. Results are provided for both TOP-ST-MIN-P and TOP-ST-MIN-PL.

Initially, a computational comparison of the two formulations introduced in Section 5.3 (Sect. 5.5.1) is provided. Following that, the impact of the valid inequalities discussed in Section 5.4.1 (Sect. 5.5.2) is evaluated. The results of the CPA proposed in Section 5.4.2 are reported in Section 5.5.3. Finally, the results of an adaptation of the CPA to solve the TOP is compared with the best known results of the literature (Sect. 5.5.4).

The CPA has been implemented in C++ using CPLEX 22.1.1 Concert Technology, run in single-threaded mode for the TOP-ST-MIN analysis and in multi-threaded mode for the TOP one. The code was compiled in Scientific Linux 7.9 (Nitrogen). The experiments were carried out on a 64-bit Linux machine, with a Six-Core AMD Opteron(tm) Processor 8425 HE, 2.10 GHz, and 24 GB of RAM. CPLEX built-in cuts have been used in all experiments. CPLEX is run in a fully-deterministic mode to

ensure the complete reproducibility of the results. All the computational experiments have been run by setting a time limit of 7200 seconds as many authors in the literature (see, e.g., Hanafi et al. [49] and Assunção and Mateus [11]).

5.5.1 Comparing route-based and flow-based formulations

The main purpose of this section is to compare the route-based formulation (Sect. 5.3.1) and the flow-based formulation (Sect. 5.3.2) from a computational point of view in order to validate the choice of using the flow-based formulation in the proposed solution approach. The comparisons have been obtained by using CPLEX as a general purpose solver.

Table 5.2 reports a brief comparison between the two models for the TOP-ST-MIN-P (formulations (5.1) and (5.3)) considering the 156 instances belonging to the small benchmark set. The results are categorised based on the number of available vehicles m (ranging from 2 to 4) to assess the performance of both models. The column OPT reports the number of instances optimally solved, CPU(s) the average running time (in seconds) used by CPLEX to certify the optimality, and NODES the average number of branching nodes. Finally, the columns Δ_{CPU} and Δ_{NODES} describe the average relative differences between the two models (with respect to the flow-based formulation) for the running time and branching nodes, respectively.

Tab. 5.2: The comparison between the route-based and the flow-based formulations for TOP-ST-MIN-P.

		Mixed formulation			**Compact formulation**			**Rel. Differences**	
m	#	**OPT**	**CPU(s)**	**NODES**	**OPT**	**CPU(s)**	**NODES**	Δ_{CPU}	Δ_{NODES}
2	52	52	203.77	10520	52	40.76	6403	-399.89	-64.29
3	52	52	505.52	14234	52	25.72	3376	-1865.45	-321.57
4	52	51	348.62	6848	52	4.94	1066	-6955.07	-542.63

The results reported in Table 5.2 show that the flow-based formulation dominates the route-based formulation when solving the TOP-ST-MIN-P. In fact, even if the number of optimally solved instances is almost the same, the gaps in terms of running time and explored nodes are in favour of the flow-based formulation and tend to grow as soon as the number of routes increases.

Tab. 5.3: The comparison between the route-based and the flow-based formulations for TOP-ST-MIN-PL.

		Mixed formulation			**Compact formulation**			**Rel. Differences**	
m	#	**OPT**	**CPU(s)**	**NODES**	**OPT**	**CPU(s)**	**NODES**	Δ_{CPU}	Δ_{NODES}
2	104	103	118.67	4665	97	583.76	41467	79.67	88.75
3	104	103	249.10	7224	103	160.61	10973	-55.10	34.17
4	104	104	229.79	5262	104	51.79	4838	-343.75	-8.78

Table 5.3 reports the results of the comparison for the TOP-ST-MIN-PL (formulations (5.1)+(5.2), and (5.3)+(5.4)) for which the dominance is less evident than those for the TOP-ST-MIN-P. As a matter of fact, the route-based formulation obtains better results for instances with only 2 available vehicles ($m = 2$). On the contrary, the flow-based formulation produces better results as soon as the number of routes increases. This behaviour may be explained by the larger number of variables in the route-based formulation due to the presence of three-index variables x_{ijr}.

5.5.2 Evaluating the impact of the valid inequalities

Before analysing the overall results of the CPA, an evaluation of the valid inequalities was conducted in order to assess their impact on the solution process, as outlined in Section 5.4.1. The tests were carried out on the small benchmark set. The column OPT reports the number of instances optimally solved, CPU(s) the average CPU time (in seconds) spent by the algorithm to certify the optimality, NODES the average number of explored branching nodes, and GAP(%) the average mip gap of the unsolved instances. The first five rows report the results obtained using a single family of inequalities while the last two rows report respectively the results using all inequalities and none (CPLEX with only built-in cuts).

Tab. 5.4: The impact of the valid inequalities for the TOP-ST-MIN-P and the TOP-ST-MIN-PL.

	TOP-ST-MIN-P				TOP-ST-MIN-PL			
Valid inequalities	**OPT**	**CPU(s)**	**NODES**	**GAP(%)**	**OPT**	**CPU(s)**	**NODES**	**GAP(%)**
RIs (5.5)	156	24.11	3219	-	302	416.11	23957	7.73
SIs (5.6)	156	26.97	3556	-	302	440.03	28224	6.83
SPIs (5.7) and (5.8)	156	26.45	3362	-	305	386.63	25314	5.09
SECs (5.9)	156	18.11	1639	-	302	446.78	23206	8.48
LIs (5.10)	–	–	–	–	312	77.47	5287	-
CPA (5.4.2)	156	17.50	1765	-	312	59.61	3254	-
NONE (CPLEX)	156	24.16	3683	-	304	257.01	18511	5.88

The results reported in Table 5.4 clearly show that the separation scheme adopted in the CPA produces the best results in terms of running time and number of explored nodes. In contrast, when each valid inequality is considered individually, the *Subtour Elimination Constraints* for TOP-ST-MIN-P and the *Logical Inequalities* for TOP-ST-MIN-PL appear to perform quite effectively on their own. In both cases, these valid inequalities are those that notably reduce the number of branching nodes explored.

It is worth noting that all the remaining valid inequalities, when used singularly, increase the number of explored nodes compared to CPLEX with only built-in cuts. Furthermore, the *Logical Inequalities* are the only inequalities that actually allow to solve all the TOP-ST-MIN-PL instances to optimality.

5.5.3 Results on new benchmark instances

The main goal of this section is to present a detailed computational analysis of the CPA applied to the new benchmark instance sets, which were generated as described in Chapter 4. The results for TOP-ST-MIN-P are reported first, followed by those for TOP-ST-MIN-PL. Table 5.5 presents the outcomes for TOP-ST-MIN-P. The results are

Tab. 5.5: The TOP-ST-MIN-P results of the CPA.

			OPT			NO OPT			INF			NO SOL	
		#	#	CPU(s)	NODES	#	NODES	GAP(%)	#	CPU(s)	NODES	#	NODES
SMALL	**ALL**	**156**	**156**	**17.50**	**1765**	**0**	**-**	**-**	**0**	**-**	**-**	**0**	**-**
	CMN	78	78	29.58	2959	0	-	-	0	-	-	0	-
	SMN	78	78	5.41	571	0	-	-	0	-	-	0	-
	CPI	78	78	15.71	1620	0	-	-	0	-	-	0	-
	DPI	78	78	19.28	1911	0	-	-	0	-	-	0	-
MEDIUM	**ALL**	**120**	**108**	**655.12**	**18148**	**7**	**96676**	**4.00**	**2**	**20.07**	**0**	**3**	**43334**
	CMN	60	54	865.84	26350	6	106596	3.34	0	-	-	0	-
	SMN	60	54	444.41	9946	1	61954	6.33	2	20.07	0	3	43334
	CPI	60	55	590.02	17155	3	91326	5.85	1	22.20	0	1	50396
	DPI	60	53	722.68	19178	4	99350	3.08	1	17.94	0	2	36272
LARGE	**ALL**	**120**	**1**	**1607.82**	**43073**	**49**	**32541**	**5.12**	**17**	**6.67**	**0**	**53**	**12599**
	CMN	60	1	1607.82	43073	41	34101	4.92	0	-	-	18	10991
	SMN	60	0	-	-	8	17467	7.13	17	6.67	0	35	13261
	CPI	60	0	-	-	24	41693	5.23	8	6.60	0	28	13031
	DPI	60	1	1607.82	43073	25	26279	5.05	9	6.73	0	25	12043
ALL		**396**	**265**	**283.36**	**8598**	**56**	**46620**	**4.88**	**19**	**8.15**	**0**	**56**	**13828**

organised into various solution categories for easier interpretation and comparison. The columns OPT describe the statistics related to the instances optimally solved. The columns NO OPT describe the statistics related to the instances for which a solution has been found without proving its optimality. The columns INF describe the statistics related to the instances proven to be infeasible. Finally, the columns NO SOL describe the statistics related to the instances for which no solution has been found without proving their infeasibility either. For each of them, the column # reports the number of instances in that category. The column CPU(s) reports the average running time (in seconds) to certify the optimality or the infeasibility, NODES the average number of branching nodes explored, and GAP(%) the average mip gap of the unsolved instances.

The results are grouped by benchmark set and then by feature generation methods. It is important to remark that each benchmark set has been partitioned by feature generation methods since, given a specific feature, all the related generation methods described in Chapter 4 create a partition of the benchmark sets. For example, taking into account the mandatory nodes, the CMN and SMN generation methods induce a partition on the set of instances considered. As a consequence, the instances considered in rows CMN and SMN on the one hand, and CPI and DPI on the other are the same, but partitioned differently. Table 5.6 reports a comparison between CPLEX and the CPA. The table presents the same structure of Table 5.5 but with

a slightly different meaning. The columns Δ_{CPU} and Δ_{NODES} report the average relative differences (in percentage), Δ_{SOL} the absolute difference, and GAP(%) the average absolute differences (in percentage). A negative value indicates that the CPA outperforms CPLEX in terms of CPU time, number of nodes, and mip gap difference. On the contrary, a positive value means that the CPA is actually better than CPLEX for the number of optimal solutions found. As reported in Table 5.5, the instances with scattered mandatory nodes (SMN) appear easier to solve to optimality compared to those with clustered mandatory nodes (CMN). As a matter of fact, in addition to the number of solved instances, the running time and the number of explored nodes are smaller for the SMN instances. Such differences are more evident for medium size instances.

This behaviour could be explained by the following facts. On one hand, the distances between mandatory nodes are expected to be large when they are scattered. On the other, if the mandatory nodes are clustered, the distances between them will tend to be smaller. This could imply that a single route may visit more mandatory nodes in those instances with clustered mandatory nodes. Thus, the number of feasible solutions could be greater than those in instances with scattered mandatory nodes, which makes them harder to solve to optimality (but easier in terms of finding a feasible solution). With regard to the physical incompatibilities, there are no huge differences between the clusters-based (CPI) and degree-based (DPI) instances. The results of the comparison between the CPA and CPLEX reported in Table 5.6 show that the CPA outperforms CPLEX, which nevertheless seems a little bit faster than the algorithm to certify the infeasibility of an instance. It can be even observed that the time required to certify optimality and the quality of the gaps for unsolved instances consistently favour the CPA, often by a significant margin. It is worth noting that

Tab. 5.6: The comparison between the CPA with CPLEX for the TOP-ST-MIN-P.

			OPT			NO OPT			INF			NO SOL	
		#	Δ_{SOL}	Δ_{CPU}	Δ_{NODES}	Δ_{SOL}	Δ_{NODES}	Δ_{GAP}	Δ_{SOL}	Δ_{CPU}	Δ_{NODES}	Δ_{SOL}	Δ_{NODES}
SMALL	**ALL**	**156**	**0**	**-38.06**	**-108.66**	**0**	**-**	**-**	**0**	**-**	**-**	**0**	**-**
	CMN	78	0	-42.32	-120.80	0	-	-	0	-	-	0	-
	SMN	78	0	-14.77	-45.74	0	-	-	0	-	-	0	-
	CPI	78	0	-43.10	-121.11	0	-	-	0	-	-	0	-
	DPI	78	0	-33.96	-98.10	0	-	-	0	-	-	0	-
MEDIUM	**ALL**	**120**	**0**	**-21.93**	**-80.32**	**1**	**-82.79**	**-0.46**	**0**	**4.68**	**0.00**	**-1**	**-133.37**
	CMN	60	0	-34.73	-98.60	1	-70.58	-0.52	0	-	-	-1	-
	SMN	60	0	-3.00	-31.88	0	-156.30	-0.23	0	4.68	0.00	0	-133.37
	CPI	60	0	-23.97	-80.67	2	-81.62	-0.47	0	4.95	0.00	-2	-117.41
	DPI	60	0	-20.20	-79.99	-1	-83.32	-0.45	0	4.35	0.00	1	-155.54
LARGE	**ALL**	**120**	**0**	**-63.05**	**-86.78**	**8**	**-37.64**	**-0.99**	**1**	**5.09**	**-**	**-9**	**-46.33**
	CMN	60	0	-63.05	-86.78	6	-37.56	-1.07	0	-	-	-6	-53.72
	SMN	60	0	-	-	2	-39.06	-0.30	1	5.09	-	-3	-
	CPI	60	0	-	-	8	-41.12	-1.08	0	6.63	-	-8	-
	DPI	60	0	-63.05	-86.78	0	-33.86	-0.94	1	3.58	-	-1	-41.91
ALL		**396**	**0**	**-23.40**	**-83.87**	**9**	**-58.19**	**-0.88**	**1**	**4.98**	**-**	**-10**	**-57.24**

Tab. 5.7: The TOP-ST-MIN-PL results of the CPA.

			OPT			NO OPT			INF			NO SOL	
		#	#	CPU(s)	NODES	#	NODES	GAP(%)	#	CPU(s)	NODES	#	NODES
SMALL	**ALL**	**312**	**312**	**59.61**	**3254**	**0**	**-**	**-**	**0**	**-**	**-**	**0**	**-**
	CMN	156	156	99.95	5453	0	-	-	0	-	-	0	-
	SMN	156	156	19.27	1055	0	-	-	0	-	-	0	-
	CPI	156	156	59.99	3154	0	-	-	0	-	-	0	-
	DPI	156	156	59.23	3354	0	-	-	0	-	-	0	-
	FLI	156	156	55.05	2845	0	-	-	0	-	-	0	-
	NLI	156	156	64.42	3686	0	-	-	0	-	-	0	-
MEDIUM	**ALL**	**240**	**137**	**807.26**	**11377**	**90**	**52525**	**5.50**	**0**	**-**	**-**	**13**	**25647**
	CMN	120	61	1127.40	15841	59	56243	5.25	0	-	-	0	-
	SMN	120	76	581.98	8236	31	43312	6.11	0	-	-	13	25647
	CPI	120	62	887.98	13219	51	56532	5.32	0	-	-	7	24835
	DPI	120	75	733.26	9689	39	47628	5.71	0	-	-	6	26621
	FLI	120	84	978.92	13588	29	53488	3.90	0	-	-	7	24316
	NLI	120	53	226.86	3903	61	52087	6.22	0	-	-	6	26755
LARGE	**ALL**	**240**	**1**	**-**	**-**	**51**	**17964**	**12.21**	**44**	**39.78**	**19**	**144**	**5724**
	CMN	120	1	-	-	49	17964	12.21	0	-	-	70	5508
	SMN	120	0	-	-	2	-	-	44	39.78	19	74	5910
	CPI	120	0	-	-	26	21449	12.26	22	62.58	37	72	5889
	DPI	120	1	-	-	25	14478	12.16	22	15.84	0	72	5559
	FLI	120	1	-	-	44	16876	10.92	19	13.46	0	56	6194
	NLI	120	0	-	-	7	32100	29.01	25	60.38	34	88	5457
ALL		**792**	**450**	**233.31**	**5141**	**141**	**40627**	**7.81**	**44**	**39.78**	**19**	**157**	**7195**

the CPA is capable to find 9 more solutions and to certify 1 more infeasibility than CPLEX (this can be seen directly from the NO OPT and INF columns).

In both tables, the time needed to certify the optimality and the quality of the gaps provided for the instances not solved, are systematically in favour of the CPA,

Tab. 5.8: The comparison between the CPA with CPLEX for the TOP-ST-MIN-PL.

			OPT			NO OPT			INF			NO SOL	
		#	Δ_{SOL}	Δ_{CPU}	Δ_{NODES}	Δ_{SOL}	Δ_{NODES}	Δ_{GAP}	Δ_{SOL}	Δ_{CPU}	Δ_{NODES}	Δ_{SOL}	Δ_{NODES}
SMALL	**ALL**	**312**	**8**	**-331.15**	**-468.83**	**-8**	**-**	**-**	**0**	**-**	**-**	**0**	**-**
	CMN	156	4	-289.25	-410.68	-4	-	-	0	-	-	0	-
	SMN	156	4	-548.48	-769.31	-4	-	-	0	-	-	0	-
	CPI	156	4	-296.99	-478.89	-4	-	-	0	-	-	0	-
	DPI	156	4	-365.74	-459.36	-4	-	-	0	-	-	0	-
	FLI	156	0	-17.95	-59.09	0	-	-	0	-	-	0	-
	NLI	156	8	-613.26	-802.20	-8	-	-	0	-	-	0	-
MEDIUM	**ALL**	**240**	**44**	**-40.43**	**-91.34**	**-32**	**-58.63**	**-5.78**	**0**	**-**	**-**	**-12**	**-134.00**
	CMN	120	22	-47.80	-107.96	-19	-57.83	-6.48	0	-	-	-3	-
	SMN	120	22	-30.38	-68.83	-13	-61.20	-4.06	0	-	-	-9	-134.00
	CPI	120	17	-33.20	-80.19	-10	-57.61	-5.53	0	-	-	-7	-160.70
	DPI	120	27	-48.45	-105.28	-22	-60.11	-6.09	0	-	-	-5	-104.10
	FLI	120	12	-23.67	-60.82	-10	-58.65	-0.46	0	-	-	-2	-129.58
	NLI	120	32	-284.88	-450.47	-22	-58.62	-8.20	0	-	-	-10	-137.34
LARGE	**ALL**	**240**	**1**	**-**	**-**	**2**	**-19.36**	**-2.44**	**3**	**-83.71**	**-245.36**	**-6**	**-54.80**
	CMN	120	1	-	-	0	-19.36	-2.44	0	-	-	-1	-70.45
	SMN	120	0	-	-	2	-	-	3	-83.71	-245.36	-5	-42.18
	CPI	120	0	-	-	2	-19.72	-3.12	1	-101.77	-245.36	-3	-58.82
	DPI	120	1	-	-	0	-18.82	-1.76	2	-8.81	-	-3	-50.53
	FLI	120	1	-	-	-2	-17.97	-1.69	1	-4.10	-	0	-37.65
	NLI	120	0	-	-	4	-28.80	-12.23	2	-97.60	-245.36	-6	-65.86
ALL		**792**	**53**	**-97.45**	**-274.76**	**-38**	**-52.65**	**-4.63**	**3**	**-83.71**	**-245.36**	**-18**	**-75.64**

sometimes by a large amount. Tables 5.7 and 5.8 are respectively the equivalent of Tables 5.5 and 5.6 but for TOP-ST-MIN-PL.

The results reported in Table 5.7 confirm also for TOP-ST-MIN-PL the remarks made on the mandatory nodes (CMN and SMN) and on the physical incompatibilities (CPI and DPI) for the TOP-ST-MIN-P. Considering the logical incompatibilities, the instances generated with farthest logical incompatibilities (FLI) seem easier than the instances with nearest logical incompatibilities (NLI). Indeed, the CPA is faster to solve the FLI instances on the small benchmark set, while it solves a larger number of FLI instances to optimality on the medium benchmark set. Finally, on the larger benchmark set where almost no instance is solved to optimality, it manages to find more feasible solutions on the FLI than the NLI. Due to the large number of NO SOL, one should take the last remark with a grain of salt.

The results reported in Table 5.8 confirm the fact that the CPA outperforms CPLEX for TOP-ST-MIN-PL, even more so than for TOP-ST-MIN-P. As a matter of fact, the CPA finds 53 additional optimal solutions, certifies 3 additional infeasibilities, and finds 18 additional solutions compared to CPLEX, all the while using approximately half of the CPU time and exploring almost a quarter of the nodes. Finally, it is important to remark that the time needed to certify the optimality and the quality of the gaps provided for the instances unsolved are systematically in favour of the CPA, sometimes by a large amount.

5.5.4 Results for TOP instances

This section aims to validate the CPA on the TOP by comparing it with state-of-the-art algorithms for the problem, specifically those described in Assunção and Mateus [11] and Hanafi et al. [49], referred to as AM and HMZ, respectively. The CPA was evaluated using the classical TOP benchmark set, running a version of the algorithm with all TOP-ST-MIN-specific elements disabled and without any additional customisations.

Tab. 5.9: The comparison between the CPA and the competitors on the TOP instances. Bold entries highlight the best algorithm with respect to the number of solved instances. When the number of solved instances is the same for all algorithms, bold entries highlight the best results in terms of running time and mip gap.

		AM vs CPA						HMZ vs CPA					
Set	**#**	**OPT**		**CPU(s)**		**GAP(%)**		**OPT**		**CPU(s)**		**GAP(%)**	
1	54	54	54	3.11	**1.23**	-	-	54	54	**0.98**	1.23	-	-
2	33	33	33	0.22	**0.07**	-	-	33	33	0.13	**0.07**	-	-
3	60	60	60	165.59	**63.04**	-	-	60	60	**37.83**	63.04	-	-
4	60	44	**50**	2479.98	2731.99	3.19	1.97	**48**	46	406.89	1578.95	1.48	2.45
5	78	62	62	715.75	**630.19**	3.01	**2.74**	62	62	**324.20**	630.19	**2.62**	2.93
6	42	**42**	41	475.26	489.39	-	0.88	39	**41**	100.11	489.39	0.81	1.76
7	60	47	**49**	1133.23	1252.87	1.94	1.59	**50**	48	464.98	1106.88	1.47	1.93
ALL	387	342	**349**	692.49	785.87	2.76	1.79	**346**	344	190.73	545.35	1.60	2.27

The results reported in Assunção and Mateus [11] are obtained running the algorithm on a 4.0 GHz Intel Core i7-4790k while a 3.50 GHz Intel i7 5930k processor is used in Hanafi et al. [49]. Both algorithms were run using 6 threads and a time limit of 7200 seconds. To enable a fairer comparison, considering the CPU frequency of 2.1 GHz, the CPA was executed twice rescaling the time limit by the factor:

$$\frac{\alpha}{2.1},$$

where α is the cpu frequency of the corresponding competitor. Same logic was applied to the running times of both competitors.

Table 5.9 reports the results of such a comparison. The column OPT reports the number of optimally solved instances, CPU(s) the average running time to certify the optimality, and GAP(%) the average mip gap of the unsolved instances. Each of these columns report on the right the results for the CPA while on the left the results for the algorithm competitor. As can be seen from the Table 5.9, the CPA is competitive with respect to the two state-of-the-art algorithms. With respect to the AM, the algorithm computes more optimal solutions on the sets 4 and 7, and less on the set 6. With respect to the HMZ, the algorithm computes more optimal solutions on the set 6 and less on the sets 4 and 7. Overall, the CPA computes 7 more optimal solutions than AM and 2 less than HMZ. The CPA average running time is the same for both tests except for Set 4 due to the 5 additional optimal solutions while the average mip gap decreases. Overall, the CPA seems a little bit slower than AM and clearly slower than HMZ.

Partial results are finally reported for the Branch-Cut-and-Price approach presented in Pessoa et al. [96], only for Set 4. This method successfully solves 55 out of 60 instances within one hour of runtime on an Intel Xeon E5-2680 v3 processor operating at 2.50 GHz.

5.6 Heuristic approaches

This section presents three algorithms developed to solve the two versions of the TOP-ST-MIN, namely the TOP-ST-MIN-P and the TOP-ST-MIN-PL. The algorithms share the same structure but only differ in the feasibility check of candidate solutions, since the TOP-ST-MIN-PL also requires to check for the logical incompatibilities. The description provided here focuses on the version with logical incompatibilities, the TOP-ST-MIN-PL. It is also important to note that finding a feasible solution is NP-complete for both problem versions, as demonstrated in Section 5.2, where the proof relies on the interaction between mandatory nodes and physical incompatibilities.

The first algorithm (Section 5.6.1) is a procedure which tries to quickly find a feasible initial solution, if it exists. The second algorithm (Section 5.6.2) is a

metaheuristic approach based on the variable neighbourhood descent methodology. It uses different neighbourhoods in order to improve the overall profit of a solution or to decrease the cost of a route when exploring infeasible solutions. The third algorithm (Section 5.6.3) consists in a matheuristic approach that dynamically separates the subtour elimination constraints and runs the metaheuristic method as a procedure to fix and improve the current solution.

To simplify the description of the above algorithms, the set $\hat{N} = \{2, \dots, n-1\}$ is introduced in order to represent the set of *customers* and the set $\hat{A} = \{(i,j) \in A : i \in \hat{N} \cup \{1\},\ j \in \hat{N} \cup \{n\},\ i \neq j\}$ to represent the set of *traversable* arcs. To each node $k \in \hat{N}$ is associated a non-negative amount of service time s_k and a profit p_k. Each arc $(i,j) \in \hat{A}$ has a non-negative travel time t_{ij} across the node i and j. Let c_r be the cost of a route r, which consists in the sum of s_k and t_{ij} of the nodes and arcs belonging to the route r. Moreover, the following three sets are introduced: let $M \subset \hat{N}$, $I \subset \hat{A}$, $C \subset \hat{N} \times \hat{N}$ be the set of mandatory nodes, the set of physical incompatibilities, and the set of logical incompatibilities, respectively.

5.6.1 The mandatory nodes allocation algorithm

As demonstrated in Section 5.2, the feasibility problem of the TOP-ST-MIN is NP-complete, therefore there is no guarantee to always find a feasible solution for the problem in a short amount of time. Given that the complexity arises from the presence of mandatory nodes, the focus is placed on the set M of mandatory nodes.

Let g_k be the number of remaining routes in which the mandatory node $k \in M$ could be inserted while respecting the logical incompatibilities: the higher the score, the easiest it is to find how to insert the node in a potential route. The Mandatory Nodes Allocation Algorithm (MNAA) starts by defining the set U of unassigned mandatory nodes ($U = M$ at the beginning) setting also $g_u = m$ for each node $u \in U$. At each iteration, the MNAA tries to assign a node in U to some route. In order to maximise the possibility of finding a feasible solution that includes all the mandatory nodes, the MNAA computes the subset $U_{\overline{g}}$ of unassigned mandatory nodes associated with the minimal score $\overline{g} = \min\{g_u : u \in U\}$. To break ties among the nodes in $U_{\overline{g}}$, the MNAA also computes the next minimal score $\hat{g}_u$ for each node $u \in U_{\overline{g}}$. Thus, the MNAA inserts the node $q = \arg\max_{u \in U_{\overline{g}}} \hat{g}_u$ in solution, i.e., the one that maximises the next minimal score. The insertion of a mandatory node could require also the insertion of further optional nodes to avoid the violation of the physical incompatibilities constraints. Let h_o^q be the smallest subset of non-mandatory nodes that allow the insertion of q, which is then inserted in the solution in such a way to minimise the total service and travelling time. After each insertion, the MNAA updates the scores of the unassigned mandatory nodes. If there is at least one mandatory node that

cannot be inserted in any route due to physical or logical incompatibilities ($\overline{g} = 0$), the procedure stops returning an infeasibility flag.

At the end of the MNAA, the solution could still be infeasible because of the violation of the time budget constraint. In the framework, this kind of solution is accepted since the metaheuristic approach, described in the following section, is able to handle it and to regain feasibility.

Algorithm 3: `Mandatory Nodes Allocation Algorithm`

Result: Solution S

$S \leftarrow \emptyset$; $U \leftarrow M$; $g \leftarrow m$; $g_{\min} \leftarrow m$;
$S \leftarrow$ `initialiseSolution` (S, m) ;
while U is not empty and $g_{\min} > 0$ **do**
  $U_{g_{\min}} \leftarrow$ `getMinimalScoreNodes` $(g, g_{\min})$;
  $q \leftarrow$ `selectMandatoryNode` $(U_{g_{\min}})$;
  $h_o^q, r, p \leftarrow$ `getInsertionList` (q, O) ;
  $S \leftarrow$ `updateSolution` (S, h_o^q, r, p) ;
  $g \leftarrow$ `updateScores` (g) ;
  $g_{\min} \leftarrow \min_{u \in U} g$;
  $U \leftarrow U \setminus \{q\}$;
if U is not empty **then**
  return $\emptyset$;
return S;

We report the MNAA pseudocode in Algorithm 3 that makes use of the following procedures: (i) `initialiseSolution`: initialises the routes with m empty routes (i.e., with only the source and destination nodes), (ii) `getMinimalScoreNodes`: calculates the subset of unassigned mandatory nodes associated with the minimal score $\overline{g}$, (iii) `selectMandatoryNode`: selects the mandatory node to insert into a route that maximises the next minimal score $\overline{g}$ over the subset $U_{\overline{g}}$, (iv) `getInsertionList`: returns the list of nodes h_o^q to insert into the route r in the position p, (v) `updateSolution`: updates the current solution, and (vi) `updateScores`: update the scores of each unassigned nodes.

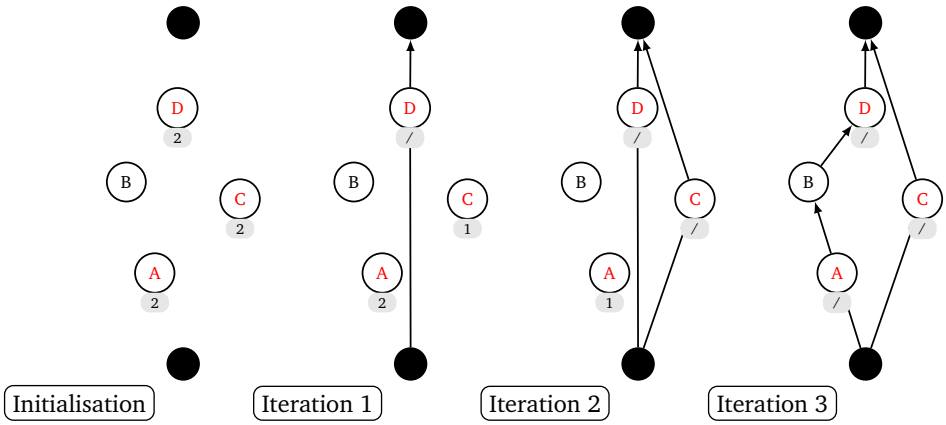


Fig. 5.2: Algorithm MNAA: small example for problem with 2 routes.

Figure 5.2 depicts how the algorithm works showing the result of the initialisation and three iterations. The figure includes three mandatory nodes (highlighted in red): A, C, and D along with a single optional node B. Note that the value of $\overline{g}$ of a mandatory node is reported in the small grey circle. The initialisation assigns the $\overline{g}$ scores to each mandatory node. In the first iteration, all nodes have the same score and therefore the MNAA considers the one associated with the minimal insertion cost (in the example, node D). Then, the score of node C is decreased by 1 since C is logically incompatible with D. In the second iteration, node C is visited by another route because it has the lowest score. Then, the score of node A is then reduced by 1 because it is logically incompatible with C. Finally, in the last iteration, node A is visited by the first route through node B due to a physical incompatibility between nodes A and D. This stops its computation since all mandatory nodes have been visited without violating any incompatibility.

5.6.2 A variable neighbourhood descent

The VND is a deterministic, systematic and efficient local search-based metaheuristic that improves upon the basic local search strategy by exploiting a sequence of different neighbourhood structures [32]. The aim is to find a local optimum with respect to one neighbourhood structure before moving on to explore another, potentially leading to better local optima.

We developed a VND based on two families of neighbourhoods (profit-based and cost-based) that uses a couple of variable-size Tabu-Lists (TLs) to fix some nodes inside or outside the solution for a given amount of iterations. In the rest of this section, all the necessary components, such as the neighbourhood families and the searches in both feasible and infeasible regions, are introduced, which will later form the basis of the general VND framework.

Neighbourhoods

We consider two families of neighbourhoods with different goals. The profit-based neighbourhoods try to increase the profits of the routes while the cost-based neighbourhoods try to reduce the cost, that is the total travel time of the routes.

The profit-based neighbourhoods

InsertNode The aim is to increase the profit of a route, and therefore of the whole solution, by inserting a non assigned node. Such a node is identified by exploring the entire set of insertions then the size of this neighbourhood is $\mathcal{O}(n^2)$. The best node is then inserted in such a way as to minimise its insertion cost. Figure 5.3 depicts an example.

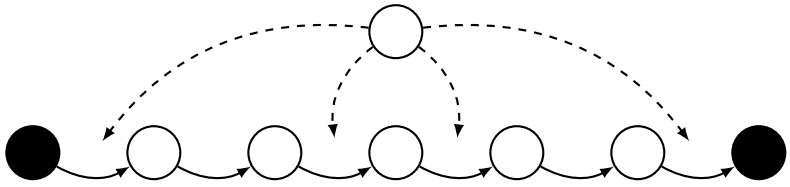


Fig. 5.3: INSERTNODE: normal arcs denote the route while dashed arcs indicate feasible insertions.

SwapNodes The goal is to increase the profit by swapping a pair of nodes (one in solution and one not) in a route. This neighbourhood therefore identifies the best swap between each pair of nodes and, to break the tie, the swap that yields the smallest cost increase (or even reduces it) is selected. The size of this neighbourhood is $\mathcal{O}(n^2)$. Figure 5.4 depicts an example.

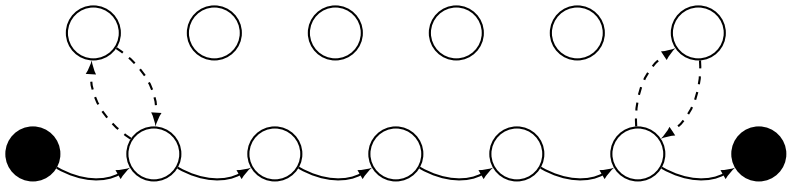


Fig. 5.4: SWAPNODES: normal arcs denote the route while dashed arcs indicate possible swaps.

The cost-based neighbourhoods

MoveNode The goal is to reduce the cost of a route by moving a node from one position to another in the same route. To determine the best pair (node, position), that is, the one with the maximal cost reduction, the neighbourhood explores all possible node moves and, therefore, its size is $\mathcal{O}(n^2)$. Figure 5.5 depicts an example.

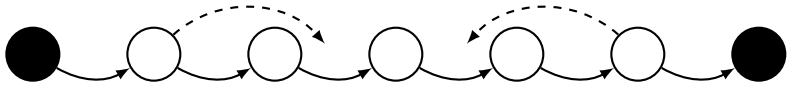


Fig. 5.5: MOVENODE: normal arcs denote the route while dashed arcs indicate possible moves.

MoveNodeRoutes The goal is to reduce the cost by moving a node to another route into a specific position. To determine the best pair (node, position), that is the one with the maximal cost reduction, the neighbourhood explores all possible node moves and, therefore, its size is $\mathcal{O}(n^2)$. Figure 5.6 depicts an example.

SwapNodesRoutes The aim of this neighbourhood is to reduce the cost of the solution by swapping a pair of nodes belonging to two different routes, maintaining therefore a constant total profit. The best swap is selected by exploring all possible swaps and, therefore, the size of this neighbourhood is $\mathcal{O}(n^2)$. Figure 5.7 depicts an example.

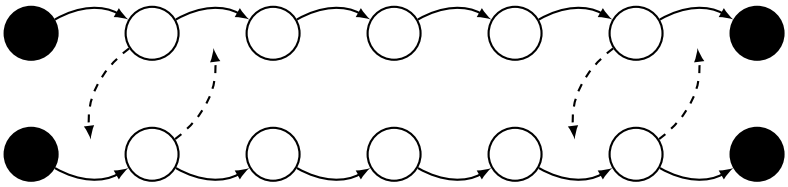


Fig. 5.6: MOVENODEROUTES: normal arcs denote the route while dashed arcs indicate possible moves.

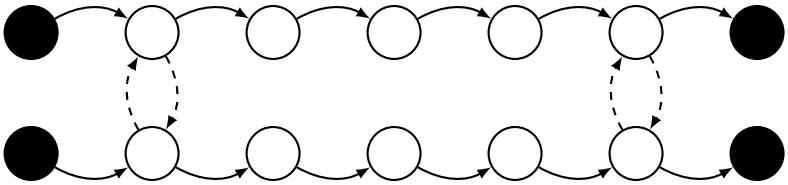


Fig. 5.7: SWAPNODESROUTES: normal arcs denote the route while dashed arcs indicate possible swaps.

ExtractNode The goal is to reduce the cost of the solution by extracting one node from a route, in such a way to minimise the profit reduction. The size of this neighbourhood is $\mathcal{O}(n)$. Figure 5.8 depicts an example.

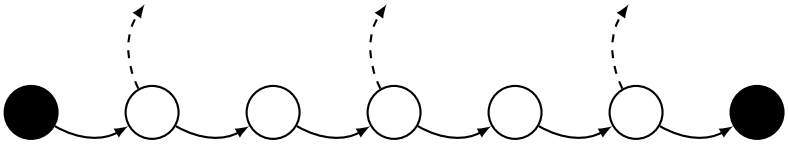


Fig. 5.8: EXTRACTNODE: normal arcs denote the route while dashed arcs indicate possible extractions.

Finally, the classical TWO-OPT neighbourhood [74] was also introduced, aiming to reduce the cost of a route by exchanging two nodes in such a way to maximise the overall cost saving. To this end, the full set of swaps are explored and then the size is $\mathcal{O}(n^2)$. Figure 5.9 depicts an example.

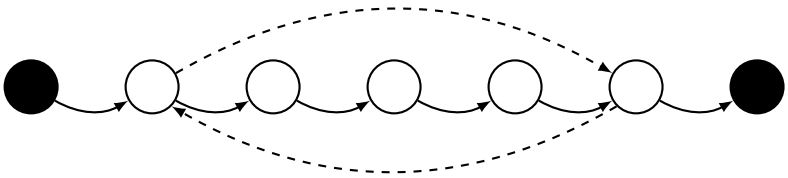


Fig. 5.9: TWOOPT: normal arcs denote the route while dashed arcs indicate possible extractions.

The exploration of the solution space

The VND algorithm explores the solution space in two different ways, by alternatively searching for a new solution in the feasible region (SFR phase) or the infeasible region (SIR phase), respectively. It makes use of a couple of variable-size Tabu-Lists

(TLs) to fix some nodes inside (list L_1) or outside (list L_2) the solution for a given amount of iterations (list L_1 of length ℓ_1 and list L_2 of length ℓ_2). Their dimension depends on the current solution, and are defined in function of the total number of customers visited by the routes. Even if the SFR and SIR phases explore the neighbourhoods in a local search fashion, the two TLs avoid restoring feasibility simply by extracting the node just inserted during the SIR phase. Furthermore, they provide a light intensification and diversification of the search by fixing nodes inside and outside the current solution through the tabu lists.

The main purpose of the SFR phase is to improve the quality of a solution trying (i) to maximise the profit, and (ii) to minimise the cost while retaining the feasibility of the explored solutions. The SFR phase is driven by the following sequence of neighbourhoods: INSERTNODE, SWAPNODES, SWAPNODESROUTES, MOVENODESROUTES, 2-OPT, and MOVENODE. Basically, the idea is to apply all the profit-based neighbourhoods in such a way to improve the profit of the current solution and, then, to apply all the cost-based neighbourhoods to decrease the cost of the obtained routes. Each move in the SFR phase is computed by sequentially exploring the above sequence of neighbourhoods until a solution that increases profit or reduces cost, is found (first-improvement strategy). Each solution S visited during the SFR phase (especially those obtained by applying a cost-based neighbourhood) is improved through a simple intensification phase by greedily adding non-visited nodes while maintaining the cost of the routes as small as possible. This intensification is similar to the INSERTNODE neighbourhood without the use of the tabu lists playing therefore a role similar to the *aspiration criteria* of the classical tabu search. Finally, the SFR phase continues from the solution S as it allows to explore a larger portion of the solution space, as in Aringhieri et al. [10].

The main purpose of the SIR phase is to improve the quality of a solution forcing the search to explore an infeasible region of the solution space. We define the time-violation of a solution as the sum of the exceeding times of each infeasible route r, that is the sum of $\max\{c_r - T_{\max}, 0\}$. Therefore, the first step of the SIR phase is to impose the insertion of a node into a route (using the INSERTNODE neighbourhood) in such a way to make the solution infeasible with respect to the cost, i.e. a solution whose time-violation is strictly greater than zero. The second step consists in exploring the infeasible region in such a way to regain the feasibility by reducing the time-violation. To this end, the search is driven by the following sequence of neighbourhoods, that is SWAPNODESROUTES, MOVENODESROUTES, 2-OPT, MOVENODE, SWAPNODES, and EXTRACTNODE. Each move of the SIR phase is computed by sequentially exploring the above sequence of neighbourhoods until the first solution that deceases the time-violation is found (first improvement exploration). It should be noted that the SIR phase cannot guarantee to always restore feasibility, specifically when applying moves from the EXTRACTNODE neighbourhood.

This could be due to the presence of mandatory nodes and physical incompatibilities in the route whose cost exceeds $T_{\max}$.

The algorithm

The VND algorithm starts from an initial solution computed by the MNAA, and iterates the SFR and SIR phases. At each iteration, the VND will apply a move selected by exploring a certain neighbourhood in accordance with the current search phase. The VND explores the solution space until a stopping condition is met, that is after reaching a specific number $\zeta_{\text{it}}^{\text{VND}}$ of non-improving iterations (or moves).

The initial solution could be infeasible since the cost of one or more routes c_r could be greater than $T_{\max}$: in this case, the VND starts its iterations from the SIR phase. It may happen that the solution remains infeasible for a number of iterations ξ during the SIR phase. In this case, the VND restarts its computation from a *recovery solution* S_{R} previously stored, and set to the first feasible solution found at the end of the previous SIR phase. Finally, if S_{R} is not available, the VND restarts from the initial solution computed by the MNAA. Note that the exploration of the solution space will be be different from the beginning of the VND due to the current status of the TLs whose lengths are updated in accordance with the total number of customers visited.

The VND pseudocode is reported in Algorithm 4, which makes use of the following procedures: (i) `nextMove`: selects the next move exploring the current (or the next in the list) neighbourhood in the current phase ($move = \emptyset$ when all the neighbourhoods in the phase are explored), (ii) `applyMove`: applies the selected move to the incumbent solution, (iii) `updateTabuLists`: updates the TLs according to the selected move, (iv) `resetTabuList`: resets the TLs, and (v) `optimise`: inserts the most profitable nodes inside the routes of S minimising their insertion costs.

5.6.3 A matheuristic algorithm based on a cuts-separation scheme

The cuts-separation method is one of a variety of optimization methods that iteratively refine a feasible region by finding and adding inequalities to a model. A cut can be added to the problem to reduce the search space and this process is repeated until an optimal feasible integer solution is found. We propose a Cuts-Separation Heuristic, called CSH.

We consider a TOP-ST-MIN formulation based on the extension of the classical formulation for TOP reported in Vansteenwegen et al. [121], which is characterised by an exponential number in the number of nodes of subtour elimination constraints (SECs). Our algorithm considers a reduced problem (RP) in which all the SECs are removed from the mathematical program. Each time a new integer solution to

Algorithm 4: The `VND` procedure.

Data: Solution S
Result: Solution S^*

$S^* \leftarrow \emptyset$; $S_R \leftarrow \emptyset$ $z^* \leftarrow 0$; $i \leftarrow 0$; $j \leftarrow 0$; phase $\leftarrow$ (S is feasible ? SFR : SIR); TLs $\leftarrow$ `resetTabuList()`;
while *not* stoppingCondition(i) **do**
  if phase is SFR **then** `/* Search Feasible Region (SFR) phase */`
    move $\leftarrow$ `nextMove` (S, neighbourhoods, TLs, phase);
    if *move* $= \emptyset$ **then** phase $\leftarrow$ SIR;
    $j \leftarrow 0$;
  else `/* Search Infeasible Region (SIR) phase */`
    if S is feasible **then** move $\leftarrow$ INSERTNODE (S, TLs);
    else move $\leftarrow$ `nextMove` (S, neighbourhoods, TLs, phase);
    $j \leftarrow j + 1$;
  if move $\neq \emptyset$ **then**
    $S \leftarrow$ `applyMove` (S, move, TLs);
    TLs $\leftarrow$ `updateTabuLists` (move, TLs);
  if S is feasible **then**
    phase $\leftarrow$ SFR;
    $S_{\text{OPT}} \leftarrow$ `optimise` (S);
    if $S_R = \emptyset$ **then** $S_R \leftarrow S$;
    if $z^* < z\,(S_{\text{OPT}})$ **then** $S^* \leftarrow S_{\text{OPT}}$; $z^* \leftarrow z\,(S_{\text{OPT}})$;
  if $j = \xi$ **then** $S \leftarrow$ assignRecoverySolution(S_R); `/* Restart the search from the recovery solution */`
  $i \leftarrow i + 1$;
return S^*;

the RP is found, all the corresponding violated SECs are added into the RP. Such a solution is (eventually) repaired and then improved applying the VND. To fully exploit the number of feasible solutions generated by the VND, an instance of the Set Packing Problem (SPP) is solved to select the most profitable feasible routes as in Hammami et al. [48]). The CSH iterates through the previously described process until a stopping condition is met.

The reduced problem

We formulate the RP on a directed graph defined on N, the set of nodes ranging from 1 (the source or depot) to n (the destination). The aim of the RP is to maximise the profit collected by m vehicles without exceeding their time budget $T_{\max}$. Additionally, all the mandatory customers have to be served and no incompatibility between nodes must be violated. The RP can be formulated extending the TOP formulation in [121] adding the TOP-ST-MIN constraints, and without considering the SECs. This formulation makes use of the following decision variables: the variable x_{ijr} is equal to 1 if and only if the arc (i, j) is traversed by the route r, 0 otherwise; the variable y_{kr} is equal to 1 if and only if the node k is visited by the route r, 0 otherwise.

$$\max \quad \sum_{r=1}^{m} \sum_{k=2}^{n-1} p_k \, y_{kr} \tag{5.11a}$$

$$\text{s.t.} \quad \sum_{r=1}^{m} \sum_{j=2}^{n} x_{1jr} = \sum_{r=1}^{m} \sum_{i=1}^{n-1} x_{inr} = m, \tag{5.11b}$$

$$\sum_{i=1}^{n-1} x_{ikr} = \sum_{j=2}^{n} x_{kjr} = y_{kr}, \quad \forall\, k \in \hat{N}, \quad r = 1, \ldots, m, \tag{5.11c}$$

$$\sum_{i=1}^{n-1} \sum_{j=2}^{n} (s_i + t_{ij})\, x_{ijr} \leq T_{\max}, \qquad r = 1, \ldots, m, \tag{5.11d}$$

$$\sum_{r=1}^{m} y_{kr} \leq 1, \quad \forall\, k \in \hat{N}, \tag{5.11e}$$

$$\sum_{r=1}^{m} y_{kr} = 1, \quad \forall\, k \in M, \tag{5.11f}$$

$$\sum_{r=1}^{m} x_{ijr} = 0, \quad \forall\, (i,j) \in I, \tag{5.11g}$$

$$|C_k|(1 - y_{kr}) \geq \sum_{i \in C_k} y_{ir}, \quad \forall\, k \in \hat{N}, \quad \forall\, r = 1, \ldots, m, \tag{5.11h}$$

$$x_{ijr} \in \{0,1\}, \quad \forall\, i = 1, \ldots, n-1, \quad j = 2, \ldots, n, \quad r = 1, \ldots, m, \tag{5.11i}$$

$$y_{kr} \in \{0,1\}, \quad \forall\, k = 2, \ldots, n-1, \quad r = 1, \ldots, m \tag{5.11j}$$

The objective function (5.11a) maximises the overall profit collected by each route. Constraint (5.11b) guarantees (along with the optimisation direction of the objective function) that each route starts from the source node and ends at the destination node. Constraints (5.11c) guarantee the connectivity of each route while constraints (5.11e) guarantee that a node can only be visited by at most one route. Constraints (5.11d) guarantee that the total visit time does not exceed $T_{\max}$ for each route. Constraints (5.11f) ensure that all the mandatory customers M will be visited. Constraints (5.11g) and (5.11h) respectively guarantee that all the physical and logical incompatibilities are satisfied. Finally, constraints (5.11i) and (5.11j) are the variable definition constraints.

Repairing and improving the reduced problem solution

Let S_{RP} be the solution obtained by solving the RP with some general purpose solver. Such a solution can be composed of a set of m routes (from node 1 to node n), and some other subtours not connected to the route because of these specific SECs are not included in the RP.

When such subtours contain some mandatory nodes, the algorithm tries to repair the m routes by inserting these mandatory nodes into the routes obtaining a solution S_M, which can be infeasible due a time-violation. The insertion is performed adopting the

same scheme depicted in Figure 5.2 for the Algorithm 3. If such an approach is not able to assign all the mandatory nodes, the CSH considers a restricted TOP-ST-MIN instance composed of the nodes in S_{RP}. Such an instance is solved by a general purpose solver exploiting the faster flow-based formulation reported in Section 5.3.2. If the general purpose solver fails to find a feasible solution, the CSH restarts its computation from the recovery solution stored during the VND computation. If this solution is not available, the CSH stop returning an infeasibility flag. At the end of the repairing routine, the obtained solution S_{M} will be the initial solution for the VND.

The set packing problem

To further improve the best solution found by the VND, an instances of the SPP is solved to generate an additional feasible solution composed by the most profitable m feasible routes generated in the previous iterations of the VND algorithm. In order to guarantee the existence of such a solution, the SPP instance includes all the routes generated during the VND whose set of visited customers are different. The SPP is formulated as follows:

$$\max \quad \sum_{r}^{\Omega} \lambda_r \, y_r \tag{5.12a}$$

$$\text{s.t.} \quad \sum_{r}^{\Omega} a_{kr} y_r \leq 1, \quad \forall \, k \in \hat{N} \setminus M, \tag{5.12b}$$

$$\sum_{r}^{\Omega} a_{kr} y_r = 1, \quad \forall \, k \in M, \tag{5.12c}$$

$$\sum_{r}^{\Omega} y_r \leq m, \quad \forall \, r \in \Omega, \tag{5.12d}$$

$$y_r \in \{0, 1\} \quad \forall \; r \in \Omega, \tag{5.12e}$$

The binary variable y_r is used to select the route r inside the solution. The set Ω represents the whole set of routes computed by several iterations of VND. The parameter a_{kr} is 1 if the customer k is visited by the route r, 0 otherwise. The objective function (5.12a) maximises the overall profit of the selected routes. Constraint (5.12b) guarantees that the non-mandatory customer k is visited by, at most, one route. Constraint (5.12c) guarantees that the mandatory customer k is visited by exactly one route. Constraints (5.12d) guarantee that a solution is composed by at most m routes.

Adding subtour elimination constraints

To ensure progress toward an optimal solution, all the Subtour Elimination Constraints (SECs) associated to the solution S_{RP} are inserted into the RP. To identify the subtours within S_{RP}, a Depth-First Search (DFS) algorithm is employed. The algorithm adopts an adaptation of the Dantzig-Fulkerson-Johnson SECs [29]:

$$\sum_{r=1}^{m} \sum_{(i,j)\in U\times U} x_{ijr} \leq \sum_{r=1}^{m} \left(\sum_{i\in U} y_{ir} - y_{kr} \right), \quad \forall\, U \subseteq \{2,\ldots,n-1\},\ k \in U \tag{5.13}$$

The validity of (5.13) follows from the constraints (5.11c) and (5.11e).

The algorithm

A pseudocode of the CSH algorithm is provided in Algorithm 5, which makes use of the following procedures: (i) `extractSubtours`: extracts all the subtours from the solution given by the RP, (ii) `repairSolution`: repairs the RP solution, (iii) `buildSECs`: builds the SECs associated with the set of subtours, and (iv) `SPP`: solves an instance of the SPP using the whole set of feasible routes generated.

Algorithm 5: The CSH-st single-thread procedure.

```
   Result: Solution S*
1  S* ← ∅; i ← 0; SECs ← ∅;
2  for i ← 0 to ζ_it^CSH do
3      S_RP ← RP (SECs);
4      if S_RP ≠ ∅ then
5          Φ_RP ← extractSubtours (S_RP);
6          if Φ_RP ∩ M ≠ ∅ then S_M ← repairSolution (S_RP);
7          S_M ← S_RP S_VND ← VND (S_M);
8          S_SPP ← SPP (S_VND);
9          if z* < z (S_SPP) then S* ← S_SPP; z* ← z (S_SPP);
10         SECs ← SECs ∪ buildSECs (Φ_RP);
11         i ← i + 1;
12     else return ∅;
13 return S*;
```

The CSH starts by initialising the set of SECs to the empty set and the iteration counter. The loop is repeated for a maximum number $\zeta_{\text{it}}^{\text{CSH}}$ of iterations. Each iteration starts by solving the RP including also the current set of SECs. If the RP has no solution, the CSH stops its computation. Otherwise, the CSH (i) extracts the subtours Φ_{RP} from the obtained solution S_{RP}, (ii) the solution S_{RP} is eventually repaired and then improved by applying the VND obtaining a new solution S_{VND}, (iii) the S_{VND} is then refined by solving the corresponding SPP problem, and (iv) the resulting solution S_{SPP} is finally stored if it improved the current optimal solution and the SECs are updated.

From a different perspective, it is worth noting that the CSH generates a sequence of $\zeta_{\text{it}}^{\text{CSH}}$ RP solutions in which eliminating subtours Φ_{RP} guides the algorithm towards convergence to an optimal solution. The process of improving the RP solutions, i.e., an iteration of Algorithm 5, can be parallelised. This leads us to introduce a multi-thread version of the CSH algorithm, which is depicted in Algorithm 6: (i) the single-thread part consists in the computation of the $\zeta_{\text{it}}^{\text{CSH}}$ RP solutions S_{M}^{i}, while (ii) the multi-thread part of the algorithm uses a thread pool (whose dimension is set to the number of available threads) to solve a S_{M}^{i} as soon as a thread in the pool is available.

Algorithm 6: The CSH-mt multi-thread procedure.

Result: Solution S^*
$S^* \leftarrow \emptyset$; SECs $\leftarrow \emptyset$;
for $i \leftarrow 1$ **to** ζ_{it}^{CSH} **do** `/* Single-thread execution */`
  $S_{\text{RP}} \leftarrow$ `RP` (SECs);
  if $S_{\text{RP}} \neq \emptyset$ **then**
    $\Phi_{\text{RP}} \leftarrow$ `extractSubtours` (S_{RP});
    if $\Phi_{RP} \cap M \neq \emptyset$ **then** $S_{\text{M}}^{i} \leftarrow$ `repairSolution` (S_{RP});
    $S_{\text{M}}^{i} \leftarrow S_{\text{RP}}$ SECs $\leftarrow$ SECs $\cup$ `buildSECs` (Φ_{RP});
  else return $\emptyset$;
  $i \leftarrow i + 1$;
$S_{\text{VND}} \leftarrow$ `VND` (S_{M}^{i}); $S_{\text{SPP}} \leftarrow$ `SPP` (S_{VND}); `/* Multi-thread execution */`
if $z^* < z\,(S_{\text{SPP}})$ **then** $S^* \leftarrow S_{\text{SPP}}$; $z^* \leftarrow z\,(S_{\text{SPP}})$;
return S^*;

Implementation details Two components of the CSH algorithm—the solution of the RP and the SPP, can be computationally expensive. This section outlines the implementation strategies employed to reduce their computational time.

To limit the runtime for solving the RP, the general-purpose solver is restricted to solve only the root node, emphasising the search for feasible solutions through built-in heuristics. Additionally, a time limit of 360 seconds was imposed, along with a limit of 10 feasible solutions found by the solver heuristics.

For the SPP, the VND algorithm has an extra stopping condition that limits the number of generated routes at most 200.000. To consider only different routes, a hash-table data structure is adopted with custom hash function in order to identify routes with the same set of visited customers. Finally, the value of the best VND solution is provided to the solver as a lower bound.

5.7 Quantitative analysis

This section presents the key results of a comprehensive computational analysis conducted to evaluate the performance of the VND and CSH algorithms. Section 5.7.1 outlines the computational environment used for the experiments. Section 5.7.2

then presents the results obtained for both variants of the TOP-ST-MIN and compares them with those achieved by the exact Cutting-Plane Algorithm (CPA) described in Section 5.4.2. Finally, Section 5.7.3 provides additional results on the classical TOP instances, obtained by a simple adaptation of the algorithms in order to further validate both approaches.

5.7.1 Setting up the computational environment

The sets of benchmark instances are the ones described in Chapter 4. Such instances have been generated in such a way to assess the real impact of the three new features of the TOP-ST-MIN.

The instances were generated starting from the 7 sets available at the KU Leuven website `https://www.mech.kuleuven.be/en/cib/op` and introduced by Chao et al. [24]. Such instances have been grouped by size (*small*, from 21 to 33 nodes, *medium*, from 64 to 66 nodes and *large* from 100 to 102 nodes) and modified in order to include service time, mandatory nodes and physical/logical incompatibilities in such a way that:

- Mandatory nodes: 5% of the nodes are labelled as mandatory; depending on the instance, those nodes are chosen as *clustered* (close to each other) or *scattered* (far from each other).
- Physical incompatibilities: 20% of the edges are eliminated from the graph, in order to form either *clusters* in the graph (*clusters-based*) or to obtain a *uniform* degree between the nodes (*degree-based*).
- Logical incompatibilities: each node is declared logically incompatible with 5% of the graph nodes, which are either *far* or *near* in terms of travel time.

These different methods are used for generating the instances are applied in sequence, that is one method for the mandatory nodes, then one for the physical incompatibilities, and one for the logical incompatibilities only for the TOP-ST-MIN-PL instances. The acronyms used for presenting and analysing the results are summarized in Table 5.10, which is divided into two parts: the left side describes the generation methods, while the right side reports the number of instances generated with each method across different instance sizes for both problems. Each feature partitions the whole set into two equally sized subsets (e.g., the SMN and CMN for mandatory nodes divide the set of 156 small instances of TOP-ST-MIN-P into two subsets of 78 instances, each containing one half of CPI and one half of DPI instances). We refer to Chapter 4 for additional details on the feature generation methods and on the process of instance generation.

The VND and the CSH have been implemented in C++ adopting CPLEX 22.1.1 Concert Technology, and compiled in Scientific Linux 7.9 (Nitrogen The experiments

Tab. 5.10: Summary of feature generation methods for the benchmark instances.

Feature	Generation methods		TOP-ST-MIN-P			TOP-ST-MIN-PL		
	Acronym	Description	Small	Medium	Large	Small	Medium	Large
Mandatory Nodes	SMN | CMN	Clustered | Scattered	78	60	60	156	120	120
Physical Incompatibilities	CPI | DPI	Clusters-based | Degree-based	78	60	60	156	120	120
Logical Incompatibilities	FLI | NLI	Farthest | Nearest	–	–	–	156	120	120

were carried out on a 64-bit Linux machine, with a Six-Core (12 threads) AMD Opteron(tm) Processor 8425 HE, 2.10 GHz and 24 GB of RAM. We use CPLEX as a general purpose solver with all built-in cuts in a fully-deterministic mode.

After several preliminary tests, the VND parameters are set as follows: (i) the value of $\zeta_{\text{it}}^{\text{VND}}$ is set to 10.000 for the small, 100.000 for the medium, and 500.000 for the large benchmark set of instances, (ii) the values of ℓ_1 and ℓ_2 is set to the 40% of the number of nodes in the current solution (so the length dynamically change), (iii) the value of ξ is set to 500 iterations. Similarly, the CSH parameters are set as follows: (i) the value of $\zeta_{\text{it}}^{\text{CSH}}$ is set equal to 10 for the small, 30 for the medium, and 50 for the large benchmark set of instances, (ii) the value of $\zeta_{\text{it}}^{\text{VND}}$ (for the VNS as inner procedure of CSH) is set equal to 5.000 for the small, 10.000 for the medium, and 30.000 for the large benchmark set of instances, (iii) the CPLEX time limit for the RP and SPP solution is set to $1,200$ seconds.

Finally, the computational results show that CSH-mt provides a speed up, on average, of 7.6 times the CSH-st. Considering the small, medium and large instances, and both TOP-ST-MIN-P and TOP-ST-MIN-PL, such a speed up ranges between 7.49 and 7.75, which represents an honourable value considering that the CSH-mt has an initial not negligible single-thread computation and the maximum possible speed up is 12 (equal to the maximum number of threads). To avoid redundancy and to emphasise the strength of the algorithm, only the results for CSH-mt are presented in the remainder of the section, and are referred to simply as CSH. In fact, the algorithmic structure of the CSH allows for a straightforward parallelization.

5.7.2 Computational results on TOP-ST-MIN instances

Table 5.11 reports the comparison between the VND and the CPA on 396 benchmark instances for the TOP-ST-MIN-P listed by feature generation methods as depicted in Table 5.10. The results are summarised in the ALL rows, also by instance size.

Focusing on the left sub-table, (i) the column # is the number of instances, (ii) the column SOL is the number of solutions found, (iii) CPU(s) is the average amount of running time (in seconds), (iv) LIT(s) is the average last improvement time (in seconds), and (v) LII is the average last improving iteration. On the right sub-tables, the columns Δ_{OBJ} and Δ_{CPU} reports the average relative gaps (in percentage) for the values of the objective function and running time, respectively, with respect to the CPA. Such gaps consider only those instances for which both methods find at

least a feasible solution. Finally, the column Δ_{SOL} reports the absolute gap in terms of number of solutions found.

It is worth noting that Δ_{OBJ} and Δ_{SOL} are positive when the VND is better than CPA while the value Δ_{CPU} is negative when the VND is faster than the CPA.

Tab. 5.11: The VND results (left) and the comparison with the CPA for the TOP-ST-MIN-P (right).

		#	CPU(s)	LIT(s)	LII	SOL
SMALL	**ALL**	**156**	**3.20**	**0.01**	**66**	**156**
	CMN	78	2.33	0.02	92	78
	SMN	78	4.07	0.01	39	78
	CPI	78	2.59	0.01	57	78
	DPI	78	3.81	0.02	75	78
MEDIUM	**ALL**	**120**	**100.74**	**1.30**	**1299**	**118**
	CMN	60	102.00	1.51	1632	60
	SMN	60	99.47	1.08	966	58
	CPI	60	111.21	1.13	954	59
	DPI	60	90.26	1.46	1644	59
LARGE	**ALL**	**120**	**1128.27**	**192.12**	**97075**	**94**
	CMN	60	1390.93	288.62	154369	60
	SMN	60	865.62	95.63	39780	34
	CPI	60	1159.04	177.67	75371	47
	DPI	60	1097.51	206.57	118778	47
ALL		**396**	**373.69**	**58.62**	**29836**	**368**

		Δ_{OBJ}	Δ_{CPU}	Δ_{SOL}
SMALL	**ALL**	**-5.27**	**-81.71**	**0**
	CMN	-5.82	-92.13	0
	SMN	-4.64	-24.75	0
	CPI	-5.36	-83.50	0
	DPI	-5.17	-80.25	0
MEDIUM	**ALL**	**-3.32**	**-92.31**	**1**
	CMN	-2.59	-93.20	0
	SMN	-4.10	-91.12	1
	CPI	-2.74	-90.25	0
	DPI	-3.92	-93.89	1
LARGE	**ALL**	**-1.79**	**-81.30**	**47**
	CMN	-1.42	-80.43	19
	SMN	-4.25	-82.56	28
	CPI	-3.10	-81.07	25
	DPI	-0.61	-81.55	22
ALL		**-3.45**	**-83.26**	**48**

The results reported in Table 5.11 show that the VND is always much faster than the CPA computing good quality solutions in terms of objective function. A clear advantage of the VND is to find almost always at least one feasible solution, even for those instances that the CPA is not able to certify as infeasible. As a matter of fact, the VND is able to find 48 additional solutions for the TOP-ST-MIN-P, 47 of which are solutions for the larger instances.

Tables 5.11 reveals some common behaviour regarding the feature generation methods. It is worth noting that the VND finds more feasible solutions for the SMN instances compared to the other ones. Specifically, they identified 28 additional feasible solutions for the SMN instances, compared to 19 for CMN instances. On the contrary, the comparison between CPI and DPI instances revealed no notable differences in the number of feasible solutions found.

Table 5.12 reports the comparison between the CSH and the multithread version of the CPA for the TOP-ST-MIN-P benchmark sets. The results are listed by generation methods as depicted in Table 5.10. Table 5.12 has the same structure of Table 5.11 except for the columns LIT(s) and LII, which are specific for the VND. We recall that Δ_{OBJ} and Δ_{SOL} are positive when the CSH is better than CPA while the value Δ_{CPU} is negative when the CSH is faster than the CPA.

Tab. 5.12: The CSH results (left) and the comparison with the multithread version of the CPA for the TOP-ST-MIN-P (right).

		#	CPU(s)	SOL			Δ_{OBJ}	Δ_{CPU}	Δ_{SOL}
SMALL	**ALL**	**156**	**86.62**	**156**	**SMALL**	**ALL**	**-1.06**	**2971.43**	**0**
	CMN	78	89.37	78		CMN	-0.97	2162.77	0
	SMN	78	75.87	78		SMN	-1.16	5204.41	0
	CPI	78	89.07	78		CPI	-1.27	3196.13	0
	DPI	78	76.18	78		DPI	-0.85	2744.68	0
MEDIUM	**ALL**	**120**	**486.02**	**118**	**MEDIUM**	**ALL**	**-0.07**	**20.11**	**0**
	CMN	60	525.71	60		CMN	0.00	96.39	0
	SMN	60	446.33	58		SMN	-0.13	-17.59	0
	CPI	60	475.88	59		CPI	-0.02	13.25	0
	DPI	60	496.16	59		DPI	-0.11	27.52	0
LARGE	**ALL**	**120**	**2340.99**	**94**	**LARGE**	**ALL**	**-0.04**	**-37.19**	**0**
	CMN	60	2556.64	60		CMN	-0.07	-44.26	0
	SMN	60	2125.34	34		SMN	0.01	-25.89	0
	CPI	60	2433.48	47		CPI	0.01	-36.61	0
	DPI	60	2248.50	47		DPI	-0.10	-37.81	0
ALL		**396**	**889.22**	**368**	**ALL**		**-0.25**	**-29.04**	**0**

The results reported in Table 5.12 shows that the CSH is capable to compute high quality solutions for all the small, medium, and large instances, on average. In fact, although the Δ_{OBJ} column indicates that the CPA is slightly better than the CSH, most of the computed solution by the CPA are optimal. Regarding the running time, the CSH performs worse on the small and medium instances and better on the large instances. Both approaches find the same number of feasible solutions for the TOP-ST-MIN-P.

Tables 5.13 and 5.14 report the corresponding comparisons on the 792 TOP-ST-MIN-PL instances, and they has the same structure of Table 5.11 and 5.12, respectively. We recall that Δ_{OBJ} and Δ_{SOL} are positive when the VND or CSH is better than CPA while the value Δ_{CPU} is negative when the VND or CSH is faster than the CPA.

The results reported in Table 5.13 show that the VND is always much faster than the CPA, on average. Contrary to what is shown on TOP-ST-MIN-P instances, the VND computes better quality solutions for the larger instances. Furthermore, the VND is capable to find 99 additional solutions with respect to the CPA, 90 of which are solutions for the larger instances.

The results reported in Table 5.14 shows that the CSH is able to further improve the gaps on larger instances. Conversely, the algorithm performs worse than CPA on the small and medium instances. Regarding the running time, the CSH is always faster than the CPA on all instances. Finally, the CSH is capable to find 69 additional solutions with respect to the CPA, 62 of which are solutions for the large instances.

Tab. 5.13: The VND results (left) and the comparison with the CPA for the TOP-ST-MIN-PL (right).

		#	CPU(s)	LIT(s)	LII	SOL			Δ_{OBJ}	Δ_{CPU}	Δ_{SOL}
SMALL	**ALL**	**312**	**4.05**	**0.02**	**65**	**312**	**SMALL**	**ALL**	**-4.00**	**-95.81**	**0**
	CMN	156	3.23	0.02	91	156		CMN	-4.44	-98.08	0
	SMN	156	4.87	0.01	40	156		SMN	-3.50	-80.80	0
	CPI	156	3.49	0.02	61	156		CPI	-4.10	-96.81	0
	DPI	156	4.60	0.02	69	156		DPI	-3.90	-94.51	0
	FLI	156	4.99	0.01	53	156		FLI	-5.14	-90.94	0
	NLI	156	3.11	0.02	78	156		NLI	-2.69	-97.75	0
MEDIUM	**ALL**	**240**	**163.74**	**3.74**	**2632**	**236**	**MEDIUM**	**ALL**	**-2.07**	**-95.95**	**9**
	CMN	120	173.20	5.70	4131	120		CMN	-1.49	-96.29	0
	SMN	120	154.28	1.79	1133	116		SMN	-2.77	-95.48	9
	CPI	120	169.08	2.42	1417	118		CPI	-2.05	-96.11	5
	DPI	120	158.40	5.07	3847	118		DPI	-2.10	-95.76	4
	FLI	120	188.08	2.40	1177	118		FLI	-2.12	-94.04	5
	NLI	120	139.41	5.09	4087	118		NLI	-2.01	-97.17	4
LARGE	**ALL**	**240**	**1638.88**	**282.98**	**59971**	**142**	**LARGE**	**ALL**	**4.05**	**-72.20**	**90**
	CMN	120	1962.45	420.25	87648	80		CMN	4.09	-72.66	30
	SMN	120	1315.31	145.72	32293	62		SMN	2.41	-71.49	60
	CPI	120	1718.79	310.02	59053	71		CPI	3.04	-70.88	45
	DPI	120	1558.97	255.95	60888	71		DPI	5.06	-73.54	45
	FLI	120	2575.79	502.96	91739	94		FLI	3.94	-57.56	49
	NLI	120	701.97	63.00	28202	48		NLI	5.17	-87.73	41
ALL		**792**	**547.84**	**86.89**	**18996**	**690**	**ALL**		**-1.76**	**-82.04**	**99**

Tab. 5.14: The CSH results (left) and the comparison with the multithread version of the CPA for the TOP-ST-MIN-PL (right).

		#	CPU(s)	SOL			Δ_{OBJ}	Δ_{CPU}	Δ_{SOL}
SMALL	**ALL**	**312**	**82.03**	**312**	**SMALL**	**ALL**	**-0.65**	**-14.72**	**0**
	CMN	156	101.46	156		CMN	-0.61	-18.74	0
	SMN	156	62.60	156		SMN	-0.70	-7.28	0
	CPI	156	82.84	156		CPI	-0.71	-18.52	0
	DPI	156	81.22	156		DPI	-0.59	-10.46	0
	FLI	156	80.26	156		FLI	-1.00	1320.81	0
	NLI	156	83.80	156		NLI	-0.25	-55.12	0
MEDIUM	**ALL**	**240**	**584.53**	**236**	**MEDIUM**	**ALL**	**0.71**	**-81.97**	**7**
	CMN	120	623.79	120		CMN	1.00	-83.59	2
	SMN	120	545.26	116		SMN	0.37	-79.68	5
	CPI	120	594.72	118		CPI	0.69	-83.01	3
	DPI	120	574.33	118		DPI	0.72	-80.75	4
	FLI	120	476.16	118		FLI	0.09	-71.25	0
	NLI	120	692.90	118		NLI	1.56	-85.65	7
LARGE	**ALL**	**240**	**2369.71**	**142**	**LARGE**	**ALL**	**3.29**	**-57.05**	**62**
	CMN	120	2483.53	80		CMN	1.83	-62.58	16
	SMN	120	2255.89	62		SMN	10.76	-48.70	46
	CPI	120	2447.66	71		CPI	3.96	-56.07	29
	DPI	120	2291.77	71		DPI	2.54	-58.05	33
	FLI	120	3500.85	94		FLI	2.65	-34.21	21
	NLI	120	1238.58	48		NLI	15.02	-78.32	41
ALL		**792**	**927.54**	**690**	**ALL**		**0.88**	**-65.55**	**69**

Tables 5.13 and 5.14 highlight an even more pronounced trend with respect to that for the TOP-ST-MIN-P in terms of number of feasible solutions. Indeed, the proposed algorithms found 60 additional feasible solutions for the SMN instances compared to 30 for the CMN instances. In this case again, no relevant difference emerge in the number of feasible solutions found by the VND and CSH for the CPI and DPI instances. The same logic also follows for the FLI and NLI instances. Therefore, it can be argued that the SMN instances, that is those instances with mandatory nodes far from each other, tend to be easier to solve compared to other types, which supports the observations made in Section 5.5.

Tab. 5.15: VND and CSH comparison for TOP-ST-MIN-P (left) and for the TOP-ST-MIN-PL (right).

		Δ_{OBJ}	Δ_{CPU}	Δ_{SOL}
SMALL	**ALL**	**4.44**	**2482.05**	**0**
	CMN	5.15	3738.70	0
	SMN	3.65	1763.47	0
	CPI	4.32	3336.06	0
	DPI	4.56	1900.64	0
	-	-	-	-
	-	-	-	-
MEDIUM	**ALL**	**3.37**	**382.48**	**0**
	CMN	2.68	415.41	0
	SMN	4.13	348.70	0
	CPI	2.81	327.90	0
	DPI	3.97	449.72	0
	-	-	-	-
	-	-	-	-
LARGE	**ALL**	**2.91**	**107.48**	**0**
	CMN	2.54	83.81	0
	SMN	5.54	145.53	0
	CPI	3.81	109.96	0
	DPI	2.13	104.87	0
	-	-	-	-
	-	-	-	-
ALL		**3.52**	**137.96**	**0**

		Δ_{OBJ}	Δ_{CPU}	Δ_{SOL}
SMALL	**ALL**	**3.47**	**1926.75**	**0**
	CMN	3.98	3043.59	0
	SMN	2.90	1186.11	0
	CPI	3.54	2271.07	0
	DPI	3.41	1665.27	0
	FLI	4.36	1509.01	0
	NLI	2.47	2597.47	0
MEDIUM	**ALL**	**3.08**	**256.98**	**0**
	CMN	2.59	260.15	0
	SMN	3.66	253.43	0
	CPI	3.16	251.74	0
	DPI	2.99	262.58	0
	FLI	2.38	153.17	0
	NLI	3.96	397.03	0
LARGE	**ALL**	**2.15**	**44.59**	**0**
	CMN	2.11	26.55	0
	SMN	3.53	71.51	0
	CPI	2.61	42.41	0
	DPI	1.68	47.01	0
	FLI	2.13	35.91	0
	NLI	2.31	76.44	0
ALL		**3.05**	**69.31**	**0**

Table 5.15 reports a direct performance comparison between the VND and CSH on the TOP-ST-MIN-P and TOP-ST-MIN-PL instances respectively on the left and on the right. The results are summarised in the ALL rows, also by instance size and generation methods. The columns Δ_{OBJ} and Δ_{CPU} measure the average relative gaps (in percentage) of the CSH with respect to the VND for the objective function and for the running time, respectively. The column Δ_{SOL} measures the absolute gap between the two algorithms in terms of number of solutions found. We would remark that a positive value in the columns Δ_{OBJ} and Δ_{SOL} means that the CSH is better than the VND. On the contrary, a negative value in the column Δ_{CPU} means that the CSH is better than the VND.

The results reported in Table 5.15 show that the CSH is effective than the VND. On the contrary, the VND results much more efficient than the CSH. It is possible to observe that the relative gaps (Δ_{OBJ} and Δ_{CPU}) are slightly better for the CSH on the TOP-ST-MIN-P instances. This means that the CSH results to be more effective but also less efficient for the TOP-ST-MIN-P. Furthermore, the Δ_{SOL} is always zero since both algorithms are capable to compute the same number of feasible solutions. This provides strong evidence about the solvability of those instances for which the CPA is unable either to find a feasible solution or to establish its infeasibility.

5.7.3 Computational results on TOP instances

In this section, a computational comparison among the best algorithms for the TOP and both VND and CSH algorithms, is given. The idea is to provide a further validation of the proposed algorithms. It is important to note that using the TOP as a baseline problem implies that the three key features characterising the TOP-ST-MIN, on which the algorithms are based, are not taken into account.

To the best of the current knowledge, the three best heuristic algorithms for the TOP are (i) the PSOiA [28], (ii) the PMA [62], (iii) the HALNS [48]. In the computational experiments, both the classical benchmark instances introduced by Chao et al. [24] and those presented by Dang et al. [28] are used. This benchmark set is composed by instances whose number of nodes ranges from 101 to 400, generated in a similar way to those ones belonging to the set of Chao et al. For the three competitors, the best value achieved across all runs is considered. All the comparisons are made by pairs of VND or CSH algorithms and one competitor. In order to provide a fair comparison, the running time of the competitors have been rescaled with respect to the CPU frequency of the machine (2.1 GHz) by the factor:

$$\frac{\alpha}{2.1},$$

where α is the cpu frequency of the corresponding competitor. Furthermore, all the three competitors have been tested multiple times using different seeds to prove their robustness since they have some random ingredients. By consequence, their average running times were rescaled in accordance with the number of runs for each instance (10 runs for the PSOiA and the PMA, and 20 for HALNS). Finally, it is important to note that the comparison between the CSH and its competitors may not be entirely fair. However, the primary goal of this analysis is qualitative (despite the fact that the algorithms running times are compared), since both the VND and CSH were not specifically designed for the TOP. In addition, the CSH has the intrinsic quality of being parallel by design, unlike its competitors, who would require some effort to achieve some form of parallelism.

Table 5.16 reports the results of the comparison between both algorithms and the three competitors on the sets 4, 5, 6 and 7 of Chao et al.'s instances. We have not considered the sets 1, 2 and 3 since they are considered the easiest ones. The columns Δ_{MAX} and Δ_{AVG} show the average relative gaps (in percentage) on the objective function with respect to the best and the average solution values of the competitors (Δ_{AVG} is available only for PSOiA). The column Δ_{CPU} reports the running time average relative differences (in percentage). A positive value of Δ_{MAX} and Δ_{AVG}, and a negative value of Δ_{CPU} indicates that the algorithm is better than the competitor.

Tab. 5.16: VND and CSH comparison with the three competitors on Chao et al.'s instances.

		PSOiA			PMA			HALNS		
	Set	Δ_{MAX}	Δ_{AVG}	Δ_{CPU}	Δ_{MAX}	Δ_{AVG}	Δ_{CPU}	Δ_{MAX}	Δ_{AVG}	Δ_{CPU}
VND	**4**	-0.71	-0.68	-57.72	-0.71	-	-31.30	-0.71	-	-1.93
	5	-0.20	-0.05	-53.54	-0.20	-	-18.40	-0.20	-	-32.35
	6	0.00	0.00	-56.64	0.00	-	-54.43	0.00	-	-27.78
	7	-0.62	-0.82	-40.81	-0.62	-	-14.14	-0.62	-	-36.10
CSH	**4**	-0.02	0.08	-35.90	-0.02	-	4.15	-0.02	-	48.68
	5	0.00	0.01	-46.50	0.00	-	-6.04	0.00	-	-22.10
	6	0.00	0.00	-3.10	0.00	-	1.83	0.00	-	61.39
	7	-0.05	-0.04	-4.37	-0.05	-	38.70	-0.05	-	3.23

As reported in Table 5.16, the CSH results more effective in terms of solution quality but less efficient in terms of average running time compared to the VND, confirming the behaviour observed for the TOP-ST-MIN. The comparison with the three competitors shows that that the VND and the CSH are competitive with state-of-the-art heuristic methods for the TOP in terms of both solution quality and running time. Even if the proposed algorithms do not improve on any best known solution from the literature (because many of them are probably optimal objective function values), the results are, on average, very close to the ones reported by the competitors, especially for the CSH. Finally, regarding the average running times, the VND seems to be the fastest solution method across all sets and algorithms, while the CSH seems to be more efficient than the PSOiA but slightly slower than the PMA and HALNS on the average.

Table 5.17 reports the results of the comparison between the proposed algorithms and the three competitors on the Dang et al.'s instances clustering them by size (number of nodes). The table has the same structure as Table 5.16.

As shown in Table 5.17, the VND is generally less effective and efficient than the other approaches, especially in the last subset (it is faster only compared to the PSOiA in the first and last sets, on average). In contrast, the CSH remains competitive with the state-of-the-art algorithms, although it is, on average, slower than the other approaches (it is slightly faster than PSOiA only in the last set). Actually, the worst

Tab. 5.17: VND and CSH comparison with the three competitors on Dang et al.'s instances.

	Size	PSOiA			PMA			HALNS		
		Δ_{MAX}	Δ_{AVG}	Δ_{CPU}	Δ_{MAX}	Δ_{AVG}	Δ_{CPU}	Δ_{MAX}	Δ_{AVG}	Δ_{CPU}
VND	101 to 200	-3.22	-2.91	-34.43	-3.23	-2.99	27.87	-3.23	-3.10	19.24
	201 to 300	-3.25	-2.90	19.40	-3.26	-2.89	213.50	-3.26	-3.05	53.24
	301 to 400	-6.94	-6.02	-82.49	-7.11	-6.26	45.47	-7.11	-6.34	-8.76
CSH	101 to 200	-0.41	-0.08	49.56	-0.42	-0.17	167.37	-0.42	-0.28	138.77
	201 to 300	-1.00	-0.64	135.83	-1.02	-0.63	343.35	-1.02	-0.80	145.21
	301 to 400	-2.52	-1.54	-74.54	-2.70	-1.80	86.75	-2.70	-1.88	28.44

decrease in the objective function among the different subsets is around 2.70% taking into account the Δ_{MAX} or 1.88% taking into account the Δ_{AVG}.

5.8 Conclusions

This chapter studies the deterministic version of the TOP-ST-MIN. To tackle the deterministic version of the problem, two mathematical formulations were introduced in order to provide different modelling points of view. Additionally, an exact cutting-plane algorithm was developed to solve small and medium-sized instances. Two heuristic approaches were designed to efficiently handle larger instances. An extensive computational analysis validated the effectiveness of the proposed methods, emphasising their strengths across different benchmark sets evaluating the impact of the three key problem features. Notably, both approaches were able to find 48 additional feasible solutions for the TOP-ST-MIN-P and 99 for the TOP-ST-MIN-PL beyond those found by the exact method. While the CSH approach demonstrated higher effectiveness than the VND, it was comparatively less efficient. The computational results confirmed that both methods, particularly the CSH, are competitive with state-of-the-art TOP algorithms. Finally, the VND developed in this chapter will be employed in Chapter 7 as an inner component to address a dynamic and stochastic variant of the TOP-ST-MIN, and in Chapter 8 in order to investigate the problem under several fairness perspectives.

6 The Stochastic Team Orienteering Problem

The Stochastic Team Orienteering Problem (STOP) is an extension of the TOP that incorporates uncertainty. In the standard TOP, a team of vehicles seeks to maximise rewards by selecting a subset of locations to visit within given resource constraints such as time or fuel. The STOP presented in this chapter modifies this framework by introducing stochasticity in travel times modelled by Normal random variables. After an initial introduction to the problem, this chapter delves into the details of the proposed models and approaches with an in-depth examination of their components. A first mathematical formulation for the STOP is presented along with an exact method based on the Integer L-shaped [69] paradigm. Moreover, a set of chance constraints will be included into the formulation to ensure a minimum level of route reliability. A detailed procedure outlining this integration will be discussed. Furthermore, an alternative two-index formulation for the STOP and an additional exact method based on the same methodology will be described. Finally, computational experiments are presented to validate the proposed method. These results illustrate the method's performance across the TOP reference dataset.

This chapter was made in collaboration with Jean-François Côté from Université Laval.

6.1 Problem statement

The STOP is formulated as a two-stage with recourse stochastic integer programming problem. In the first stage problem, the decisions are made regarding which nodes to visit and how to assign them to different routes. The second stage problem (recourse problem), consists in a function that evaluated the expected profit of the selected first-stage solution. Therefore, no second-stage variables are involved. Unlike traditional stochastic methods that employ scenario generation procedures to sample a finite number of scenarios, the STOP assumes an infinite scenario space, since it formulates the uncertainty parameters through continuous probability distributions. This makes the problem particularly challenging as it does not rely on scenarios generation methods to approximate the value of an optimal solution. The goal of the STOP is to find a set of m routes that maximise the expected total profit collected respecting an additional set of chance constraints in order to take into account a reliability measure for each route.

6.2 A three-index mathematical formulation

In this section, a three-index mathematical formulation for the STOP is presented. After introducing additional notation that extends the one previously introduced in Section 5.1, the Section 6.2.1 further presents the *probabilistic triangle inequality* as the classical triangle inequality expressed in term of probabilities. The first-stage problem will be discussed in Section 6.2.2 and the second-stage problem will be described in Section 6.2.3. Finally, Section 6.2.4 will present how the chance constraints are linearised and embedded into the formulation.

6.2.1 Notation

This section extends the notation introduced for the TOP-ST-MIN (Sect. 5.1). The STOP is modelled on a directed graph $G = (N, A)$ where N is the set of nodes denoted from 1 (the source or depot) to n (the destination), and A is the set of directed arcs across the nodes in N. Each arc (i, j) is associated with a Normal independent random variable $\xi_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma^2_{ij})$ to represent its uncertain cost.

The sets $N(r)$ and $A(r)$ represent the set of nodes and arcs that compose the route r, respectively. The sets $\hat{N}(r)$ and $\hat{A}(r)$ are the same sets already defined without considering the source and destination nodes for the first one and the arcs connected to them for the second one.

In the following, some useful definitions will be stated. The *route uncertain cost* ξ_r defines the sum of random variables associated with the arcs composing the route r that is, $\xi_r = \sum_{(i,j)\in A(r)} \xi_{ij}$. Accordingly to the closure property of the Normal distribution with respect to the sum [13], the random variable ξ_r is still distributed as a Normal random variable. Its expectation, the *route expectation*, is defined as $\mu_r = \sum_{(i,j)\in A(r)} \mu_{ij}$ while, its variance, the *route variance*, is defined as $\sigma^2_r = \sum_{(i,j)\in A(r)} \sigma^2_{ij}$. Finally, the *route reliability* δ_r describes the degree of trustworthiness of the route r to be executed. The latter is defined as the probability that ξ_r is less or equals to $T_{\max}$ that is, $\delta_r = \mathbb{P}(\xi_r \leq T_{\max})$.

It is well known that in the Euclidean geometry and in some other geometries, the triangle inequality is a theorem about distances and it is written using vectors norms (taking two vectors u and v):

$$||v + u|| \leq ||v|| + ||u||.$$

Without loss of generality, a triple of vectors (i, j, k) can be considered. For each pair of vectors (p, q) of this triple, a Normal probability distribution $\xi_{pq} \sim \mathcal{N}(\mu_{pq}, \sigma^2_{pq})$ is defined. Now, taking into account the paths $i \to j \to k$ and $i \to k$, if the classical triangle inequality holds for expectations and variances, it follows that:

$$\mu_{ik} \leq \mu_{ij} + \mu_{jk} \quad \text{and} \quad \sigma_{ik}^2 \leq \sigma_{ij}^2 + \sigma_{jk}^2.$$

Due to the closure property of the Normal distributions with respect to the sum [13], it holds that:

$$\mathbb{P}(\xi_{ij} + \xi_{jk} \leq T_{\max}) \leq \mathbb{P}(\xi_{ik} \leq T_{\max}). \tag{6.1}$$

The equation (6.1) represents the triangle inequality expressed in terms of probabilities.

6.2.2 The first-stage problem

The first-stage problem of the STOP can be modelled as an adaptation of the mixed formulation introduced in Chapter 5.3.1 for the TOP-ST-MIN. The model is the following:

$$\max \quad \mathcal{Q}(x) \tag{6.2a}$$

$$\text{s.t.} \quad \sum_{(1,j)\in\hat{A}} x_{1jr} = \sum_{(i,n)\in\hat{A}} x_{inr} = m, \quad \forall\, r \in R, \tag{6.2b}$$

$$\sum_{(i,k)\in\hat{A}} x_{ikr} = \sum_{(k,j)\in\hat{A}} x_{kjr} = y_{kr}, \quad \forall\, k \in \hat{N},\ \forall\, r \in R, \tag{6.2c}$$

$$\sum_{(k,j)\in\hat{A}} z_{kjr} - \sum_{(i,k)\in\hat{A}} z_{ikr} = \sum_{(k,j)\in\hat{A}} \mu_{kj} x_{kjr}, \quad \forall\, k \in \hat{N},\ r \in R, \tag{6.2d}$$

$$\sum_{r\in R} y_{kr} \leq 1, \quad \forall\, k \in \hat{N}, \tag{6.2e}$$

$$z_{ijr} \leq (T_{\max} - \mu_{jn})\, x_{ijr}, \quad \forall\, (i,j) \in \hat{A},\ r \in R, \tag{6.2f}$$

$$z_{ijr} \geq (\mu_{1i} + \mu_{ij})\, x_{ijr}, \quad \forall\, (i,j) \in \hat{A},\ r \in R, \tag{6.2g}$$

$$z_{1kr} = \mu_{1k} x_{1kr}, \quad \forall\, k \in \hat{N},\ r \in R, \tag{6.2h}$$

$$y_{kr} \in \{0,1\}, \quad \forall\, k \in \hat{N}, \quad r \in R, \tag{6.2i}$$

$$x_{ijr} \in \{0,1\}, \quad \forall\, (i,j) \in \hat{A} \setminus \{(1,n)\}, \quad r \in R, \tag{6.2j}$$

$$x_{1nr} \in \mathbb{N}^+, \quad \forall\, r \in R, \tag{6.2k}$$

$$z_{ijr} \in \mathbb{R}^+, \quad \forall\, (i,j) \in \hat{A}, \quad r \in R, \tag{6.2l}$$

The proposed model makes use of the following decision variables: the variable x_{ijr} is equal to 1 if and only if the arc (i,j) is traversed by the route r, 0 otherwise; the variable y_{kr} is equal to 1 if and only if the node k is visited by the route r, 0 otherwise; the variable z_{ijr} defines the expected arrival time at node j coming from the node i and can be thought as the amount of flow that passes through the arc (i,j) inside the route r.

The objective function (6.2a) maximises the value of the recourse function $\mathcal{Q}(x)$ expressed as the expected profit collected by the routes (the value of the recourse function will be computed by the second-stage problem described in Section 6.2.3). Since the recourse function is non-linear, it will be linearised using the optimality cuts presented in Section 6.3.1. Constraint (6.2b) guarantees that each vehicle starts its route from the source node and ends to the destination one. Constraints (6.2c) impose the connectivity of each route. Constraints (6.2e) guarantee that a node can only be visited by at most one route. The (6.2d) and (6.2f) are the classical Gavish-Graves (GG) subtour elimination constraints adapted for TOP [43]. Constraints (6.2g) set the lower bound on the *route expectation* for each route. Constraints (6.2h) bound the flow originating by each route from the initial depot. Finally, the (6.2i), (6.2j), (6.2k) and (6.2l) are variable definition constraints.

The model also takes into account the following chance constraints in order to cut off all the solutions that contain routes whose *route reliability* is less than α:

$$\mathbb{P}\left(\sum_{i=0}^{n-1}\sum_{j=1}^{n}\xi_{ij}x_{ijr} \leq T_{\max}\right) \geq \alpha, \quad \forall\, r \in R. \tag{6.3}$$

A detailed integration of the presented chance constraints will be given in Section 6.2.4. After presenting the mathematical formulation for the STOP, the following lemma is stated:

Lemma 6.2.1. *The constraints* (6.2d) *and* (6.2f) *guarantee that every route contained in a feasible integer solution of the first-stage problem is associated with a route reliability greater or equal to 0.5.*

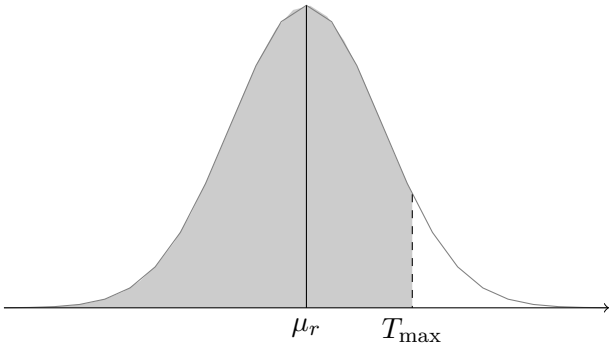


Fig. 6.1: The graphical representation of δ_r.

Proof. The constraints (6.2d) and (6.2f) imply that every integer solution satisfies $\mu_r \leq T_{\max}$ for each $r \in R$. Then, for any value of σ_r^2, the *route reliability* δ_r will always result to be greater or equals to 0.5 because the Normal distribution is symmetric with respect to its expectation (Figure 6.1 graphically motivates the proof of Lemma 6.2.1.). □

6.2.3 The second-stage problem

The second-stage problem of the STOP does not consist in a combinatorial optimisation problem. Instead, it just evaluates the recourse value corresponding to the first-stage solution.

The recourse function for any specific first-stage solution x^ν is defined as $\mathcal{Q}(x^\nu) = \sum_{r \in R} \mathcal{Q}(x_r^\nu)$ where, x_r^ν represents the route r of x^ν and $\mathcal{Q}(x_r^\nu)$ its corresponding evaluation of the recourse function. This quantity is defined as the product between the *route reliability* δ_r and the profit of the route q_r that is, $\mathcal{Q}(x_r^\nu) = \delta_r \cdot q_r$ (the profit of the route r is computed by summing up all the profits of the customers served by r). On the other words, the recourse presented is defined as the expected profit associated with a first-stage solution. In order to linearise the objective function, three different recourse function schemes are considered:

Laporte and Louveaux This scheme [69] computes the value of the recourse function with a family of optimality cuts evaluated on the entire solution x^ν. Essentially, there is a single variable θ that represents the value of the recourse function for a specific first-stage solution x^ν. In this case, the objective function is defined as $\mathcal{Q}(x) = \theta$.

Séguin This scheme [112] decomposes the computation of the value of the recourse function by routes. Essentially, a set of variables ω_k is introduced, each representing the value of the recourse function for any specific route that starts by visiting the node k at the beginning. In this case, the objective function is defined as $\mathcal{Q}(x) = \sum_{k \in \hat{N}} \omega_k$.

Disaggregated This scheme [95] further decomposes the computation of the value of the recourse function by customers. Essentially, a set of variables λ_k is introduced to represent the value of the recourse function for each customer $k \in \hat{N}$. In this case, the objective function is defined as $\mathcal{Q}(x) = \sum_{k \in \hat{N}} \lambda_k$.

6.2.4 The chance constraints

This section describes how the chance constraints (6.3), initially introduced in Section 6.2.2 are embedded into the overall formulation. Including these constraints directly within a model introduces non-linearity, significantly increasing the complexity of the problem. The proposed approach aims to linearise the chance constraints ensuring that the resulting formulation remains linear. Due to the closure property of Normal distributions, it holds that:

$$\sum_{i=0}^{n-1} \sum_{j=1}^{n} \xi_{ij} x_{ijr} \sim \mathcal{N}\left(\sum_{i=0}^{n-1} \sum_{j=1}^{n} \mu_{ij} x_{ijr}, \sum_{i=0}^{n-1} \sum_{j=1}^{n} \sigma_{ij}^2 x_{ijr} \right), \quad \forall\, r \in R.$$

Subtracting the constant $T_{\max}$:

$$\sum_{i=0}^{n-1}\sum_{j=1}^{n}\xi_{ij}x_{ijr} - T_{\max} \sim \mathcal{N}\left(\sum_{i=0}^{n-1}\sum_{j=1}^{n}\mu_{ij}x_{ijr} - T_{\max}, \sum_{i=0}^{n-1}\sum_{j=1}^{n}\sigma_{ij}^2 x_{ijr}\right), \quad \forall\, r \in R.$$

Standardising the previously introduced random variable, the value of the corresponding distribution function evaluated on a generic variable g can be computed as follows:

$$\mathbb{P}\left(\sum_{i=0}^{n-1}\sum_{j=1}^{n}\xi_{ij}x_{ijr} - T_{\max} \leq g\right) = \Phi\left(\frac{g + T_{\max} - \sum_{i=0}^{n-1}\sum_{j=1}^{n}\mu_{ij}x_{ijr}}{\sqrt{\sum_{i=0}^{n-1}\sum_{j=1}^{n}\sigma_{ij}^2 x_{ijr}}}\right), \quad \forall\, r \in R,$$

where, the function Φ represents the standard Normal distribution function. Imposing $g = 0$ for the left part of the equality and recalling the form of the previously introduced chance constraints (6.3), it follows that:

$$\Phi\left(\frac{T_{\max} - \sum_{i=0}^{n-1}\sum_{j=1}^{n}\mu_{ij}x_{ijr}}{\sqrt{\sum_{i=0}^{n-1}\sum_{j=1}^{n}\sigma_{ij}^2 x_{ijr}}}\right) \geq \alpha, \quad \forall\, r \in R.$$

Applying Φ^{-1} on both sides (standardising the denominators):

$$T_{\max} - \sum_{i=0}^{n-1}\sum_{j=1}^{n}\mu_{ij}x_{ijr} \geq \Phi^{-1}(\alpha)\sqrt{\sum_{i=0}^{n-1}\sum_{j=1}^{n}\sigma_{ij}^2 x_{ijr}}, \quad \forall\, r \in R.$$

Squaring both sides (Φ^{-2} represents the square of the Φ^{-1} function) and focusing on the left-hand side of the bi-implication:

$$\left(T_{\max} - \sum_{i=0}^{n-1}\sum_{j=1}^{n}\mu_{ij}x_{ijr}\right)^2 \geq \Phi^{-2}(\alpha)\sum_{i=0}^{n-1}\sum_{j=1}^{n}\sigma_{ij}^2 x_{ijr}, \quad \forall\, r \in R.$$

Expanding the left square:

$${T_{\max}}^2 + \left(2\sum_{(p,q)\in\Omega}\mu_{p_1p_2}\mu_{q_1q_2}x_{p_1p_2r}x_{q_1q_2r} + \sum_{i=0}^{n-1}\sum_{j=1}^{n}\mu_{ij}^2 x_{ijr}^2\right) - 2\,T_{\max}\sum_{i=0}^{n-1}\sum_{j=1}^{n}\mu_{ij}x_{ijr},$$

where, Ω is an arcs set containing all the combinations of arcs $p = (p_1, p_2)$ and $q = (q_1, q_2)$ taken from $\hat{A}$. In this way, all the double products involved in a square

of a polynomial are considered. From the previous term, two non-linear terms can be easily identified:

$$2 \sum_{(p,q)\in\Omega} \mu_{p_1p_2}\mu_{q_1q_2}x_{p_1p_2r}x_{q_1q_2r} \quad \text{and} \quad \sum_{i=0}^{n-1}\sum_{j=1}^{n} \mu_{ij}^2 x_{ijr}^2. \tag{6.4}$$

The pair (p, q) represent a couple of arcs while, p_1, p_2 and q_1, q_2 are the nodes associated to p and q, respectively. Both non-linear terms are linearised using the classical technique for the product of two binary variables. To achieve this, a set of non-negative continuous variables, $g_{p_1p_2q_1q_2r}$ for the first term and b_{ijr} for the second, are introduced. After that, the following sets of linear constraints are embedded into (6.2.2):

$$g_{p_1p_2q_1q_2r} \leq x_{p_1p_2r}, \quad \forall\, r \in R, \tag{6.5}$$
$$g_{p_1p_2q_1q_2r} \leq x_{q_1q_2r}, \quad \forall\, r \in R, \tag{6.6}$$
$$g_{p_1p_2q_1q_2r} \geq x_{p_1p_2r} + x_{q_1q_2r} - 1, \quad \forall\, r \in R, \tag{6.7}$$
$$b_{ijr} \leq x_{ijr}, \quad \forall\, r \in R, \tag{6.8}$$
$$b_{ijr} \geq 2x_{ijr} - 1, \quad \forall\, r \in R, \tag{6.9}$$

Constraints (6.5), (6.6) and (6.7) represent the linearisation of the left term of (6.4) while, constraints (6.8) and (6.9) represent the linearisation of the right one. To summarise, the amount of variables and linear constraints added to the formulation of the first-stage problem (6.2.2) is of the order $\mathcal{O}(|N|^4)$.

6.3 An exact method based on the three-index formulation

This section presents an exact Integer L-shaped method for solving the STOP. It introduces a tailored set of optimality cuts specific to each scheme, along with static valid inequalities for the first-stage problem, both of which are essential for ensuring the convergence of the algorithm.

6.3.1 Optimality cuts

A new family of optimality cuts is introduced for each scheme. They are designed to bound the value of the objective function.

Laporte and Louveaux Given a feasible first-stage solution x^ν, the Laporte and Louveaux optimality cuts can be formulated as follows:

$$\theta \leq \mathcal{Q}\left(x^{\nu}\right) \overbrace{\left(\sum_{r \in R}\left(\sum_{(i,j) \in A(x_r^{\nu})} x_{ijr}\right) - |A(x_r^{\nu})| + 1\right)}^{t_1} + \mathrm{P}_{\mathrm{UB}}^{m} \overbrace{\left(\sum_{r \in R}\left(|A(x_r^{\nu})| - \sum_{(i,j) \in A(x_r^{\nu})} x_{ijr}\right)\right)}^{t_2}, \quad (6.10)$$

where, $\mathrm{P}_{\mathrm{UB}}^{m}$ represents an upper bound for the recourse function of a solution as described in Section 6.3.3. The validity of (6.10) is demonstrated in Lemma 6.3.1:

Lemma 6.3.1. *The Laporte and Louveaux optimality cuts* (6.10) *do not cut off any optimal solution.*

Proof. Consider τ to be such an inequality (6.10). There are two different situations:

- If τ is active in x^{ν} then, it follows that $t_2 = t_1 - 1 = 0$. Hence, (6.10) simplifies to $\theta \leq \mathcal{Q}(x^{\nu})$, which means that the variable θ is bounded by the recourse function evaluated for the solution x^{ν}.

- If τ is not active in x^{ν} then, the variable θ must not be underestimated. To ensure this, it is necessary that:

  $$\mathrm{P}_{\mathrm{UB}}^{m} \geq \mathcal{Q}\left(x^{\nu}\right) \quad \text{and} \quad t_2 \geq t_1.$$

  The left inequality trivially holds because $\mathrm{P}_{\mathrm{UB}}^{m}$ represents an upper bound for the recourse of a solution. With regard to the right inequality, as the two terms t_1 and t_2 are tied, it holds that:

  $$t_2 \geq 1 \iff t_1 \leq 0.$$

  Finally, since τ is not active in x^{ν} as assumption, it follows that $t_2 \geq 1$ and then $t_2 \geq t_1$.

It follows that τ does not cut any optimal solution. □

Séguin Given a specific route $\bar{r}$ belonging to a feasible first-stage solution x^{ν}, the Séguin optimality cuts can be formulated as follows:

$$\omega_{\bar{r}_1} \leq \mathcal{Q}\left(x_{\bar{r}}^{\nu}\right) \overbrace{\left(\left(\sum_{r \in R} \sum_{(i,j) \in A(\bar{r})} x_{ijr}\right) - |A(\bar{r})| + 1\right)}^{t_1}$$

$$+ \mathrm{P}^1_{\mathrm{UB}} \overbrace{\left(|A(\bar{r})| - \sum_{r \in R} \sum_{(i,j) \in A(\bar{r})} x_{ijr} \right)}^{t_2}, \quad (6.11)$$

where, $\bar{r}_1$ represents the first customer visited by the route $\bar{r}$ and $\mathrm{P}^1_{\mathrm{UB}}$ an upper bound for the recourse of a single route as described in Section 6.3.3. The validity of (6.11) is demonstrated in Lemma 6.3.2:

Lemma 6.3.2. *The Séguin optimality cuts* (6.11) *do not cut off any optimal solution.*

Proof. Consider τ to be such an inequality (6.11). There are two different situations:

- If τ is active in $\bar{r}$ then, it follows that $t_2 = t_1 - 1 = 0$. Hence, (6.10) simplifies to $\omega_{\bar{r}_1} \leq \mathcal{Q}\left(x^{\nu}_{\bar{r}}\right)$, which means that the variable $\omega_{\bar{r}_1}$ is bounded by the recourse function evaluated for the route $\bar{r}$.

- If τ is not active in $\bar{r}$ then, the variable $\omega_{\bar{r}_1}$ must not be underestimated. To ensure this, it is necessary that:

  $$\mathrm{P}^1_{\mathrm{UB}} \geq \mathcal{Q}\left(x^{\nu}_{\bar{r}}\right) \quad \text{and} \quad t_2 \geq t_1.$$

  The left inequality trivially holds because $\mathrm{P}^1_{\mathrm{UB}}$ represents an upper bound for the recourse of a single route. With regard to the right inequality, as the two terms t_1 and t_2 are tied, it holds that:

  $$t_2 \geq 1 \iff t_1 \leq 0.$$

  Finally, since τ is not active in $\bar{r}$ as assumption, it follows that $t_2 \geq 1$ and then $t_2 \geq t_1$.

It follows that τ does not cut any optimal solution. □

Disaggregated Given a specific route $\bar{r}$ belonging to a feasible first-stage solution x^{ν}, the Disaggregated optimality cuts can be formulated as follows:

$$\sum_{k \in \hat{N}(\bar{r})} \lambda_k \leq \mathcal{Q}\left(x^{\nu}_{\bar{r}}\right) \overbrace{\left(\sum_{r \in R} \sum_{(i,j) \in \hat{A}(\bar{r})} x_{ijr} - |\hat{A}\left(\bar{r}\right)| + 1 \right)}^{t_1}$$
$$+ \beta_{\hat{N}(\bar{r})} \overbrace{\left(|\hat{A}\left(\bar{r}\right)| - \sum_{r \in R} \sum_{(i,j) \in \hat{A}(\bar{r})} x_{ijr} \right)}^{t_2}, \quad (6.12)$$

where, $\beta_{\hat{N}(\bar{r})}$ represents the sum of the recourse functions of the customers of $\bar{r}$ assuming that each of them is visited by a different route. Accordingly to the triangle inequality (6.1), this quantity represents an upper bound for the sum of recourses of the customers of $\bar{r}$. Mathematically, it is defined as $\beta_{\hat{N}(\bar{r})} = \sum_{k \in \hat{N}(\bar{r})} p_k \cdot \delta_k$ where p_k is the profit associated with the customer k and δ_k represents the *route reliability* of the so-called *singleton route* associated with the customer k defined as $\delta_k = \mathbb{P}(\xi_{1kr} + \xi_{knr} \leq T_{\max})$. Every *singleton route* is defined to be a route that only visits a specific customer. The validity of (6.12) is demonstrated in Lemma 6.3.3:

Lemma 6.3.3. *The Disaggregated optimality cuts* (6.12) *do not cut off any optimal solution.*

Proof. Consider τ to be such an inequality (6.12). There are two different situations:

- If τ is active in $\bar{r}$ then, it follows that $t_2 = t_1 - 1 = 0$. Hence, (6.12) simplifies to $\sum_{k \in \hat{N}(\bar{r})} \lambda_k \leq \mathcal{Q}(x^{\nu}_{\bar{r}})$, which means that the sum $\sum_{k \in \hat{N}(\bar{r})} \lambda_k$ is bounded by the recourse function evaluated for the route $\bar{r}$.

- If τ is not active in $\bar{r}$ then, the sum $\sum_{k \in \hat{N}(\bar{r})} \lambda_k$ must not be underestimated. To ensure this, it is necessary that:

$$\beta_{\hat{N}(\bar{r})} \geq \mathcal{Q}(x^{\nu}_{\bar{r}}) \quad \text{and} \quad t_2 \geq t_1.$$

  The left inequality trivially holds because $\beta_{\hat{N}(\bar{r})}$ represents an upper bound for the recourse of the set of customers served by the route $\bar{r}$. With regard to the right inequality, as the two terms t_1 and t_2 are tied, it holds that:

$$t_2 \geq 1 \iff t_1 \leq 0.$$

  Finally, since τ is not active in $\bar{r}$ as assumption, it follows that $t_2 \geq 1$ and then $t_2 \geq t_1$.

It follows that τ does not cut any optimal solution. □

6.3.2 Upper-bounding functionals

A further variant of the previously introduced optimality cuts (only for the Disaggregate scheme), known as Upper-Bounding Functionals (UBFs), is presented. The UBFs build upon and adapt the concept of Lower-Bounding Functionals proposed by Parada et al. [95] originally designed for minimisation problems. They are defined exclusively for the Disaggregated scheme and are stronger than the corresponding optimality cuts, since they disregard the sequence in which customers are visited

within a route. Given a specific route $\bar{r}$ belonging to a feasible first-stage solution x^{ν}, the UBFs can be formulated as follows:

$$\sum_{k \in \hat{N}(\bar{r})} \lambda_k \leq \mathcal{Q}_{\text{UB}}(x^{\nu}_{\bar{r}}) \overbrace{\left(\left(\sum_{r \in R} \sum_{i \in \hat{N}(r)} \sum_{j \in \hat{N}(r)} x_{ijr} \right) - |\hat{N}(\bar{r})| + 2 \right)}^{t_1} + \beta_{\hat{N}(\bar{r})} \overbrace{\left(|\hat{N}(\bar{r})| - 1 - \sum_{r \in R} \sum_{i \in \hat{N}(r)} \sum_{j \in \hat{N}(r)} x_{ijr} \right)}^{t_2}, \quad (6.13)$$

where, $\beta_{\hat{N}(\bar{r})}$ is the same upper bound introduced for the Disaggregated optimality cuts and $\mathcal{Q}_{\text{UB}}(x^{\nu}_{\bar{r}})$ represents an upper bound for the recourse function of any route that visits all the customers of $\bar{r}$. In order to calculate $\mathcal{Q}_{\text{UB}}(x^{\nu}_{\bar{r}})$, two directed subgraphs G_{μ} and G_{σ^2} defined on the sets $N(\bar{r})$ and $A(\bar{r})$ are built.

The first subgraph defines its distances on the arcs expectations μ_{ij} while the second one on the arcs variances σ^2_{ij}. Then, a lower bound H^{μ}_{LB} for the *route expectation* and a lower bound $H^{\sigma^2}_{\text{LB}}$ for the *route variance* are computed by using G_{μ} and G_{σ^2} respectively and following the Helsgaun methodology.

The Helsgaun bound [53], as well as the Held-Karp bound [52], relies on the Lagrangian relaxation of the classical TSP formulation. The subgradient method is employed to solve the Lagrangian dual problem by using a simple decreasing step size and modifying the Lagrangian multipliers in accordance with the approach outlined by Volgenant and Jonker [123].

Subsequently, $\xi_{\text{LB}} \sim \mathcal{N}(H^{\mu}_{\text{LB}}, H^{\sigma^2}_{\text{LB}})$ is defined as a Normal random variable that provides a lower estimation for the *route uncertain cost* ξ_{LB}. By evaluating its cumulative distribution function at $T_{\max}$, an upper bound δ_{UB} for the *route reliability* of any route serving all customers of $\bar{r}$ can be computed:

$$\delta_{\text{UB}} = \mathbb{P}(\xi_{\text{LB}} \leq T_{\max}).$$

Finally, $\mathcal{Q}_{\text{UB}}(x^{\nu}_{\bar{r}})$ is given by:

$$\mathcal{Q}_{\text{UB}}(x^{\nu}_{\bar{r}}) = \delta_{\text{UB}} \cdot q_{\bar{r}}.$$

The UBFs result to be stronger than the optimality cuts because they do not take into account the order of the visited customers inside a route. The validity of (6.13) is demonstrated in Lemma 6.3.4:

Lemma 6.3.4. *The Upper-bounding functionals* (6.13) *do not cut off any optimal solution.*

Proof. Consider τ to be such an inequality (6.13). There are two different situations:

- If τ is active in $\bar{r}$ then, it follows that $t_2 = t_1 - 1 = 0$. Hence, (6.10) simplifies to $\sum_{k \in \hat{N}(\bar{r})} \lambda_k \leq \mathcal{Q}_{\text{UB}}\left(x^{\nu}_{\bar{r}}\right)$, which means that the sum $\sum_{k \in \hat{N}(\bar{r})} \lambda_k$ is bounded by the recourse function evaluated for the route $x^{\nu}_{\bar{r}}$.

- If τ is not active in $\bar{r}$ then, the variable $\sum_{k \in \hat{N}(\bar{r})} \lambda_k$ must not be underestimated. To ensure this, it is necessary that:

$$\beta_{\hat{N}(\bar{r})} \geq \mathcal{Q}_{\text{UB}}\left(x^{\nu}_{\bar{r}}\right) \quad \text{and} \quad t_2 \geq t_1. \tag{6.14}$$

  With regard to the right inequality, as the two terms t_1 and t_2 are tied, it holds that:

$$t_2 \geq 1 \iff t_1 \leq 0.$$

  Finally, since τ is not active in $\bar{r}$ as assumption, it follows that $t_2 \geq 1$ and then $t_2 \geq t_1$. The left inequality is a bit more complicated to demonstrate. From the definitions previously introduced:

$$\mathcal{Q}_{\text{UB}}(x^{\nu}_{\bar{r}}) = \delta_{\text{UB}} \cdot q_{\bar{r}} \quad \text{and} \quad \beta_{\hat{N}(\bar{r})} = \sum_{k \in \hat{N}(\bar{r})} p_k \cdot \delta_k. \tag{6.15}$$

  Without loss of generality, it can be assumed that $p_k = l$ for each $k \in \hat{N}(\bar{r})$ then, the left inequality of (6.14) becomes (using the definitions (6.15)):

$$l \cdot \sum_{k \in \hat{N}(\bar{r})} \delta_k \geq \delta_{\text{UB}} \cdot |\hat{N}(\bar{r})| \cdot l, \tag{6.16}$$

  recalling that $q_{\bar{r}}$ is the sum of the profits of the customers served by $\bar{r}$. Simplifying both sides of (6.16), it is necessary to prove that:

$$\sum_{k \in \hat{N}(\bar{r})} \delta_k \geq \delta_{\text{UB}} \cdot |\hat{N}(\bar{r})|. \tag{6.17}$$

  For each *singleton route* h_k associated with the customers visited by the route x^{ν}_{r}, the *route uncertain cost* $\xi_k \sim \mathcal{N}(\mu_k, \sigma^2_k)$ is defined where, $\mu_k = \mu_{1k} + \mu_{kn}$ and $\sigma^2_k = \sigma^2_{1k} + \sigma^2_{kn}$ represent the *route expectation* and the *route variance* of h_k. Thanks to the *probabilistic triangle inequality* (6.1), it follows that:

$$H^{\mu}_{LB} \geq \mu_k \quad \forall\, k \in \hat{N}(\bar{r}) \quad \text{and} \quad H^{\sigma^2}_{\text{LB}} \geq \sigma^2_k \quad \forall\, k \in \hat{N}(\bar{r}).$$

  Calculating the *route reliability* δ_k of h_k, it holds that:

$$\delta_k \geq \delta_{\text{UB}} \quad \forall\, k \in \hat{N}(\bar{r}).$$

Finally, summing up for each $k \in \hat{N}(\bar{r})$ both sides directly leads to (6.17) which concludes the proof.

□

6.3.3 Static valid inequalities

This section outlines a set of static valid inequalities employed to tighten the value of the objective function. They are subdivided in two types: (i) the *customers upper bounds* and (ii) the *profit upper bounds* (this approach was inspired by the work of El-Hajj et al. [34]). Each of them is subdivided in two versions: (i) *single route* and (ii) *multi route*. C^1_{UB} and P^1_{UB} represent the customers and profit upper bounds for one single route, respectively. C^m_{UB} and P^m_{UB} represent the same bounds considering m routes instead of only one.

All of them are calculated by solving an adaptation of the Continuous Stochastic Team Orienteering Problem (CSTOP) that consists in the continuous relaxation of the STOP. To solve it efficiently, all the inequalities previously introduced in Section 6.3.1 and 6.3.2 are adopted. To calculate C^1_{UB} and C^m_{UB}, the objective function of the CSTOP is modified in order to maximise the number of served customers. For P^1_{UB} and P^m_{UB}, the objective function is left unchanged. Then, the values of C^1_{UB}, C^m_{UB} and P^1_{UB}, P^m_{UB} are cyclically updated until a stopping criteria is met.

A couple of static inequalities is introduced for each scheme:

Laporte and Louveaux

$$\theta \leq P^m_{UB} \quad \text{and} \quad \theta \leq \sum_{r \in R} \sum_{k \in \hat{N}} y_{kr}\, p_k\, \delta_k.$$

Séguin

$$\omega_k \leq P^1_{UB}\, x_{1kr} \quad \forall\, k \in \hat{N} \quad \text{and} \quad \sum_{k \in \hat{N}} \omega_k \leq \sum_{k \in \hat{N}} y_{kr}\, \delta_k, \quad \forall\, r \in R.$$

Disaggregated

$$\sum_{k \in \hat{N}} \lambda_k \leq P^m_{UB} \quad \text{and} \quad \lambda_k \leq y_{kr}\, p_k\, \delta_k, \quad \forall\, k \in \hat{N},\ r \in R.$$

Finally, it is notable to remark that δ_k represents the *route reliability* of the *singleton route* associated with the customer k introduced in Section 6.3.1 for the Disaggregated optimality cuts.

The mathematical formulation of the first-stage problem presented in Section 6.2.2 is further strengthened by adding an additional inequality in order to impose a threshold on the maximal number of served customers in a solution:

$$\sum_{r \in R} \sum_{k \in \hat{N}} y_{kr} \leq \mathrm{C}^{m}_{\mathrm{UB}}.$$

6.3.4 The algorithm

In this section an exact algorithm for solving the STOP based on the Integer L-shaped methodology is proposed. This algorithm is referred to as the three-index Integer L-shaped Algorithm (3ILA) due to the three-index formulation presented in Section 6.2.2. The main idea of the algorithm is to approximate the non-linear term in the objective (the recourse function), which is done by subdividing the problem into a Master Problem (MP) and a Sub-Problem (SP). Essentially, the SP takes a solution given by the MP and generates valid inequalities which will be added to the MP. Thus, the MP is solved again.

The loop continues until an optimal solution is found or until a maximum gap between upper and lower bound is reached. For any optimal relaxed solution ($\overline{x}$, $\overline{y}$, $\overline{z}$), the graph $\overline{G} = (N, \overline{A})$ is built with $\overline{A} = \{(i,j) : \sum_{r \in R} \overline{x}_{ijr} > 0\}$. The graph $\overline{G}$ can contain multiple routes. All these routes are represented by the set $\mathcal{R}$ and enumerated using a variant of the classical Depth First Search (DFS) algorithm (in any integer solution, it holds that $|\mathcal{R}| = m$).

Optimality cuts The value of the objective function is bounded using the *Upper-bounding functionals* (6.3.2) and the *Optimality cuts* (6.3.1). Computing the upper bound on the *route reliability* δ_{UB} as reported in Section 6.3.2, the first inequality is checked for violation. Additionally, the $\phi = 3$ most violated optimality cuts are examined for violation. It is important to remark that the family of *Optimality cuts* adopted strictly depends on the selected scheme (the Laporte and Louveaux optimality cuts are only evaluated for integer feasible solutions).

Subtour elimination constraints All the elementary cycles inside $\overline{G}$ are computed [51] and the following *Subtour Elimination Constraints* (5.9) (adapted for the three-index model presented in Section 6.2) are checked for violation:

$$\sum_{r=1}^{m} \sum_{(i,j) \in U \times U} x_{ijr} \leq \sum_{r=1}^{m} \left(\sum_{i \in U} y_{ir} - y_{kr} \right), \quad \forall\, U \subseteq \{2, \ldots, n-1\},\ k \in U \tag{6.18}$$

Static inequalities All the static valid inequalities presented in Section 6.3.3 are inserted to the formulation.

Implementation The pseudocode of the procedure callback is reported in Algorithm 7 and makes use of the following procedures: (i) `buildGraph`: builds the

Algorithm 7: `3ILA` callback.

Data: $T_{\max}$, μ_{ij}, σ^2_{ij}, with $i, j \in \{1, \dots, n\}$, $\overline{x}$

1 $\overline{G} \leftarrow$ `buildGraph`$(\overline{x})$;
2 $\mathcal{R} \leftarrow$ `buildRoutes`$(\overline{G}, 1, n)$;
3 **for** *route* $r \in \mathcal{R}$ **do**
  `/* Optimality cuts */`
4 $G_\mu, G_{\sigma^2} \leftarrow$ `buildSubGraphs`$(\overline{x}_r, \mu_{ij}, \sigma^2_{ij})$;
5 $H^\mu_{\text{LB}}, H^{\sigma^2}_{\text{LB}} \leftarrow$ `calculateProbBounds`(G_μ, G_{σ^2}) ;
6 $\xi_{\text{LB}} \leftarrow \mathcal{N}(H^\mu_{\text{LB}}, H^{\sigma^2}_{\text{LB}})$;
7 $\delta_{\text{UB}} \leftarrow \mathbb{P}(\xi_{\text{LB}} \leq T_{\max})$;
8 $\mathcal{Q}_{\text{UB}}(\overline{x}_r) \leftarrow \delta_{\text{UB}} \cdot q_r$;
9 check inequality (6.13) associated with $\hat{N}(r)$ and $\mathcal{Q}_{\text{UB}}(\overline{x}_r)$;
10 check ϕ inequalities (6.3.1) associated with r ;
11 **for** *cycle* $c \in \overline{G}$ **do**
    `/* Subtour elimination constraints */`
12 check inequalities (6.18) associated with c ;

graph $\overline{G}$ from a solution, (ii) `buildRoutes`: builds the set $\mathcal{R}$ of routes from the graph $\overline{G}$ with a variant of the DFS algorithm, (iii) `buildSubRoute`: builds the subroute from the subpath p, (iv) `buildSubGraphs`: builds the subgraphs G_μ and G_{σ^2} from the nodes contained in $N(r)$ and the set of arcs $A(r)$, (v) `calculateProbBounds`: calculate the Helsgaun lower bounds for the subgraphs G_μ and G_{σ^2} and (vi) `buildSets`: build the sets $L(p)$ and $R(p)$ from the subpath p. The overall 3ILA callback has been developed as an embedded procedure inside the CPLEX environment, which is called for every node corresponding to an optimal fractional solution.

6.4 A two-index mathematical formulation

The formulation introduced in Section 6.2 directly integrates the chance constraints through a linearisation process that leads to an increase in the number of variables and constraints by an order of $\mathcal{O}(|N|^4)$. Preliminary results reported in Section 6.6 indicate that this formulation is inefficient due to the large number of variables and constraints. Consequently, leveraging the flow-based formulation of the TOP [18], a more compact two-index model for the STOP is presented:

$$\max \quad \mathcal{Q}(x) \tag{6.19a}$$

$$\text{s.t.} \quad \sum_{(1,j)\in\hat{A}} x_{1j} = \sum_{(i,n)\in\hat{A}} x_{in} = m, \tag{6.19b}$$

$$\sum_{(i,k)\in\hat{A}} x_{ik} = \sum_{(k,j)\in\hat{A}} x_{kj} = y_k, \quad \forall\, k \in \hat{N}, \tag{6.19c}$$

$$\sum_{(k,j)\in\hat{A}} z_{kj} - \sum_{(i,k)\in\hat{A}} z_{ik} = \sum_{(k,j)\in\hat{A}} \mu_{kj} x_{kj}, \quad \forall\, k \in \hat{N}, \tag{6.19d}$$

$$z_{ij} \leq (T_{\max} - \mu_{jn})\, x_{ij}, \quad \forall\, (i,j) \in \hat{A}, \tag{6.19e}$$

$$z_{ij} \geq (\mu_{1i} + \mu_{ij})\, x_{ij}, \quad \forall\, (i,j) \in \hat{A}, \tag{6.19f}$$

$$z_{1k} = \mu_{1k} x_{1k}, \quad \forall\, k \in \hat{N}, \tag{6.19g}$$

$$y_k \in \{0,1\}, \quad \forall\, k \in \hat{N}, \quad r \in R, \tag{6.19h}$$

$$x_{ij} \in \{0,1\}, \quad \forall\, (i,j) \in \hat{A} \setminus \{(1,n)\}, \quad r \in R, \tag{6.19i}$$

$$x_{1n} \in \mathbb{N}^+, \quad \forall\, r \in R, \tag{6.19j}$$

$$z_{ij} \in \mathbb{R}^+, \quad \forall\, (i,j) \in \hat{A}, \quad r \in R, \tag{6.19k}$$

The proposed model makes use of the following decision variables: the variable x_{ij} is equal to 1 if and only if the arc (i,j) is traversed, 0 otherwise; the variable y_k is equal to 1 if and only if the node k is visited, 0 otherwise; the variable z_{ij} defines the expected arrival time at node j coming from the node i and can be thought as the amount of flow that passes through the arc (i,j).

The objective function (6.19a) maximizes the recourse value, which corresponds to the expected profit collected by the routes, as for the three-index formulation introduced in Section 6.2.2. Constraint (6.19b) guarantees that each vehicle starts its route from the source node and ends to the destination one. Constraints (6.19c) impose the connectivity of each route. The (6.19d) and (6.19e) are the classical Gavish-Graves (GG) subtour elimination constraints adapted for the TOP [43]. Constraints (6.19f) set the lower bound on the *route expectation*. Constraints (6.19g) bound the flow originating from the initial depot. Finally, the (6.19h), (6.19i), (6.19j) and (6.19k) are the variable definition constraints.

6.5 An exact method based on the two-index formulation

In this section an additional exact Integer L-shaped method is introduced to solve the STOP leveraging the two-index formulation described in Section 6.4. This formulation permits the violation of the chance constraints, as they are not directly embedded within the model itself but dynamically added. Firstly, two distinct lift constraints are presented to strengthen of the mathematical formulation. Secondly, a detailed explanation of the algorithm is given.

6.5.1 Lift constraints

In this section two lift constraints are introduced to further strengthen the formulation of the first-stage problem presented in Section 6.4. The first one, the Time budget

lift, consists in tightening the value of $T_{\max}$ in (6.19e). The second one, the Sedrakyan lift, consists in strengthening the right-hand size of (6.19d), (6.19e), (6.19f) and (6.19g).

Time budget lift In this type of constraints lift, the value of $T_{\max}$ in (6.19e) is tightened without cutting off any feasible solution. The main idea consists in refining the value of $T_{\max}$ by alternating cyclically the calculations of a lower bound for the *route variance* σ^2_{LB} with the calculation of an upper bound for the *route expectation* μ_{UB}. In order to calculate σ^2_{LB}, a modified instance of CSTOP (for a single route) in which the objective function is modified in order to minimise σ^2_{LB} maintaining a minimum value of μ_{UB}.

Now, the value of μ_{UB} is updated by taking the $(1-\alpha)$-quantile of the Normal distribution $\mathcal{N}(T_{\max}, \sigma^2_{\text{LB}})$. The quantile calculated in this way guarantees that any route r with *route expectation* $\mu_r \leq \mu_{\text{UB}}$ does not violate any chance constraint considering a minimum level of *route variance* equal to σ^2_{LB}. Therefore, no feasible solution is cut off. On the other hand, σ^2_{LB} and μ_{UB} are cyclically updated until a convergence for both values is reached (initially, $\mu_{\text{UB}} = T_{\max}$).

Algorithm 8: `The Time budget lift` procedure.

Data: $T_{\max}$, α
Result: μ_{UB}
1 $\mu_{\text{UB}} \leftarrow T_{\max}$;
2 **while** *not convergence* $(\mu_{UB}, \sigma^2_{LB})$ **do**
3 $\quad \sigma^2_{\text{LB}} \leftarrow$ `RVLB` (μ_{UB}) ;
4 $\quad \mu_{\text{UB}} \leftarrow$ `quantile` $(\mathcal{N}(T_{\max}, \sigma^2_{\text{LB}}), 1 - \alpha)$;
5 **return** μ_{UB};

The pseudocode of the procedure is reported in Algorithm 8 and makes use of the following procedures: (i) `RVLB`: calculates the lower bound for the *route variance* given a specified route expectation given as a parameter and (ii) `quantile`: calculates the specified quantile for the given distribution.

Sedrakyan lift In this type of constraints lift, the aim is to strengthen the right-hand size of (6.19d), (6.19e), (6.19f) and (6.19g) by inserting the variance information inside them. Given a generic first-stage solution x, all the violated chance constraints for any route r can be rephrased into these non-linear constraints accordingly to Hillier [55]:

$$\sum_{(i,j)\in A(r)} \mu_{ij} x_{ij} + q^{\mathcal{N}}_{\alpha} \sqrt{\sum_{(i,j)\in A(r)} \sigma^2_{ij} x_{ij}} \leq T_{\max}, \tag{6.20}$$

where, $q_\alpha^{\mathcal{N}}$ represents the α-quantile of a standard Normal distribution $\mathcal{N}(0,1)$. The idea is to linearly relax the previous inequality introduced. This is accomplished through the adoption of a useful inequality credited to Sedrakyan [107]:

$$\sum_{k=1}^{t} \frac{a_k^2}{b_k} \geq \frac{\left(\sum_{k=1}^{t} a_k\right)^2}{\sum_{k=1}^{t} b_k} \quad \forall\, t \in \mathbb{N}^+, \quad a_1, \ldots, a_t \in \mathbb{R}, \quad b_1, \ldots, b_t \in \mathbb{R}^+.$$

Thus, imposing $t = |A(r)|$, $b_k = 1$ and $a_k = \sigma_{ij} x_{ij}$, it follows that:

$$\sum_{(i,j)\in A(r)} \sigma_{ij}^2 x_{ij} \geq \frac{\left(\sum_{(i,j)\in A(r)} \sigma_{ij} x_{ij}\right)^2}{|A(r)|}.$$

Taking the square root of both sides:

$$\sqrt{\sum_{(i,j)\in A(r)} \sigma_{ij}^2 x_{ij}} \geq \frac{\sum_{(i,j)\in A(r)} \sigma_{ij} x_{ij}}{\sqrt{|A(r)|}}.$$

Then, the constraint in (6.20) can be linearly relaxed by replacing the square root term with the right-hand side of the previous linear inequality:

$$\sum_{(i,j)\in A(r)} \mu_{ij} x_{ij} + q_\alpha^{\mathcal{N}} \frac{\sum_{(i,j)\in A(r)} \sigma_{ij} x_{ij}}{\sqrt{|A(r)|}} \leq T_{\max},$$

where, $q_\alpha^{\mathcal{N}}$ is a non-negative quantity for $\alpha \geq 0.5$. Finally, the previous linear inequality can be rephrased in order to replace the inequalities (6.19d), (6.19e), (6.19f) and (6.19g) respectively of the mathematical formulation of the first-stage problem reported in Section 6.4:

$$\sum_{(k,j)\in \hat{A}} z_{kj} - \sum_{(i,k)\in \hat{A}} z_{ik} = \sum_{(k,j)\in \hat{A}} \left(\mu_{kj} + q_\alpha^{\mathcal{N}} \frac{\sigma_{kj}}{\sqrt{(C_{\text{UB}}^1 + 1)}}\right) x_{kj}, \quad \forall\, k \in \hat{N},$$

$$z_{ij} \leq \left(T_{\max} - \mu_{jn} - q_\alpha^{\mathcal{N}} \frac{\sigma_{jn}}{\sqrt{(C_{\text{UB}}^1 + 1)}}\right) x_{ij}, \quad \forall\, (i,j) \in \hat{A},$$

$$z_{ij} \geq \left(\mu_{1i} + \mu_{ij} + q_\alpha^{\mathcal{N}} \frac{\sigma_{1i} + \sigma_{ij}}{\sqrt{(C_{\text{UB}}^1 + 1)}}\right) x_{ij}, \quad \forall\, (i,j) \in \hat{A},$$

$$z_{1k} = \left(\mu_{1k} + q_\alpha^{\mathcal{N}} \frac{\sigma_{1k}}{\sqrt{(C_{\text{UB}}^1 + 1)}}\right) x_{1k}, \quad \forall\, k \in \hat{N},$$

where, the term $|A(r)|$ has been replaced by $(C_{\text{UB}}^1 + 1)$ that represents an upper bound for the number of arcs composing a single route.

6.5.2 The algorithm

In this section another exact algorithm for solving the STOP based on the Integer L-shaped methodology is proposed. This algorithm is referred to as the two-index Integer L-shaped algorithm (2ILA). For any optimal relaxed solution ($\overline{x}$, $\overline{y}$, $\overline{z}$), the graph $\overline{G} = (N, \overline{A})$ is built with $\overline{A} = \{(i,j) : \sum_{r \in R} \overline{x}_{ijr} > 0\}$. The graph $\overline{G}$ can contain multiple routes. All these routes are represented by the set $\mathcal{R}$ and enumerated using a variant of the classical Depth First Search (DFS) algorithm (in any integer solution, it holds that $|\mathcal{R}| = m$).

Feasibility cuts The formulation presented in Section 6.4 does not include the chance constraints, allowing solutions that may violate them. Then, the goal consists in separating them dynamically using a set of feasibility cuts. The first family of feasibility cuts considered are the *Subpath inequalities* (5.7) and (5.8). To separate them, all the routes $r \in \mathcal{R}$ with a *route reliability* less than α ($\delta_r < \alpha$) are considered. For each feasible subpath of customers p within r, the corresponding subroute s is constructed by appending the source node at the beginning and the destination node at the end of p. If δ_s is greater than or equal to α, the sets $L(p)$ and $R(p)$ are generated, and the associated inequalities are checked for potential violations.

The *Set inequalities* (5.6) and *Route inequalities* (5.5) are then evaluated. For each route $r \in \mathcal{R}$, an upper bound on the *route reliability* δ_{UB} is computed as outlined in Section (6.3.2). If δ_{UB} is less than α, the first inequality is checked for violation. Otherwise, the second inequality is examined for violation.

Optimality cuts A two-index adaptation of the optimality cuts introduced in Section 6.3.1 is considered.

Subtour elimination constraints All the elementary cycles inside $\overline{G}$ are computed [51] and the associated *Subtour Elimination Constraints* (5.9) are checked for violation.

Static inequalities A two-index adaptation of the static valid inequalities presented in Section 6.3.3 are inserted to the formulation.

Lift Constraints One of the two lift constraints reported in Section 6.5.1 is adopted (they are mutually exclusive).

Implementation The pseudocode of the procedure callback is reported in Algorithm 9 and makes use of the following procedures: (i) `buildGraph`: builds the graph $\overline{G}$ from a solution, (ii) `buildRoutes`: builds the set $\mathcal{R}$ of routes from the graph $\overline{G}$ with a modified version of the DFS algorithm, (iii) `buildSubRoute`: builds the subroute from the subpath p, (iv) `buildSubGraphs`: builds the subgraphs G_μ and G_{σ^2} from

Algorithm 9: `2ILA` callback.

Data: $T_{\max}$, μ_{ij}, σ^2_{ij}, with $i, j \in \{1, \ldots, n\}$, $\overline{x}$

$\overline{G} \leftarrow$ `buildGraph`$(\overline{x})$;
$\mathcal{R} \leftarrow$ `buildRoutes`$(\overline{G}, 1, n)$;
for *route* $r \in \mathcal{R}$ **do**
  $G_\mu, G_{\sigma^2} \leftarrow$ `buildSubGraphs`$(\overline{x}_r, \mu_{ij}, \sigma^2_{ij})$;
  $H^{\mu}_{\text{LB}}, H^{\sigma^2}_{\text{LB}} \leftarrow$ `calculateProbBounds`(G_μ, G_{σ^2}) ;
  $\xi_{\text{LB}} \leftarrow \mathcal{N}(H^{\mu}_{\text{LB}}, H^{\sigma^2}_{\text{LB}})$;
  $\delta_{\text{UB}} \leftarrow \mathbb{P}(\xi_{\text{LB}} \leq T_{\max})$;
  if $\delta_r < \alpha$ **then**
    `/* Feasibility cuts */`
    for *subpath* $p \in \overline{x}_r$ **do**
      if $1 \notin p$ *and* $n \notin p$ **then**
        $s \leftarrow$ `buildSubRoute`$(p, 1, n)$;
        if $\delta_s \geq \alpha$ **then**
          $L(p), R(p) \leftarrow$ `buildSets`$(p, N, T_{\max})$;
          check inequality (5.7) associated with p and $L(p)$;
          check inequality (5.8) associated with p and $R(p)$;
    if $\delta_{UB} < \alpha$ **then**
      check inequality (5.6) associated with $\hat{N}(r)$;
    else
      check inequality (5.5) associated with r ;
  else
    `/* Optimality cuts */`
    $\mathcal{Q}_{\text{UB}}(\overline{x}_r) \leftarrow \delta_{\text{UB}} \cdot q_r$;
    check inequality (6.13) associated with $\hat{N}(r)$ and $\mathcal{Q}_{\text{UB}}(\overline{x}_r)$;
    check ϕ inequalities (6.3.1) associated with r ;
for *cycle* $c \in \overline{G}$ **do**
  `/* Subtour elimination constraints */`
  check inequalities (5.9) associated with c ;

the nodes contained in $N(r)$ and the set of arcs $A(r)$, (v) `calculateProbBounds`: calculate the Helsgaun lower bounds for the subgraphs G_μ and G_{σ^2} and (vi) `buildSets`: build the sets $L(p)$ and $R(p)$ from the subpath p. The overall 2ILA callback has been developed as an embedded procedure inside the CPLEX environment, which is called for every node corresponding to an optimal fractional solution.

6.6 Quantitative analysis

This section offers a deep quantitative analysis in order to validate the performance of the proposed formulations and algorithms. It starts by giving a brief description of the preliminary tests conducted in order to prove the inefficiency of the three-index

model (Sect. 6.2) and the corresponding 3ILA (Sect. 6.3) with respect to the two-index formulation (Sect. 6.4) solved by the 2ILA (Sect. 6.5.2). After that, additional tests have been performed in order to measure the impact of the Upper-Bounding Functionals (UBFs) on improving performance. Finally, the section provides detailed insights about the performance and computational efficiency of the 2ILA. In this regard, a comprehensive examination of the results derived from the experiments on the complete TOP reference dataset is provided. The analysis investigates the solution quality, the computational time and the scalability of the approach. Furthermore, it is notable to remark that all tests were conducted providing the solver with a lower cutoff (given by a simple metaheuristic algorithm) in order to enhance computational efficiency. Finally, the parameter α is set to 0.95 in order to ensure that only highly reliable routes are considered.

The 2ILA and the 3ILA have been implemented in C++ (CPLEX 22.1.1 Concert Technology is adopted for implementing the callbacks) and run in single-threaded mode. The code was compiled in Ubuntu 22.04. The experiments were carried out on a 64-bit Linux machine, with an AMD EPYC 7532 (Zen 2), 2.40 GHz and 24 GB of RAM. CPLEX built-in cuts have been used in all experiments. Finally, the solver is run in a fully-deterministic mode in order to be able to reproduce the results (if the same platform is used).

6.6.1 Comparing mathematical formulations

In this section, a brief comparison between the two mathematical formulations for the STOP is reported in Table 6.1. The results are based on the first three benchmark sets proposed by Chao et al. [24] (Set 1, 2 and 3). These sets are characterised by 32, 21 and 33 nodes, respectively and they are categorised based on the number of available vehicles m (ranging from 2 to 4).

The column OPT reports the number of instances optimally solved, CPU(s) the average CPU time (in seconds) used by CPLEX to certify the optimality, and NODES the average number of nodes. The results reported in Table 6.1 show that the 2-index

Tab. 6.1: The comparison between the two formulations (and algorithms) for the STOP.

		3ILA (3-Index formulation)				2ILA (2-Index formulation)			
m	#	OPT	CPU(s)	NODES	GAP(%)	OPT	CPU(s)	NODES	GAP(%)
2	49	17	83.82	886.18	21.77	31	111.41	26554.42	6.49
3	49	26	44.78	1028.73	19.90	36	49.53	22202.64	8.93
4	49	32	34.08	910.69	22.47	43	44.70	29147.56	7.31

formulation (in combination with the 2ILA) completely dominates the three-index formulation (solved by the 3ILA). In fact, the number of instances optimally solved results always in favour to the compact two-index formulation. A similar trend is

observed for the average mip gap. With regard to the average running time, the 2ILA is slightly slower than the 3ILA in proving optimality, which can be attributed to the higher number of optimally solved instances. Finally, the 2ILA consistently uses a higher average number of nodes. An interesting observed behavior is that as the number of available vehicles increases, the STOP appears easier to solve. This is likely because, in instances with a small $T_{\max}$ value, the chance constraints limit the feasible search space, simplifying the problem.

6.6.2 The impact of the upper-bounding functionals

This section provides a brief assessment of the impact of the Upper-Bounding Functionals (UBFs) introduced in Section 6.3.2. Table 6.2 displays the results of the two 2ILA versions (with and without UBFs) and is organised into two parts. A presented in Section 6.5.2 without separating the UBFs. The right one, reports the results for the same algorithm but separating the UBFs. The results are further subdivided by instances size (the ALL row groups all the results). The columns OPT, CPU(s),

Tab. 6.2: Computational results without (left) and with (right) separating the UBFs.

		2ILA (without UBFs)				2ILA (with UBFs)			
	#	OPT	CPU(s)	NODES	GAP(%)	OPT	CPU(s)	NODES	GAP(%)
SMALL	147	104	51.34	72861.52	11.92	110	351.63	67515	7.48
MEDIUM	120	52	18.99	55922.71	25.69	60	650.27	57395	11.16
LARGE	120	31	24.95	34343.29	17.03	35	878.00	31462	11.62
ALL	387	187	75.94	55665.54	18.88	205	607.44	53198	10.62

NODES and GAP(%) report the number of optimal instances solved, the average running time (in seconds) of the algorithm for the solved instances, the average number of nodes and the average mip gap, respectively.

As shown in the table, the UBFs have a significant impact on the performance of the 2ILA in particular for the medium and large instances where the 2ILA (with the UBFs) is able to certify the optimality much faster. The number of optimally solved instances is always in favour to the 2ILA (with the UBFs), proving the optimality of 18 additional instances. Finally, with regards to the average mip gap, the 2ILA (with the UBFs) achieve a much better average mip gap.

6.6.3 Results on the TOP reference dataset

The main purpose of this section is to report the extensive computational analysis of the 2ILA method on the TOP reference dataset available online at `https://www.mech.kuleuven.be/en/cib/op`. This benchmark set proposed by Chao et al. [24] is composed of 387 instances divided into 7 sets in accordance with the number of nodes ranging from 21 to 102. Each set differs from the number of teams (vehicles) available to serve the customers (from 2 to 4) and for the value of the maximum

time duration of a route. The instances are composed by nodes represented though two-dimensional coordinates in a Euclidean plane. In order to present the results, the TOP instances were grouped into three different benchmark sets. The *small* benchmark set is composed of the instances belonging to Set 1, 2 and 3 for a grand total of 147 instances. The *medium* benchmark set is composed of the instances belonging to Set 5 and 6 for a grand total of 120 instances. The *large* benchmark set is composed of the instances belonging to Set 4 and 7 for a grand total of 120 instances.

To represent the stochastic travel times between two nodes i and j, the random variables $\xi_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$ were defined by setting:

$$\mu_{ij} = t_{ij} \quad \text{and} \quad \sigma_{ij}^2 = \sqrt{t_{ij}},$$

where, the quantity t_{ij} describes the deterministic travel time between the nodes i and j calculated by using the two-dimensional Euclidean distance formula.

Tables 6.3, 6.4 and 6.5 report the computational results for the *small*, *medium* and *large* benchmark sets, respectively. Table 6.6 gives a summary of the computational results for all the instances. The tables are organised in four parts. The first and the second parts report the scheme and the constraints lift adopted, respectively. The third and the fourth parts describe the statistics for the 2ILA method and for the upper bounds, respectively. The columns OPT, GAP(%), NODES and CUTS report the number of optimal instances solved, the average mip gap, the average number of explored nodes and the average number of cuts added, respectively. The column $P_{UB}^m(\%)$ reports the average relative differences between the *profit upper bound* and the value of an optimal solution. The column $C_{UB}^m(\%)$ reports instead the average relative differences between the *customers upper bound* and the number of visited customers in an optimal solution.

Tab. 6.3: Computational results for small instances (147 in total). The schemes considered are: (1) Laporte and Louveaux, (2) Séguin, (3) Disaggregated. The lift constraints considered are: (A) Sedrakyan, (B) Time budget, (C) No lift.

Scheme			Lift			2ILA					Upper Bounds		
1	2	3	A	B	C	OPT	CPU(s)	GAP(%)	NODES	CUTS	$P_{UB}^m(\%)$	$C_{UB}^m(\%)$	CPU(s)
✓			✓			107	43.53	7.46	66270	736	7.13	22.50	0.40
✓				✓		102	56.83	11.75	71623	816	8.29	23.99	0.37
✓					✓	86	28.88	24.27	92265	1345	9.26	27.79	0.38
	✓		✓			109	61.43	7.82	65742	672	7.37	22.55	0.40
	✓			✓		101	48.78	11.57	68848	802	8.09	23.90	0.34
	✓				✓	86	33.42	24.28	90336	1350	9.22	27.79	0.35
		✓	✓			110	65.08	7.48	67515	643	7.39	22.46	0.39
		✓		✓		102	36.24	11.60	71526	762	8.28	23.99	0.34
		✓			✓	87	51.87	25.01	94006	1286	9.37	27.75	0.37

Tab. 6.4: Computational results for medium instances (120 in total). The schemes considered are: (1) Laporte and Louveaux, (2) Séguin, (3) Disaggregated. The lift constraints considered are: (A) Sedrakyan, (B) Time budget, (C) No lift.

Scheme			Lift			2ILA					Upper Bounds		
1	2	3	A	B	C	OPT	CPU(s)	GAP(%)	NODES	CUTS	P^m_{UB}(%)	C^m_{UB}(%)	CPU(s)
✓			✓			57	38.88	10.96	55956	1506	12.83	21.05	7.08
✓				✓		52	17.81	25.94	52645	1588	16.44	25.07	7.27
✓					✓	50	30.74	55.82	56769	2682	24.42	32.90	9.01
	✓		✓			59	72.33	11.26	59025	1235	13.49	20.37	6.32
	✓			✓		52	23.29	25.73	55482	1515	16.37	25.07	6.20
	✓				✓	49	18.77	55.35	57416	2535	22.94	32.35	8.80
		✓	✓			60	81.00	11.16	57395	1211	13.37	20.09	6.79
		✓		✓		52	25.30	25.73	54983	1525	16.34	25.07	7.02
		✓			✓	48	1.97	54.79	55946	2777	22.18	31.24	8.39

Tab. 6.5: Computational results for large instances (120 in total). The schemes considered are: (1) Laporte and Louveaux, (2) Séguin, (3) Disaggregated. The lift constraints considered are: (A) Sedrakyan, (B) Time budget, (C) No lift.

Scheme			Lift			2ILA					Upper Bounds		
1	2	3	A	B	C	OPT	CPU(s)	GAP(%)	NODES	CUTS	P^m_{UB}(%)	C^m_{UB}(%)	CPU(s)
✓			✓			34	74.01	11.33	37485	989	9.08	25.12	167.95
✓				✓		31	20.80	16.96	34583	1114	10.74	26.91	137.89
✓					✓	30	92.39	27.54	38043	1506	11.71	28.64	152.62
	✓		✓			33	49.48	11.35	34763	1060	9.08	24.81	162.97
	✓			✓		31	21.49	17.10	33348	1153	10.74	26.58	119.18
	✓				✓	30	55.36	27.41	37601	1512	11.76	28.34	126.02
		✓	✓			35	90.35	11.62	31462	1062	9.89	25.27	138.41
		✓		✓		31	24.12	17.00	31760	1139	10.69	26.58	105.44
		✓			✓	29	30.78	27.35	35811	1519	11.14	28.37	111.82

Finally, the two columns CPU(s) describe the average running time (in seconds) to prove the optimality (for the 2ILA method) and the average running time (in seconds) for computing the upper bounds. In case the Disaggregated scheme is adopted combined with the Sedrakyan lift, the 2ILA method is capable to solve approximately 75%, 50% and 30% of the *small*, *medium* and *large* benchmark sets, respectively. The average cpu running time required to prove the optimality of the instances is similar across each benchmark set. This is because, although the small benchmark set is smaller than the others, fewer instances are solved in the medium and large benchmark sets. The average mip gap appears slightly lower for the small benchmark set, while it tends to increase for the medium and large sets. The average number of branching nodes and cuts is comparable across all benchmark sets. Regarding the upper bounds section of the table, the average *profit upper bound* and *customers upper bound* are also quite similar across the benchmark sets, for the same reason discussed for the average cpu running time. As evident, the average cpu time for computing the upper bounds increases with the size of the instances. With regard to the results for all the instances, the algorithm is able to solve approximately the 53% of the instances in case the Disaggregated scheme is adopted in combinations with the Sedrakyan lift. The Disaggregated scheme seems

Tab. 6.6: Computational results for all instances (387 in total). The schemes considered are: (1) Laporte and Louveaux, (2) Séguin, (3) Disaggregated. The lift constraints considered are: (A) Sedrakyan, (B) Time budget, (C) No lift.

Scheme			Lift			2ILA					Upper Bounds		
1	2	3	A	B	C	OPT	CPU(s)	GAP(%)	NODES	CUTS	P^m_{UB}(%)	C^m_{UB}(%)	CPU(s)
✓			✓			198	47.42	10.39	54146	1053	8.46	22.68	54.43
✓				✓		185	39.83	18.82	54253	1148	9.85	24.59	45.15
✓					✓	166	40.92	35.59	64445	1810	11.99	28.71	50.26
	✓		✓			201	62.67	10.60	54053	967	8.72	22.50	52.65
	✓			✓		184	36.98	18.74	53696	1132	9.71	24.48	39.01
	✓				✓	165	33.06	35.49	63776	1768	11.64	28.54	41.94
		✓	✓			205	74.05	10.62	53198	949	8.87	22.48	45.17
		✓		✓		185	31.14	18.74	54066	1115	9.82	24.54	35.00
		✓			✓	164	33.54	35.58	64159	1821	11.41	28.33	37.42

to produce the best results in terms of number of optimalities while, the Sedrakyan and the Time budget lifts clearly outperform the model with no constraints lift with respect to almost every measure.

6.7 Conclusions

This chapter introduced the Stochastic Team Orienteering Problem (STOP), an extension of the standard TOP that addresses the uncertainty in travel times through a set of Normal random variables. An initial mathematical formulation was developed in combination with an exact algorithm based on the Integer L-shaped methodology. Furthermore, a complete linearisation procedure for the chance constraint is presented as a part of the formulation itself. A two-index formulation for the STOP was then introduced with an additional exact algorithm that adopts the same approach. Preliminary experiments demonstrated the efficiency of the two-index formulation (and the corresponding algorithm). Extensive computational results on the TOP benchmark dataset were presented in order to evaluate the performance of the two-index formulation though different schemes. Two different constraint lifts are introduced to strengthen the formulation. Finally, a comparative analysis confirmed the effectiveness of the newly introduced Upper-Bounding Functionals (UBFs).

7 Solutions for the TOP-ST-MIN under uncertainty

This chapter explores the TOP-ST-MIN under uncertainty conditions. It starts by presenting a stochastic and dynamic variant of the problem that extends the STOP introduced in Chapter 6. This variant is considerably more complex since it incorporates uncertain in service and travel times and dynamic factors such as the arrival of new customers, new mandatory nodes and the emergence of new physical or logical incompatibilities. In order to overcome these challenges, it is essential to employ methodologies that adapt to dynamic situations by ensuring efficiency. The Branch & Regret paradigm [56] and [27] is a scenario-driven approach designed for dynamic problems, combining Discrete Event Simulation (DES) with online optimisation. It provides an effective framework for handling dynamicity by integrating a branching mechanism that helps to choose a specific high-level feasible action with a consensus-based function that evaluates outcomes across various scenarios, interpreted as potential future requests.

This chapter introduces four different algorithms based on the Branch & Regret approach. Extensive computational results are presented in order to validate the proposed methodologies providing insights into their performance. It is important to remark that part of the notation used in this chapter has been introduced in Section 5.1.

This chapter was made in collaboration with Jean-François Côté from Université Laval and Alessandro Druetto from University of Turin.

7.1 The problem

This section introduces the Dynamic and Stochastic Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (DSTOP-ST-MIN) as the dynamic and stochastic variant of the TOP-ST-MIN. The DSTOP-ST-MIN generalises the STOP by considering the travel and service times as Normal random variables and incorporates the following dynamic aspects:

1. New customer requests can arise in the future. For example, emergency calls for medical assistance can come in at any time.

2. Some customers may become mandatory. For example, when the health conditions of some patient may worsen.

3. A connection between two nodes can be deleted. For example, a road may be deleted from a network if it is closed due to construction or severe weather conditions.

4. A pair of nodes can become logically incompatible when certain conditions prevent them from being served together. For example, when some patients contract an infection such as COVID-19, they must be served separately from non-infected individuals.

5. A vehicle concludes a customer's request.

The aim of the DSTOP-ST-MIN is to identify a set of m routes that maximise the expected profit from the served customers respecting all the chance constraints introduced for the STOP in chapter 6 and all the TOP-ST-MIN related constraints introduced in Chapter 5. Following the same classification done for the TOP-ST-MIN, the DSTOP-ST-MIN-P only accounts for physical incompatibilities while the DSTOP-ST-MIN-PL considers both types of incompatibilities.

7.2 The instances adaptation

The instance generation procedure outlined in Chapter 4 has been adapted in order to fulfil new requirements. The customer requests from the original TOP-ST-MIN instances were divided into two distinct subsets. The first subset represents the initial set of customer requests, while the second one corresponds to future requests for which the arrival time is unknown. Unlike for the STOP, where a general scenario is represented by a complete realisation of all the random variables involved in the problem, in this case the notion of scenario is introduced in order to capture additional information derived from prior knowledge. For example, consider a company that provides weekly hospitalisation services. To this end, a scenario may represent a subset of frequent patients requests likely to arise together. The algorithms described in Section 7.3 leverage a set of scenarios to support the decision-making process.

Except for the initial customer requests and scenarios, all other information are timestamped in order to indicate the moment when the corresponding event will be processed from the DES queue. The initial customer requests set comprises 40% of the requests of the original TOP-ST-MIN instances and is generated using a discrete uniform probability distribution (the remaining 60% are future requests). A similar approach is applied to the scenarios, where the number of customers in each scenario is set to 30% of the future requests. The number of scenarios is set to 10 (a uniform probability mass ν_s of $1 \,/\, 10$ is assigned to each scenario s). The sequence of future requests has been generated accordingly to a Poisson process with intensity $\lambda = 10 \,/\, T_{\max}$. The same logic follows for the future mandatory requests

and future physical and logical incompatibilities. Moreover, the set of customers is subdivided into *known customers*, representing all known customer requests ($g_k = 1$ if the customer k is known, 0 otherwise) and *future customers*, which consist of all customer requests that will arrive in the future. Finally, the *scenario customers* represent all of the customer requests belonging to a specific scenario.

7.3 The algorithms

In this section four Branch & Regret algorithms will be presented relying on a DES mechanism in order to manage the incoming events. In this regard, the first two proposed approaches are referred to as 'myopic' because they consider only the immediate next decision, without accounting for subsequent ones. In contrast, the other two algorithms, the 'foresight' ones, also account for future decisions. Each time a vehicle concludes a customer's request, an optimisation step is triggered and therefore a decision is taken for each vehicle (if a vehicle is in transit, its decision is disregarded). The decision consists in serving a specific known customer or, alternatively, waiting $w = 1$ unit of time at its current location. The former is applied if the same known customer is served in each scenario while the latter is executed when the vehicle does not serve the same customer in all scenarios. A feasible solution is progressively constructed, step by step. This solution $(v_1, \ldots, v_m)$ is referred to as *partial solution*. For each vehicle r, the corresponding route v_r includes the source node and a sequence of served customers. All algorithms continue to perform optimisation steps until each vehicle reaches the destination node.

It may occur that no further action can be taken because some mandatory nodes can no longer be included in any route due to the previous decisions taken. Therefore, it is important to note that the constraints for mandatory nodes are not treated as hard constraints (which could frequently result in infeasible solutions); instead, they are enforced through a significant penalty in the objective function, providing greater flexibility for the algorithms.

7.3.1 Myopic approaches

This section introduces two myopic algorithms for the Branch & Regret approach. The myopic methods determine the next decision based solely on the current *partial solution* and the information presently available. The first algorithm, the `Myopic Algorithm 1` (MA1), adopts an Integer Linear Program (ILP) to choose the next vehicles' decisions. In contrast, the second one, the `Myopic Algorithm 2` (MA2), employs a customised Branch & Bound procedure to do the same.

The Myopic Algorithm 1

The optimisation step of the MA1 consists in solving the following ILP:

$$\max \quad \sum_{s\in S}\sum_{r\in R}\sum_{k\in\hat{N}\cup\{n\}} \nu_s\, \mathcal{Q}(p^s_{kr})\, y_{srk} \tag{7.1a}$$

$$\text{s.t.} \quad \sum_{r\in R} y_{srk} \leq 1, \quad \forall\, s\in S,\, k\in\hat{N}, \tag{7.1b}$$

$$\sum_{k\in\hat{N}\cup\{n\}} y_{srk} \leq 1, \quad \forall\, s\in S,\, r\in R, \tag{7.1c}$$

$$\sum_{s\in S\setminus\{1\}} y_{srk} = (|S|-1)\, y_{1rk}, \quad \forall\, r\in R,\, k\in\hat{N} : g_k = 1, \tag{7.1d}$$

$$y_{srk} \in \{0,1\}, \quad \forall\, s\in S,\, r\in R,\, k\in\hat{N}. \tag{7.1e}$$

The proposed model makes use of the following decision variables: the variable y_{skr} is equal to 1 if and only if the node k is visited by the route r in the scenario s, 0 otherwise. $\mathcal{Q}(p^s_{kr})$ represents the evaluation of the recourse function on the *partial solution* p^s_{kr} as described in Section 6.2.3. The *partial solution* p^s_{kr} is built from the current *partial solution* p after that the node k is visited at the end of the sequence of the nodes visited by the vehicle r in the scenario s. It is important to remark that $\mathcal{Q}(p^s_{kr})$ is set to $-\infty$ if serving customer k with vehicle r in the scenario s is not permitted by the incompatibilities or by the chance constraints, or if the customer k does not belong to the scenario s.

The objective function (7.1a) maximises the weighted expected profit of the *partial solution*. Constraints (7.1b) guarantee that a customer can only be visited by at most one vehicle in a specific scenario. Constraints (7.1c) guarantee that a vehicle visits at most one customer in a specific scenario. Constraints (7.1d) impose the non-anticipativity constraints for each vehicle and for the *known customers*. In particular, if a known customer is chosen for a vehicle in a scenario, it implies that the same customer will have to be chosen in all the other scenarios (7.1d).

The Myopic Algorithm 2

The optimisation step of the MA2 consists in a customised branching procedure. The MA2 is a Branch & Bound algorithm that leverages a branching tree in order to take a decision for each vehicle. Each node in this tree represent a *partial solution*, where scenarios may not be aligned with a common decision for each vehicle. When such a case occurs, the current branching node does not correspond to a decision because certain non-anticipativity constraints are violated. Therefore, a branching phase is performed in which the child nodes are created, solved and added to a frontier of open tree nodes. This process continues until the first node representing a decision for each vehicle is found. An upper bound, derived from a relaxation that permits the violation of certain non-anticipativity constraints, guides the selection of the next branching node from the frontier.

Solving a branching node Solving a branching node involves finding an assignment of vehicles-to-customers (still unassigned) for each scenario in order to maximise the expected profit of the *partial solution*. This can be modelled as an assignment problem that is solved in polynomial time by the Hungarian algorithm [66]. The cost matrix θ^s_{kr} for the Hungarian algorithm is built by computing $\mathcal{Q}(p^s_{kr})$:

$$\theta^s_{kr} = \frac{1}{\nu_s \, \mathcal{Q}(p^s_{kr})},$$

where, $\mathcal{Q}(p^s_{kr})$ represents the expected profit of the *partial solution* p^s_{kr}. It is important to remark that the assignment cost is inversely proportional to the weighted expected cost because the Hungarian algorithm is designed to solve minimisation problems. It is also worth noting that the value of θ^s_{kr} is set to $+\infty$ if serving customer k with vehicle r in the scenario s is not permitted by the incompatibilities or by the chance constraints, or if the customer k does not belong to the scenario s. If such a feasible assignment can be computed, a branching node is created and associated to an upper bound for the expected profit equal to:

$$\sum_{s \in S} \sum_{r \in R} \sum_{k \in \hat{N} \cup \{n\}} \nu_s \, \mathcal{Q}(p^s_{kr}) \, \mathrm{H}^s_{kr}(p),$$

where, $\mathrm{H}^s_{kr}(p)$ represents the application of the assignment made by the Hungarian algorithm on the current *partial solution* p ($\mathrm{H}^s_{kr}(p)$ is equal to 1 if the customer k is assigned at the end of the sequence of the customers served by the vehicle r in the scenario s, 0 otherwise). The quantity reported above corresponds to the value of the objective function after applying the assignment determined by the Hungarian algorithm. Since this assignment may be different across all the scenarios (leading to solutions that violate one or more non-anticipativity constraints), its solution is considered a relaxation for the current optimisation step.

This branching node is then added to the frontier of open tree nodes which remains sorted based on the value of the upper bound. When a node is removed from the head of the frontier (the one associated with the maximal upper bound value), it is verified whether it represents a decision or not. This check consists in creating a consensus matrix ζ_{kr}, which counts the number of scenarios in which vehicle r serves the customer k as a next stop from its current location l_r. In this regard, the following quantity is introduced in order to simplify the notation:

$$\eta_r = \max_{k \in \hat{N} \colon g_k = 1} \zeta_{kr}.$$

The matrix ζ_{kr} acts as a consensus function in order to determine if the current branching node represents a decision for each vehicle. If it does not, ζ_{kr} is used to identify a specific (vehicle, customer) pair on which to perform the branching.

Branching phase If $|S| > \eta_r > 0$ for a vehicle r and some known customer k ($g_k = 1$), a binary branch is performed on the pair

$$(\bar{r}, \bar{k}) = \underset{k \in \hat{N}: g_k = 1}{\arg\max}\, \zeta_{kr}.$$

Figure 7.1 graphically depicts this binary branch. In the figure shown, P represents a generic branching node that does not correspond to a decision, necessitating a branch on it. P_1 and P_2, on the other hand, are the child nodes of P. These nodes will be solved sequentially and then added to the frontier. The node P_1 is the result of the high-level branching rule b_1 that negates the connection $(l_{\bar{r}}, \bar{k})$ and it is ensured imposing $\theta^s_{\bar{k}\bar{r}} = +\infty$ for each scenario $s \in S$. On the contrary, the node P_2 is the result of the high-level branching rule b_2 which makes the same connection mandatory and it is ensured imposing $\theta^s_{\bar{r}a} = +\infty$ for each $a \in \hat{N} : a \neq \bar{k}$ for each scenario $s \in S$.

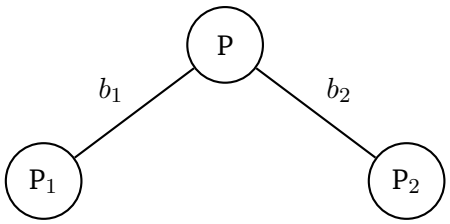


Fig. 7.1: A graphical representation of the binary branch.

Taking a decision If η_r is equal to $|S|$ for a specific vehicle r and known customer k, all scenarios align on the common decision that consists in moving r toward k. Otherwise, if η_r is equal to 0 for each known customer k and vehicle r, all scenarios align on a common decision for r that consists in making r wait for w unit of time in its current location l_r. If this happens for all vehicles, the current branching node corresponds to a decision.

7.3.2 Foresight approaches

Two foresight approaches of Branch & Regret will be proposed in this section. The first approach, the `Foresight Algorithm 1` (FA1), employs a customised Mixed Integer Linear Program (MILP) to determine the next decision also considering potential future decisions. In contrast, the second approach, the `Foresight Algorithm 2` (FA2), employs an ad-hoc heuristic Branch & Price procedure.

The Foresight Algorithm 1

The optimisation step of the FA1 consists in solving the following MILP that is an extension of the formulation introduced in Section 6.2.2):

$$\max \quad \sum_{s \in S} \nu_s \sum_{r \in R} \mathcal{Q}(x_{sr}) \tag{7.2a}$$

$$\text{s.t.} \quad \sum_{(1,j) \in \hat{A}} x_{sr1j} = \sum_{(i,n) \in \hat{A}} x_{srin} = m, \quad \forall\, r \in R,\ s \in S, \tag{7.2b}$$

$$\sum_{(i,k) \in \hat{A}} x_{srik} = \sum_{(k,j) \in \hat{A}} x_{srkj} = y_{srk}, \quad \forall\, k \in \hat{N},\ r \in R,\ s \in S, \tag{7.2c}$$

$$\sum_{(k,j) \in \hat{A}} z_{srkj} - \sum_{(i,k) \in \hat{A}} z_{srik} = \sum_{(k,j) \in \hat{A}} \alpha_{rkj}\, x_{srkj}, \quad \forall\, k \in \hat{N},\ r \in R,\ s \in S \tag{7.2d}$$

$$\sum_{r \in R} y_{srk} \leq 1, \quad \forall\, k \in \hat{N},\ s \in S, \tag{7.2e}$$

$$\sum_{r \in R} x_{srij} = 0, \quad \forall\, (i,j) \in I,\ s \in S, \tag{7.2f}$$

$$|C_k|(1 - y_{srk}) \geq \sum_{i \in C_k} y_{sri}, \quad \forall\, k \in \hat{N},\ r \in R,\ s \in S, \tag{7.2g}$$

$$z_{srij} \leq \eta_{rij}\, x_{srij}, \quad \forall\, (i,j) \in \hat{A},\ r \in R,\ s \in S, \tag{7.2h}$$

$$z_{srij} \geq \gamma_{rij}\, x_{srij}, \quad \forall\, (i,j) \in \hat{A}, r \in R,\ s \in S, \tag{7.2i}$$

$$z_{sr1k} = \beta_{ri}\, x_{sr1k}, \quad \forall\, (i,j) \in \hat{A}, r \in R,\ s \in S, \tag{7.2j}$$

$$x_{srij} \in \{0,1\}, \quad \forall\, (i,j) \in \hat{A} \setminus \{(1,n)\},\ r \in R,\ s \in S, \tag{7.2k}$$

$$y_{srk} \in \{0,1\}, \quad \forall\, k \in \hat{N},\ r \in R,\ s \in S, \tag{7.2l}$$

$$x_{sr1n} \in \mathbb{N}^+, \quad \forall\, r \in R,\ s \in S, \tag{7.2m}$$

$$z_{srij} \in \mathbb{R}^+, \quad \forall\, (i,j) \in \hat{A},\ r \in R,\ s \in S, \tag{7.2n}$$

The formulation makes use of the following decision variables: the variable x_{srij} is equal to 1 if and only if the arc (i,j) is traversed by the route r in the scenario s, 0 otherwise; the variable y_{srk} is equal to 1 if and only if the node k is visited by the route r in the scenario s, 0 otherwise; the variable z_{srij} defines the expected arrival time at node j coming from the node i and can be thought as the amount of flow that passes through the arc (i,j) inside the route r in the scenario s.

The objective function (7.2a) maximises the weighted expected profit collected by the routes (x_{sr} represents the route r of the solution x in the scenario s). Constraints (7.2b) guarantee that each vehicle starts its route from the source node and ends to the destination one in each scenario. Constraints (7.2c) impose the connectivity of each route in each scenario. Constraints (7.2e) guarantee that a node can only be visited by at most one route in each scenario. Constraints (7.2f) and (7.2g) respectively guarantee that all the physical and logical incompatibilities are satisfied in each scenario. The (7.2d) and (7.2h) are the classical Gavish-Graves (GG) subtour elimination constraints adapted for the TOP [43]. Constraints (7.2i) set the lower bound on the *route expectation* for each route in each scenario. Constraints (7.2j)

bound the flow originating by each route from the initial depot in each scenario. Finally, the (7.2k), (7.2l), (7.2m) and (7.2n) are variable definition constraints.

The variables x_{srij} are fixed according to the current *partial solution*. The proposed model is then solved with an adaptation of the 3ILA presented in Section 6.5.

The quantities introduced in the formulation are defined as:

- $\alpha_{rij} = (\mu_i + \mu_{ij} + \tau_{ir} + (1 - g_i)\, w)$.
- $\eta_{rij} = (T_{\max} - \mu_j - \mu_{jn} - \tau_{ir} - (1 - g_j)\, w)$.
- $\gamma_{rij} = (\mu_{1i} + \mu_i + \mu_{ij} + \tau_{ir} + (1 - g_j)\, w)$.
- $\beta_{ri} = (\mu_{1i} + \tau_{1r})$.

In these definitions, μ_k and μ_{ij} are the *route expectation* of the Normal random variables ξ_k and ξ_{ij}, respectively (ξ_k and ξ_{ij} represent the random variables associated with the service and travel times). The quantity τ_{kr} represents instead the accumulated waiting time on the node k for the vehicle r. In addition to this formulation, a further set of constraints is added in order to respect the current *partial solution* computed during the past optimisation steps.

The Foresight Algorithm 2

The optimisation step of FA1 is computationally prohibitive even for the small benchmark set. To address this limitation, the FA2 improves the computational efficiency of FA1. To do that, the optimisation of the FA2 employs a specialised heuristic branch-and-price algorithm with a customised branching procedure. Similar to the MA2, a branching tree is built to determine a decision for each vehicle.

Solving a branching node In order to solve a branching node, a column generation procedure has been implemented. The Master Problem (MP) is defined as a linear relaxation of the Set Packing Problem (SSP) as follows:

$$\max \quad \sum_{s\in S}\sum_{r\in R}\sum_{\omega\in\Omega_{sr}} \nu_s\, \mathcal{Q}(\omega)\, x_\omega \tag{7.3a}$$

$$\text{s.t.} \quad \sum_{\omega\in\Omega sr} a_{k\omega}x_\omega = 1, \quad \forall\, k \in F_r,\ r \in R,\ s \in S, \tag{7.3b}$$

$$\sum_{r\in R}\sum_{\omega\in\Omega sr} a_{k\omega}x_\omega + v_{ks} - f_{ks} = 1, \quad \forall\, k \in M,\ s \in S, \tag{7.3c}$$

$$\sum_{r\in R}\sum_{\omega\in\Omega_{sr}} a_{k\omega}x_\omega - b_{ks} \le 1, \quad \forall\, k \in \hat{N} \setminus M,\ s \in S, \tag{7.3d}$$

$$\sum_{\omega\in\Omega_{sr}} x_\omega \le 1, \quad \forall r \in R,\ s \in S, \tag{7.3e}$$

$$v_{ks}, f_{ks} \ge 0 \quad \forall\, k \in M,\ s \in S, \tag{7.3f}$$

$$b_{ks} \ge 0 \quad \forall\, k \in \hat{N} \setminus M,\ s \in S, \tag{7.3g}$$

$$x_\omega \geq 0 \quad \forall\, \omega \in \Omega_{sr},\ s \in S,\ r \in R. \tag{7.3h}$$

The proposed model makes use of the following decision variables: the variable x_ω is equal to 1 if and only if the route ω is chosen, 0 otherwise (Ω_{sr} is the set of routes belonging to the scenario s of the vehicle r). The objective function (7.3a) maximises the expected profit of the selected routes ($\mathcal{Q}(\omega)$ represents the recourse function evaluated for the route ω as reported in Section 6.2.3). Constraints (7.3b) guarantee that each customer visited in the current *partial solution* is served (the set F_r represents the set of customers already served by the vehicle r in the current *partial solution*). Constraints (7.3c) guarantee that each mandatory customer is served (if possible). In this case, the variables v_{ks} and f_{ks} act as slack and surplus variables. Constraints (7.3d) ensure that each optional customer is served by no more than one vehicle. In this case, the variables b_{ks} act as slack variables. Constraints (7.3e) ensure at most a single route is selected for each vehicle r in the scenario s. Finally, the (7.3f), (7.3g), and (7.3h) are variable definition constraints.

As evident from the formulation, the set of variables x_ω are partitioned by scenario and vehicle. This proved to be necessary because, during the optimisation steps, the vehicles' current locations differ (except for the first optimisation step, where each vehicle starts from the source node).

The model presented above (7.3) consists in a Linear Problem (LP) which can be dualised. Therefore, the dual objective function is the following:

$$\min \sum_{s\in S}\sum_{r\in R}\sum_{k\in F_r} \pi_{ks} + \sum_{s\in S}\sum_{k\in M} \delta_{ks} + \sum_{s\in S}\sum_{k\in \hat{N}\setminus M} \gamma_{ks} + m \sum_{s\in S} \beta_{sr}.$$

This function must respect the set of dual constraints:

$$\sum_{r\in R}\sum_{k\in F_r} a_{k\omega}\, \pi_{ks} + \sum_{k\in M} a_{k\omega}\, \delta_{ks} + \sum_{k\in \hat{N}\setminus M} a_{k\omega}\, \gamma_{ks} + \beta_{sr} \geq \nu_s\, \mathcal{Q}(\omega), \tag{7.4}$$

for each $\omega \in \Omega_{sr},\ s \in S,\ r \in R$. The variables $\pi_{ks} \in \mathbb{R}$, $\delta_{ks} \in \mathbb{R}$, $\gamma_{ks} \geq 0$ and $\beta_{sr} \geq 0$ represent the dual variables associated to the constraints (7.3b), (7.3c), (7.3d) and (7.3e), respectively.

Taking into account the constraints (7.4), the Pricing Problem (PP) for the MP (7.3) consists in finding the route ω such that:

$$\max \sum_{s\in S} \nu_s\, \mathcal{Q}(\omega) - \sum_{r\in R}\sum_{k\in F_r} a_{k\omega}\, \pi_{ks} - \sum_{k\in M} a_{k\omega}\, \delta_{ks} - \sum_{k\in \hat{N}\setminus M} a_{k\omega}\, \gamma_{ks} - \sum_{r\in R} \beta_{sr}.$$

The routes generated by the PP must respect all the incompatibilities presented in Section 5.3 and a set of chance constraints as introduced in Section 6.2.2. On the other hand, the PP corresponds to the Stochastic Orienteering Problem (SOP), which has been shown to be at least NP-hard since its deterministic counterpart is [45].

Finally, considering that the set of routes is partitioned by scenario s and vehicle r, the PP can be decomposed into a set of pricing problems PP_r^s for each scenario s and vehicle r (because the set of routes belonging to a partition are independent with respect to the set of routes belonging to another one):

$$\max_{\omega \in \Omega_{sr}} \nu_s \, \mathcal{Q}(\omega) - \sum_{r \in R} \sum_{k \in F_r} a_{k\omega} \, \pi_{ks} - \sum_{k \in M} a_{k\omega} \, \delta_{ks} - \sum_{k \in \hat{N} \setminus M} a_{k\omega} \, \gamma_{ks} - \beta_{sr}.$$

Each PP_r^s is heuristically solved by using an adaptation of the VND described in Section 5.6.2, which has been extended to handle chance constraints. Since the routes of set Ω_{sr} are independent from each other, all the PP_r^s are solved in parallel by using a thread pool methodology.

Once the column generation procedure is completed, a branching node is then created and added to a priority queue, which is kept sorted based on the objective function value. When a node is removed from the head of the priority queue, it is checked to determine whether it represents a decision or not. This check involves constructing the consensus matrix, ζ_{rk} (similar to MA2), which counts the number of routes in which the vehicle r has served the customer k as a next stop from its current location l_r (if k corresponds to an unassigned customer request). In this regard, the following quantity is introduced in order to simplify the notation:

$$\eta_r = \max_{k \in \hat{N}:\, g_k = 1} \zeta_{kr}.$$

Finally, it is worth noting that, unlike in MA2, ζ_{rk} can even take non-integer values since a route can be selected a fractional number of times (since the MP is a LP).

Branching phase If $|S| > \eta_r > 0$ for a vehicle r and some known customer k, a binary branch is performed on the pair

$$(\bar{r}, \bar{k}) = \arg\max_{k \in \hat{N}: g_k = 1} \zeta_{kr}.$$

The same branching logic applied for the MA2 is also adopted in this case (see Figure 7.1). Then, two different branching nodes P_1 and P_2 are created and solved from the parent node P (they will be added to the priority queue in case they represent a feasible *partial solution*). On the node P_1, the high-level branching rule b_1 imposes that the connection $(l_{\bar{r}}, \bar{k})$ is forbidden. On the contrary, on the node P_2, the high-level branching rule b_2 imposes that the same connection as mandatory. Both branching rules are guaranteed by adding additional constraints (which forbid some connections on the graph) to the heuristic approach adopted to solve the pricing problems.

Taking a decision If η_r is equal to $|S|$ for a specific vehicle r and known customer k ($g_k = 1$), all scenarios align on the common decision that consists in moving r toward k. Otherwise, if η_r is equal to 0 for each known customer k and vehicle r, all scenarios align on a common decision for r that consists in making r wait for w unit of time in its current location l_r. If this happens for all vehicles, the current branching node corresponds to a decision.

The pseudocode that summarises the optimisation step of the MA1 and FA1 is reported in Algorithm 10 that makes use of the following procedures:

(i) `solveModel`: solves the corresponding formulation related to the MA1 or FA1 and (ii) `applyDecision`: apply the decision (expressed as a set of actions $\{a^1, \ldots, a^m\}$) to the current *partial solution* p.

Algorithm 10: The MA1 and FA1 optimisation step.

Data: Partial solution p
1 $\{a^1, \ldots, a^m\} \leftarrow$ `solveModel` (p);
2 $p \leftarrow$ `applyDecision` $(p, \{a^1, \ldots, a^m\})$;
3 **return** p;

The pseudocode that summarises the optimisation step of the MA2 and FA2 is reported in Algorithm 11 that makes use of the following procedures:

Algorithm 11: The MA2 and FA2 optimisation step.

Data: Partial solution p
1 $f \leftarrow \{$ `createBranchingNode` (p) $\}$;
2 **while** $f \neq \emptyset$ **do**
3 $\phi \leftarrow$ `extractNode` (f);
4 $\zeta \leftarrow$ `createConsensusMatrix` (ν);
5 $\varphi \leftarrow$ true;
6 **for** $r \in R$ **do**
7 $\eta_r \leftarrow \max_{k \in \hat{N}:\, g_k = 1} \zeta_{kr}$;
8 **if** $|S| > \eta_r > 0$ **then** `/* ` ϕ `is not a decision */`
9 $\phi_1, \phi_2 \leftarrow$ `branchNode` (ϕ, ζ_{kr});
10 `solveNode` (ϕ_1), `solveNode` (ϕ_2);
11 `insertNode` (ϕ_1, f), `insertNode` (ϕ_2, f);
12 $\varphi \leftarrow$ false;
13 **if** φ is true **then** `/* ` ϕ `is a decision */`
14 $\{a^1, \ldots, a^m\} \leftarrow$ `getActions` (ϕ);
15 $p \leftarrow$ `applyDecision` $(p, \{a^1, \ldots, a^m\})$;
16 **return** p;

(i) `createBranchingNode`: creates a branching node that wraps a specific *partial solution* p, (ii) `createConsensusMatrix`: creates the consensus matrix from a specified branching node, (iii) `extractNode`: extracts the node from the frontier f associated

with the maximal upper bound, (iv) `insertNode`: inserts a branching node to the frontier f, (v) `branchNode`: create a pair of branching child nodes accordingly to the additional branching rules specific to each child, (vi) `solveNode`: solve a specific branching node, (vii) `getActions`: gets the set of actions (one for each vehicle) from a specified branching node, and (viii) `applyDecision`: apply the decision (expressed as a set of actions $\{a^1, \ldots, a^m\}$) to the current *partial solution* p.

It is worth noting that all the algorithms presented always ensure to apply a decision, since each vehicle can always take the trivial action of moving toward the destination node.

7.4 Quantitative analysis

This section provides a detailed quantitative analysis in order to validate the performance of the proposed Branch & Regret algorithms on the adapted TOP-ST-MIN instances (Sect. 7.2). The analysis examines solution quality, computational time and the scalability of the approaches. It is important to note that only routes with a *route reliability* (Sect. 6.2.1) greater or equals to 0.75 are considered in order to ensure a broader exploration of the search space.

All the algorithms presented have been implemented in C++ (CPLEX 22.1.1 Concert Technology is adopted for implementing the column generation procedure for the FA2) and run in single-threaded mode (except for the FA2, where the pricing problem was parallelised using up to all available threads of the machine). The code was compiled in Ubuntu 22.04. The experiments were carried out on a 64-bit Linux machine, with an Intel(R) Core(TM) i7-8700, 3.20GHz and 16 GB of RAM. CPLEX built-in cuts have been used inside the column generation procedure for the FA2. Finally, the solver is run in a fully-deterministic mode in order to be able to reproduce the results (if the same platform is used).

The rows structure of the following tables is consistent with that presented in Section 5.5.3. The results are structured into different solution categories in order to facilitate the interpretation and comparison. They are first grouped by benchmark set and then by feature generation methods. Specifically, each benchmark set is subdivided based on feature generation methods since, for a given feature, all related methods described in Chapter 4 create a distinct partition of the benchmark sets.

Table 7.1 reports the results of the MA1 for the DSTOP-ST-MIN-P. The columns Θ_R and Θ_U represent the average expected reward collected by the vehicles and the average number of unvisited mandatory nodes, respectively. The column STEPS indicates the number of optimisation steps that the algorithm made. Finally, the columns TPS(s) and CPU(s) report the average time required to complete an optimisation

Tab. 7.1: The MA1 results for the DSTOP-ST-MIN-P.

		Θ_R	Θ_U	STEPS	TPS(s)	CPU(s)
SMALL	**ALL**	**145.12**	**0.88**	**18**	**0.01**	**0.05**
	CMN	154.00	0.53	17	0.01	0.05
	SMN	136.23	1.23	18	0.01	0.06
	CPI	148.99	0.86	17	0.01	0.05
	DPI	141.24	0.90	18	0.01	0.05
MEDPIUM	**ALL**	**237.77**	**1.49**	**25**	**0.01**	**0.40**
	CMN	264.89	0.98	26	0.01	0.45
	SMN	210.66	2.00	24	0.01	0.36
	CPI	242.88	1.13	25	0.01	0.41
	DPI	232.67	1.85	25	0.01	0.40
LARGE	**ALL**	**253.94**	**1.98**	**25**	**0.01**	**0.40**
	CMN	276.40	1.10	29	0.01	0.47
	SMN	231.48	2.85	22	0.01	0.32
	CPI	252.50	1.87	25	0.01	0.41
	DPI	255.39	2.08	25	0.01	0.39

step and the average total execution time of the approach, measured in seconds, respectively.

Table 7.2 reports the computational results for the MA2 and follows a structure similar to the previous table with the exception of the NODES column which represents the average number of branching nodes generated to determine a decision to apply for each vehicle.

Tab. 7.2: The MA2 results for the DSTOP-ST-MIN-P.

		Θ_R	Θ_U	STEPS	TPS(s)	NODES	CPU(s)
SMALL	**ALL**	**144.91**	**0.92**	**17**	**0.01**	**4**	**0.01**
	CMN	152.33	0.56	17	0.01	3	0.01
	SMN	137.50	1.28	18	0.01	4	0.01
	CPI	146.35	0.90	17	0.01	5	0.01
	DPI	143.48	0.95	17	0.01	3	0.01
MEDPIUM	**ALL**	**229.97**	**1.68**	**24**	**0.01**	**4**	**0.19**
	CMN	254.34	1.10	23	0.01	5	0.23
	SMN	205.61	2.25	24	0.01	3	0.14
	CPI	236.90	1.45	24	0.01	4	0.19
	DPI	223.04	1.90	23	0.01	3	0.18
LARGE	**ALL**	**255.68**	**2.01**	**25**	**0.01**	**3**	**0.17**
	CMN	269.99	1.27	29	0.01	2	0.18
	SMN	241.37	2.75	22	0.01	3	0.15
	CPI	261.73	2.05	26	0.01	3	0.17
	DPI	249.63	1.97	25	0.01	3	0.16

Finally, Table 7.3 presents the results for the FA2 and follows a structure similar to Table 7.2, except for the COLS and CGT(s) columns. These columns represent, respectively, the average number of variables (columns) generated and the average time (in seconds) spent by the algorithm to complete the column generation procedure each optimisation step.

Tab. 7.3: The FA2 results for the DSTOP-ST-MIN-P.

		Θ_R	Θ_U	STEPS	TPS(s)	NODES	COLS	CGT(s)	CPU(s)
SMALL	**ALL**	**146.83**	**0.96**	**20**	**1.82**	**6**	**139**	**2.22**	**46.61**
	CMN	152.95	0.60	20	2.12	6	156	2.50	53.57
	SMN	140.71	1.31	20	1.52	6	121	1.94	39.66
	CPI	142.36	0.99	20	1.66	6	143	2.43	52.34
	DPI	151.29	0.92	20	2.20	6	134	2.01	40.88
MEDPIUM	**ALL**	**351.25**	**0.94**	**31**	**8.44**	**6**	**149**	**11.89**	**432.11**
	CMN	346.34	0.57	32	7.77	6	159	12.15	435.30
	SMN	356.15	1.32	30	9.12	6	140	11.63	428.91
	CPI	358.25	1.03	31	9.86	6	158	13.23	485.09
	DPI	344.24	0.85	31	8.01	6	141	10.54	379.12
LARGE	**ALL**	**292.34**	**3.32**	**44**	**38.31**	**10**	**351**	**74.76**	**3351.36**
	CMN	304.40	2.08	41	53.86	9	440	92.17	3746.63
	SMN	280.28	4.55	48	22.75	10	262	57.35	2956.10
	CPI	294.70	3.25	45	43.87	10	380	87.13	3920.76
	DPI	289.97	3.38	44	40.33	9	322	62.39	2781.97

A comparison between the MA1 and MA2 algorithms (Tables 7.1 and 7.2) indicates that the MA1 slightly outperforms the MA2 in terms of average expected reward collected (except for the large benchmark set), as shown in the Θ_R column. While, considering the number of optimisation steps, the average time per step TPS(s) and the total running time CPU(s), both algorithms demonstrate comparable performance with no significant differences.

In contrast, the results for the FA2 follow a different pattern. As observed from Tables 7.1, 7.2, and 7.3, the FA2 consistently achieves a higher average expected reward, particularly in the medium and large benchmark sets. However, regarding the average number of unvisited mandatory nodes, the MA1 generally performs better on the small and large benchmark sets, whereas the FA2 shows better results on the medium benchmark set. When comparing the average number of branching nodes NODES between the MA2 and the FA2, the latter generally requires slightly more nodes to reach a decision than the MA2. Additionally, both the average number of generated columns COLS and the column generation time CGT(s) increase as the instance size grows. Finally, both the average time per step TPS(s) and the total running time CPU(s) of the FA2 are consistently higher than those of the MA1 and MA2 and the differences increase with the size of the instances.

Figure 7.2 provides a graphical summary of the results (for the DSTOP-ST-MIN-P) presented in the previous tables subdivided by instances size and number of available vehicles (m). An observation from the figure (which confirms the analysis from the previous tables) is that the FA2 dominates the other two approaches with respect to the expected reward collected (especially for the medium and large benchmark sets). In fact, the blue circles, representing the FA2 points, are concentrated in the upper right corner of each sub-figure (the section of the figure corresponding to a high expected profit and a low number of unvisited mandatory customers).

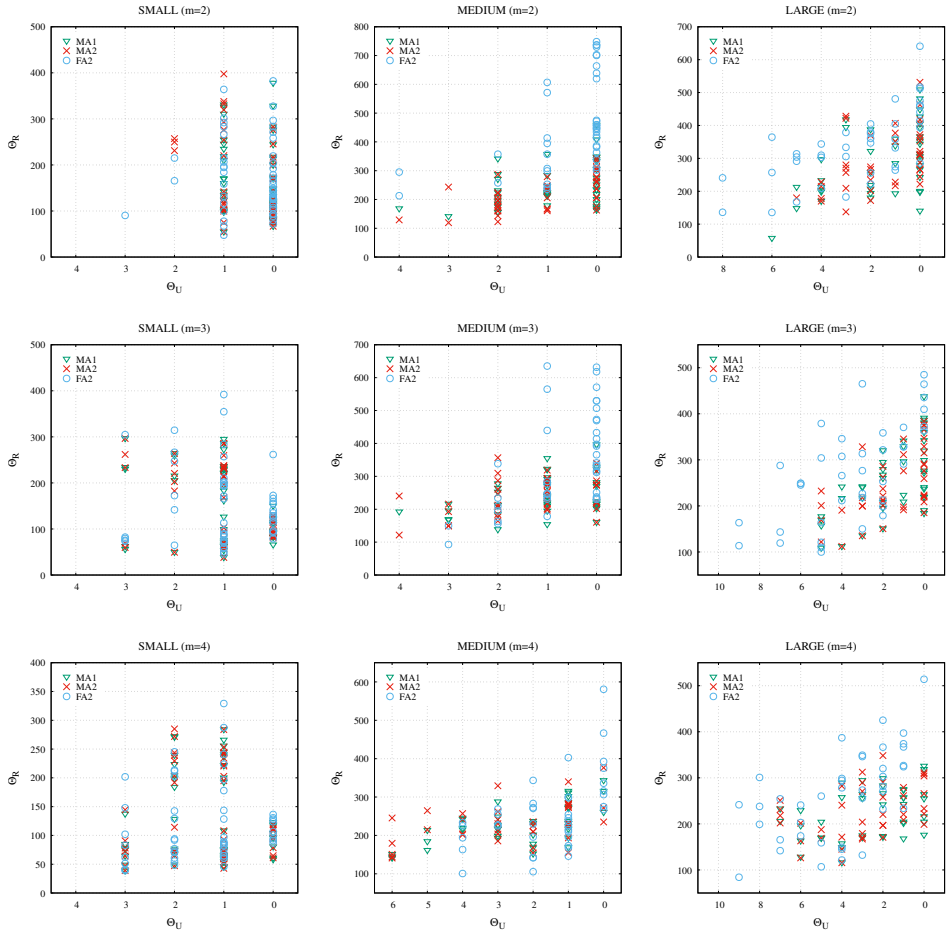


Fig. 7.2: A summary of the results of the Branch & Regret algorithms for the DSTOP-ST-MIN-P.

Examining the tables in detail reveals a clear trend. Instances with a Scattered Mandatory (SM) node distribution tend to be slightly more challenging than others in terms of average expected reward collected. A similar pattern is observed for the unvisited mandatory nodes, but only for the large benchmark set, where the average number of unvisited mandatory customers exceeds one. This behaviour can be attributed to the scattered mandatory node distribution which increases the distances travelled by the vehicles. Consequently, each vehicle serves fewer customers, which directly reduces the total reward collected and increases the likelihood that some mandatory customers remain unvisited. In contrast, the instances characterised by the Clustered Mandatory (CM) nodes tend to be slightly easier to solve in terms of average expected reward collected and unvisited mandatory nodes. Finally, with regard to the physical incompatibilities generation methods, there are no relevant differences to highlight.

Tables 7.4, 7.5 and 7.6 present the results for the same algorithms but for the DSTOP-ST-MIN-PL. The structure of the tables remain the same.

Tab. 7.4: The MA1 results for the DSTOP-ST-MIN-PL.

		Θ_R	Θ_U	Steps	TPS	CPU(s)
SMALL	**ALL**	**146.25**	**0.92**	**17**	**0.01**	**0.05**
	CMN	155.29	0.55	17	0.01	0.05
	SMN	137.22	1.29	18	0.01	0.05
	CPI	150.89	0.92	17	0.01	0.05
	DPI	141.61	0.93	17	0.01	0.05
	FLI	145.03	0.88	18	0.01	0.05
	NLI	147.48	0.97	17	0.01	0.05
MEDPIUM	**ALL**	**258.62**	**1.32**	**20**	**0.01**	**0.17**
	CMN	271.34	1.07	21	0.01	0.18
	SMN	245.90	1.57	20	0.01	0.16
	CPI	270.57	1.38	21	0.01	0.17
	DPI	246.67	1.25	20	0.01	0.17
	FLI	266.14	1.23	21	0.01	0.17
	NLI	251.11	1.40	20	0.01	0.17
LARGE	**ALL**	**223.92**	**2.48**	**29**	**0.01**	**0.62**
	CMN	262.12	1.53	32	0.01	0.73
	SMN	185.71	3.43	25	0.01	0.51
	CPI	221.65	2.03	29	0.01	0.63
	DPI	226.19	2.93	29	0.01	0.61
	FLI	223.46	2.34	29	0.01	0.62
	NLI	224.38	2.61	28	0.01	0.62

A comparison between the MA1 and MA2 algorithms (Tables 7.4 and 7.5) reveals no clear dominance of one approach over the other, as observed in the DSTOP-ST-MIN-P case. Specifically, the MA1 slightly outperforms the MA2 in terms of average

Tab. 7.5: The MA2 results for the DSTOP-ST-MIN-PL.

		Θ_R	Θ_U	Steps	TPS	Nodes	CPU(s)
SMALL	**ALL**	**145.62**	**0.95**	**17**	**0.01**	**4**	**0.01**
	CMN	154.24	0.58	16	0.01	3	0.01
	SMN	137.00	1.32	17	0.01	4	0.01
	CPI	148.78	0.94	17	0.01	4	0.01
	DPI	142.45	0.97	17	0.01	3	0.01
	FLI	144.84	0.92	17	0.01	4	0.01
	NLI	146.40	0.98	17	0.01	3	0.01
MEDPIUM	**ALL**	**256.83**	**1.43**	**19**	**0.01**	**3**	**0.06**
	CMN	260.43	1.18	19	0.01	3	0.06
	SMN	253.24	1.68	20	0.01	3	0.05
	CPI	268.43	1.52	19	0.01	3	0.06
	DPI	245.24	1.34	19	0.01	3	0.05
	FLI	263.02	1.34	20	0.01	3	0.06
	NLI	250.65	1.52	19	0.01	3	0.06
LARGE	**ALL**	**224.56**	**2.54**	**29**	**0.01**	**4**	**0.30**
	CMN	261.33	1.62	31	0.01	5	0.37
	SMN	187.79	3.46	26	0.01	3	0.24
	CPI	224.12	2.19	28	0.01	4	0.31
	DPI	225.00	2.88	29	0.01	3	0.30
	FLI	224.01	2.43	29	0.01	4	0.30
	NLI	225.10	2.65	28	0.01	4	0.31

Tab. 7.6: The FA2 results for the DSTOP-ST-MIN-PL.

		Θ_R	Θ_U	Steps	TPS	Nodes	Cols	CGT	CPU(s)
SMALL	**ALL**	**139.86**	**1.04**	**22**	**1.50**	**6**	**127**	**2.16**	**44.43**
	CMN	146.62	0.65	22	1.74	6	140	2.43	50.59
	SMN	133.10	1.43	21	1.26	6	114	1.89	38.28
	CPI	135.38	1.08	22	1.38	6	128	2.27	46.99
	DPI	144.34	1.00	22	1.63	6	125	2.05	41.87
	FLI	145.87	0.99	24	1.35	6	138	2.62	56.93
	NLI	133.85	1.10	20	1.66	6	115	1.70	31.93
MEDPIUM	**ALL**	**298.76**	**1.19**	**50**	**4.35**	**7**	**149**	**12.00**	**394.27**
	CMN	296.64	0.78	52	3.55	6	160	12.75	418.32
	SMN	300.88	1.59	48	5.16	7	138	11.26	370.21
	CPI	308.15	1.23	53	5.53	7	157	13.31	437.96
	DPI	289.37	1.15	48	3.18	6	141	10.69	350.58
	FLI	336.19	1.03	52	5.07	6	144	11.97	416.59
	NLI	261.34	1.34	49	3.64	7	153	12.03	371.95
LARGE	**ALL**	**281.09**	**3.49**	**143**	**32.15**	**9**	**315**	**74.75**	**3386.76**
	CMN	303.83	2.65	166	56.94	9	395	95.04	4017.64
	SMN	258.34	4.33	119	7.35	8	236	54.45	2755.88
	CPI	290.32	3.53	153	42.69	9	330	83.67	3861.40
	DPI	271.86	3.45	132	21.61	8	301	65.83	2912.12
	FLI	293.88	3.29	174	31.20	9	344	95.50	4474.49
	NLI	268.30	3.68	111	33.10	8	286	54.00	2299.04

expected reward collected for the small and medium benchmark sets while, the MA2 shows a slight advantage for the large benchmark set. However, when considering the number of optimisation steps, the average time per step and the total running time, both algorithms exhibit similar performance with no significant differences.

The results for the FA2 exhibit a distinct behaviour. As indicated in Table 7.6 and in comparison with the previous two tables, the FA2 consistently achieves a higher average expected reward for the medium and large benchmark sets (for the small benchmark set, the MA1 and the MA2 slightly dominate the FA2). With regard to the average number of unvisited mandatory nodes, the results follow a similar trend highlighted for the DSTOP-ST-MIN-P. Similar observations (made for the DSTOP-ST-MIN-P results) regarding the average number of branching nodes, the number of generated columns and the column generation time also hold for the DSTOP-ST-MIN-PL. An analysis of the reported tables reveals that the same trend observed for the DSTOP-ST-MIN-P also applies to the DSTOP-ST-MIN-PL.

Figure 7.3 provides a graphical summary of the results (for the DSTOP-ST-MIN-PL) presented in the previous tables subdivided by instances size and number of available vehicles (m). The same trend observed for the TOP-ST-MIN is also evident in these sub-figures.

Finally, although not explicitly shown in the tables, it is worth noting that approximately 40% of the optimisation steps involve decisions where no vehicles move,

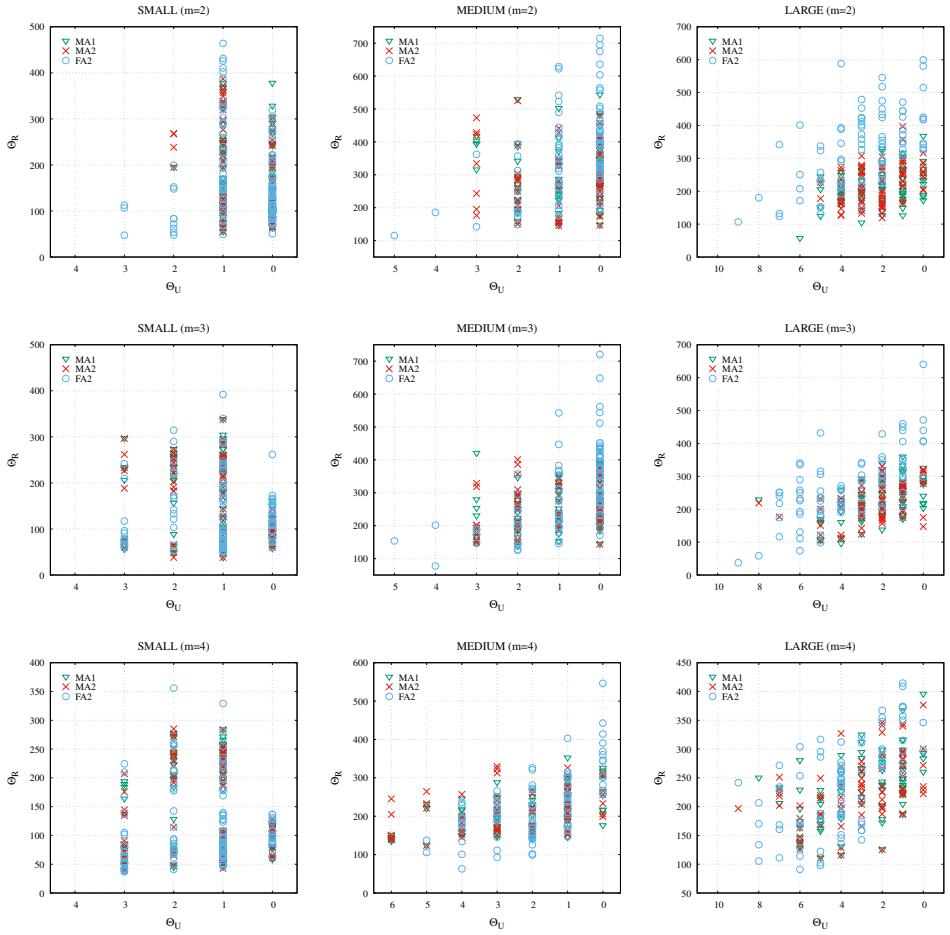


Fig. 7.3: A summary of the results of the Branch & Regret algorithms for the DSTOP-ST-MIN-PL.

while around 60% involve decisions where only a single vehicle moves. Only a negligible fraction of the optimisation steps involve multiple vehicles moving simultaneously which occurs primarily at the start when each vehicle departs from the source node.

7.5 Conclusions

In conclusion, addressing the complexities of the TOP-ST-MIN under uncertainty and dynamic conditions requires adaptive and efficient methodologies. This chapter has introduced four algorithms based on the Branch & Regret paradigm that leverage multiple scenario information in order to take decisions. Extensive computational analysis also highlights that the algorithms yield varying results depending on the instances size. In particular, the FA2 outperforms the other two approaches in terms of solution quality, especially on the medium and large benchmark sets, although it proves to be significantly slower. Finally, the deterministic variant of the DSTOP-

ST-MIN introduced in Section 7.1 will be used in Chapter 8 in order to dynamically evaluate the impact of the fairness on solution quality.

8 Fairness in static and dynamic TOP-ST-MIN contexts

The challenge of distributing resources equitably among self-interested parties (players or agents) arises frequently in several real scenarios and has been extensively studied in disciplines such as mathematics, game theory and operations research. These problems aim to ensure each party receives a fair share optimising resource efficiency from a central perspective. The literature covers a broad range of such problems, distinguished by resource types, fairness criteria, agent preferences and other factors influencing allocation quality.

The content covered in this chapter is inspired by the study on the Ambulance Routing Problem (ARP) discussed in Section 2.3, where the problem was also examined from a fairness perspective. That analysis also provided insights into the concept of the *price of fairness*, which quantifies the difference between an efficient solution, obtained without considering fairness, and a fair solution that incorporates equity-related criteria.

This chapter investigates the impact on the TOP-ST-MIN solution quality in three different settings from a fairness standpoint. Such settings represent three different stakeholders involved in the problem, that is the service provider, the customers, and the workers. In the analysis, the impact of fairness on both *static* and *dynamic* operative context is evaluated. In a static context, the planning decisions are taken once for the whole time horizon while in a dynamic context the final planning is the result of a sequence of decisions taken in successive time intervals.

The chapter begins with a brief overview of the three main fairness functions commonly discussed in the literature, followed by illustrative examples that recall how fairness was modelled in the ARP (Sect.8.1). After that, Section 8.2 presents several indices needed to measure the impact of fairness on the solution quality. Next, Section 8.3 describes the three settings considered in a static operative context. The first setting represents the perspective of the service provider, and is focused on balancing team profits. The second one shifts to a more practical setting related to the operative context in which the TOP-ST-MIN can be used. The aim is to balance the cumulative waiting times of the served customers as a quality measure of the service provided. The last setting focuses on the workers perspective in which the workload of the teams should be balanced. The quantitative analysis in a static context is finally reported (Sect. 8.3.4).

Finally, Section 8.4 extends the above analysis to the dynamic operative context reporting the results in Section 8.4.1.

8.1 How the fairness is modelled in the literature

This section provides an overview of the three fairness functions introduced in Bertsimas et al. [16] and Nicosia et al. [90]. Both papers described the functions in a general context where agents either cooperate or compete to achieve an objective. The value of each agent's contribution is measured through a generic utility measure to maximise.

Max-min fairness Following the principle of the *Rawlsian justice* [97], the objective is to identify a solution where the least satisfied agent achieves the maximum possible utility. This ensures that the agent with the lowest utility still receives the highest feasible benefit. Formally, denoting f_a a generic utility measure for the agent a, the task consists in finding a solution $\bar{x}$ such that:

$$\bar{x} = \arg\max_{x \in X} \min_{a \in A} f_a(x), \tag{8.1}$$

where, X and A represent the set of feasible solutions and the set of agents.

Relative max-min fairness The absolute max-min fairness guarantees a certain utility level for one agent but can result in significant losses for others, especially when the agent in question would not achieve substantial utility independently. To address this, [58] introduced an alternative approach that maximises the minimum utility in proportion to the maximum utility each agent can attain. Formally, denoting f_a^* the maximal utility score that the agent a can reach over all feasible solutions, the task consists in finding a solution $\bar{x}$ such that:

$$\bar{x} = \arg\max_{x \in X} \min_{a \in A} \frac{f_a(x)}{f_a^*}, \tag{8.2}$$

where, X and A represent the set of feasible solutions and the set of agents.

Proportional fairness The proportional fairness [63] is achieved when no other solution provides a total relative improvement for a subset of agents that exceeds the total relative loss inflicted to the remaining agents. More precisely, the goal is to find a solution $\bar{x}$ satisfying:

$$\sum_{a \in A} \frac{f_a(y) - f_a(\bar{x})}{f_a(\bar{x})} \leq 0, \quad \forall\, y \in X, \tag{8.3}$$

where, X and A represent the set of feasible solutions and the set of agents.

The differences among the solutions for a certain problem whose optimisation is driven by one of the above three objective functions are not straightforward. To foster a better understanding of such differences, this section reports the examples made for the ARP, the problem described in Section 2.3. To this end, the metaheuristic introduced in Section 2.4.2 for the ARP was extended to consider relative max-min and proportional fairness, alongside the max-min fairness already addressed in the study.

The following figures illustrate three examples of solutions composed by two routes, depicted in blue and orange, where the hierarchical objective function was designed in order to maximise the overall profit collected before balancing the waiting time for the urgent patients (red nodes). Figures 8.1, 8.2, and 8.3 present an example of the specific ARP instance "P10S3O1" (with scattered urgent patients displacement as reported in Table 2.6). The examples were obtained by adopting the min-max, proportional and relative min-max fairness functions, respectively.

Each figure further reports two indices in order to evaluate the impact of the fairness in a solution. The first one, the Maximum Waiting Time (MWT) measures the maximum waiting time experienced by an urgent patient. The second one, the Sum of Waiting Times (SWT) measures the sum of waiting times experienced by the urgent patients. As shown, while the overall profit collected (i.e., the efficiency objective) remains the same across all examples, the fairness-related measures differ.

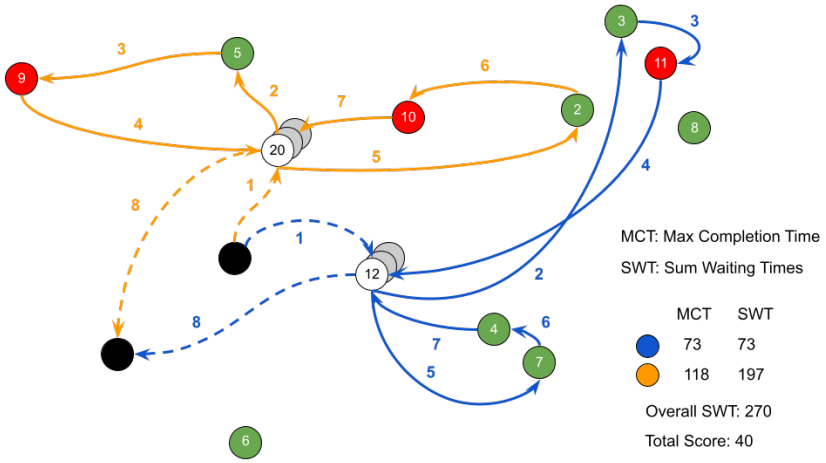


Fig. 8.1: An example of a solution considering the min-max fairness function (eq. 8.1).

Specifically, in Figure 8.1, where the min-max fairness function (eq. 8.1) is used inside the objective function, the MWT is 118 (as represented by the orange route), while the SWT is 270. Therefore, the min-max approach seems to prioritise the urgent customer experiencing the maximum waiting time.

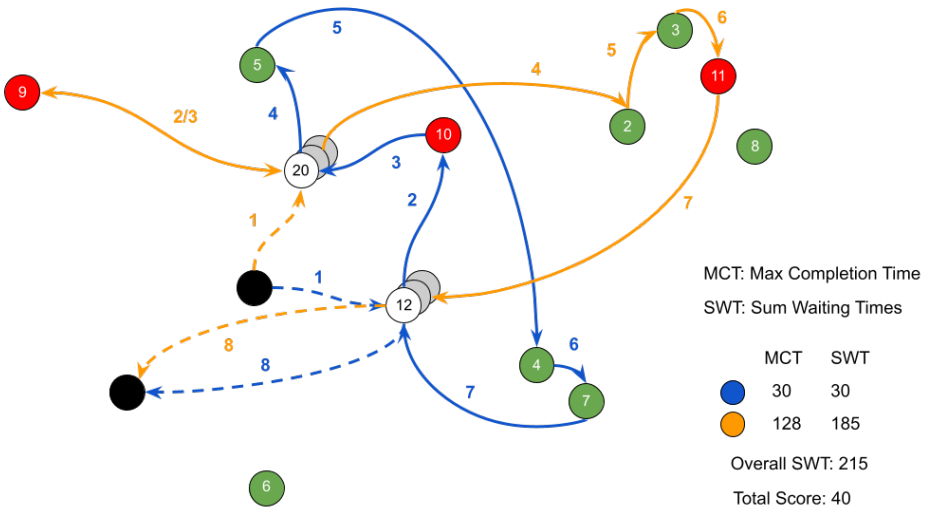


Fig. 8.2: An example of a solution considering the proportional fairness function (eq. 8.3).

Instead, in Figure 8.2, where the proportional min-max fairness function (eq. 8.3) is used inside the objective function, the MWT is 128 (as represented by the blue route), while the SWT decreases to 215. Consequently, the proportional fairness appears to provide a more equitable solution for all teams.

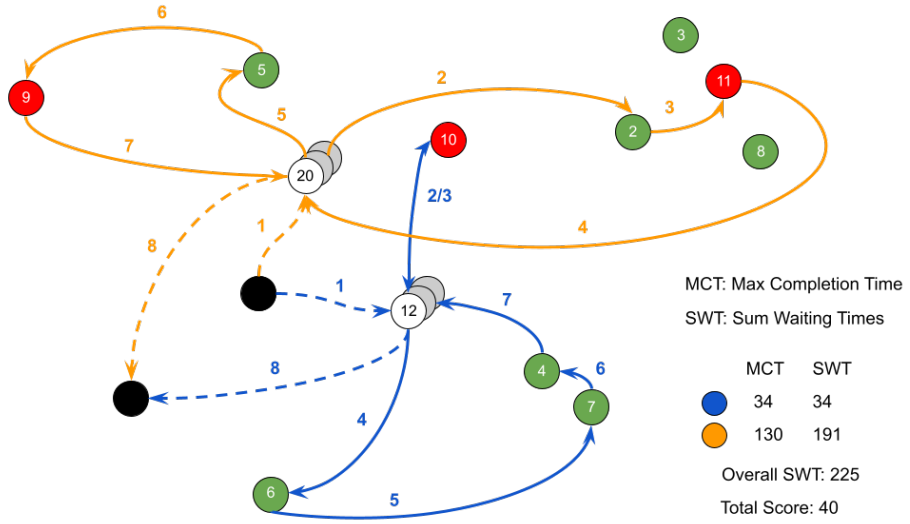


Fig. 8.3: An example of a solution considering the relative min-max fairness function (eq. 8.2).

Finally, in Figure 8.3, where the relative min-max fairness function (eq. 8.2) is used inside the objective function, the MWT increases to 130 (as represented by the blue route), while the SWT decreases to 225. Then, the relative min-max fairness function appears to represent a middle ground between the other two approaches.

8.2 How to evaluate the impact of fairness

As highlighted by the previous examples, it becomes evident that evaluating the fairness of a solution becomes essential. For this reason, this section introduces a set of fairness indices developed to evaluate the effects of the previously defined fairness functions on a solution. These indices will be used to formulate the objective functions discussed in Section 8.3.

Min fairness indices The min and relative min fairness indices are defined as follows:

$$\min_{a \in A} f_a(x), \tag{8.4}$$

where, x is a feasible solution and A is the set of agents.

Max-min difference fairness indices The max-min and relative max-min fairness indices are defined as follows:

$$\max_{a \in A} f_a(x) - \min_{a \in A} f_a(x), \tag{8.5}$$

where, x is a feasible solution and A is the set of agents.

Sum of absolute differences fairness indices The sum of absolute differences fairness indices are defined as follows:

$$\sum_{a_1 \in A} \sum_{a_2 \in A} \left| f_{a_1}(x) - f_{a_2}(x) \right|, \tag{8.6}$$

where, x is a feasible solution and A is the set of agents.

The first fairness index is based solely on the agent with the minimum (or relative minimum) utility score. The second index considers both the agents with the maximum (or relative maximum) and minimum (or relative minimum) utility scores. Finally, the third index takes into account the utility (or relative utility) scores of all agents in the system. Preliminary tests indicate that the last set of indices provides a more accurate evaluation of the fairness of a solution. For this reason, this index has been employed throughout all the experiments.

8.3 Fairness in a static operative context

This section introduces the fairness objective functions relative to three different settings considered for the TOP-ST-MIN. The first function seeks to balance the profits collected by the teams from the customers, based on a setting derived from the literature. The second objective aims to balance the cumulative waiting times of served customers, capturing a practical operational concern and serving as an measure of quality service from the customers viewpoints. The last objective, motivated by a different real-world requirement, aims to balance the workload among teams. It is worth noting that the three objective functions represent three different stakeholder perspectives, that are the provider of the service, the customers and the workers, respectively.

8.3.1 Balancing the profits

An initial fairness application in the context of the TOP-ST-MIN focuses on balancing the profits collected by each team. In this regard, three distinct objective functions are introduced:

Max-min fairness The objective function can be formulated as follows:

$$\max_{x \in X} \min_{r \in R} q_r(x), \tag{8.7}$$

where, X, R and q_r represent the set of feasible solutions, the set of teams and the profit collected by the vehicle r in the solution x, respectively. The aim of this objective function is to maximise the minimal profit collected by the teams.

Relative max-min fairness The objective function can be formulated as follows:

$$\max_{x \in X} \min_{r \in R} \frac{q_r(x)}{q_r^*}. \tag{8.8}$$

where, q_r^* represents the maximal profit that the vehicle r can collect. The aim of this objective function is to maximise the minimal relative profit collected by the teams.

Proportional fairness Mathematically, the aim is to find a solution $\bar{x}$ such that:

$$\sum_{r \in R} \frac{q_r(x) - q_r(\bar{x})}{q_r(\bar{x})} \leq 0, \quad \forall\, x \in X. \tag{8.9}$$

Unfortunately, the above definition cannot be directly applied unless the solution $\bar{x}$ is already known, which would imply that the problem has been solved beforehand.

It is worth noting that the quantity previously defined as q_r^* (for the relative max-min fairness) simplifies to q^* because the teams are indistinguishable from each other. This indicates that the max-min fairness and the relative max-min fairness represent the same approach.

8.3.2 Balancing the quality of the service

Another practical fairness application in the context of the TOP-ST-MIN involves balancing the cumulative waiting times intended as a measure of the quality of the service from the customers perspective. In this regard, three different objective functions are proposed:

Min-max fairness The objective function can be formulated as follows:

$$\max_{x \in X} \; \alpha_1 \sum_{r \in R} q_r(x) - \alpha_2 \max_{r \in R} \sum_{i \in \hat{N}} w_r^i(x), \tag{8.10}$$

where, X, R and w_r^i represent the set of feasible solutions, the set of teams and the waiting time experienced by the customer i served by the route r in the solution x, respectively.

Max-difference fairness The objective function can be formulated as follows:

$$\max_{x \in X} \; \alpha_1 \sum_{r \in R} q_r(x) - \alpha_2 \left(\max_{r \in R} \sum_{i \in \hat{N}} w_r^i(x) - \min_{r \in R} \sum_{i \in \hat{N}} w_r^i(x) \right), \tag{8.11}$$

where, X, R and w_r^i represent the set of feasible solutions, the set of teams and the waiting time experienced by the customer i served by the route r in the solution x, respectively.

Sum of absolute differences fairness The objective function can be formulated as follows:

$$\max_{x \in X} \alpha_1 \sum_{r \in R} q_r(x) - \alpha_2 \left(\sum_{r_1 \in R} \sum_{r_2 \in R} \left| \sum_{i \in \hat{N}} w^i_{r_1}(x) - \sum_{i \in \hat{N}} w^i_{r_2}(x) \right| \right), \tag{8.12}$$

where, X, R and w^i_r represent the set of feasible solutions, the set of teams and the waiting time experienced by the customer i served by the route r in the solution x, respectively.

All the objective functions are formulated using a multi-objective framework, considering both the profit and the cumulative waiting times of the served customers. It is important to note that, for each objective function, the coefficients α_1 and α_2 are chosen in order to form a convex combination (i.e., $\alpha_1 + \alpha_2 = 1$). These coefficients were determined through preliminary experiments to ensure a balanced trade-off, preventing one objective from dominating the other.

8.3.3 Balancing the workload

A more practical fairness application in the context of the TOP-ST-MIN involves balancing the teams workloads which can be defined as: (i) the sum of service and travel times of a team, or (ii) the sum of service times of a team. In this regard, three different objective functions are introduced:

Min-max fairness The objective function can be formulated as follows:

$$\max_{x \in X} \alpha_1 \sum_{r \in R} q_r(x) - \alpha_2 \max_{r \in R} w_r(x), \tag{8.13}$$

where, X, R and w_r represent the set of feasible solutions, the set of teams and the workload associated to the route r in the solution x, respectively.

Max-difference fairness The objective function can be formulated as follows:

$$\max_{x \in X} \alpha_1 \sum_{r \in R} q_r(x) - \alpha_2 \left(\max_{r \in R} w_r(x) - \min_{r \in R} w_r(x) \right), \tag{8.14}$$

where, X, R and w_r represent the set of feasible solutions, the set of teams and the workload associated to the route r in the solution x, respectively.

Sum of absolute differences fairness The objective function can be formulated as follows:

$$\max_{x \in X} \alpha_1 \sum_{r \in R} q_r(x) - \alpha_2 \left(\sum_{r_1 \in R} \left| w_{r_1}(x) - w_{r_2}(x) \right| \right), \tag{8.15}$$

where, X, R and w_r represent the set of feasible solutions, the set of teams and the workload associated to the route r in the solution x, respectively.

All the objective functions are formulated using a multi-objective framework, considering both the profit and the teams workload. It is important to note that, for each objective function, the coefficients α_1 and α_2 are chosen in order to form a convex combination (i.e., $\alpha_1 + \alpha_2 = 1$). These coefficients were determined through preliminary experiments to ensure a balanced trade-off, preventing one objective from dominating the other.

8.3.4 Quantitative analysis

This section provides a detailed quantitative analysis to validate the impact of the fairness measure. The analysis compares solution quality between the efficiency case, where only the total collected profit is maximised, and the fairness objectives that also take into account the balancing of a specific measure. The experiments were carried out on a 64-bit Windows machine with an Intel(R) Core(TM) i9-12900H, 3.20GHz and 16 GB of RAM.

It is worth noting that all the objective functions defined below were implemented and evaluated by adapting the VND described in Section 5.6.2 for the TOP-ST-MIN, modifying its objective function in order to incorporate the fairness measures introduced in Section 8.3. Finally, all the fairness indices mentioned in this quantitative analysis correspond to the sum of absolute differences (eq. (8.6)).

Table 8.1 presents a summary of the statistics for the efficient solutions of the DSTOP-ST-MIN-P associated to the solutions obtained using the VND. In particular, the five central columns (the ones labelled with Φ) represent an average of the fairness index applied to different measures. The measures considered are the profit collected (Φ_{P}), the sum of travel and service times (Φ_{SSTT}), the sum of service times (Φ_{SST}), the number of served customers (Φ_{SC}) and the cumulative waiting times of the served customers (Φ_{CWT}). Such an index can be interpreted as an unbalancing factor for a specific measure.

The purpose of Table 8.1 is to identify which measures are not balanced in the efficient solutions, defined as the solutions obtained taking only into account the total profit gathered. It is important to note that a value close to zero for a given measure indicates that the measure is quite balanced. As shown in the table, the

Tab. 8.1: A summary of some measures concerning the efficiency case for the TOP-ST-MIN-P.

m		$\Phi_{\mathbf{P}}$	$\Phi_{\mathbf{SSTT}}$	$\Phi_{\mathbf{SST}}$	$\Phi_{\mathbf{SC}}$	$\Phi_{\mathbf{CWT}}$	$T_{\max}$
2	**ALL**	**67.11**	**0.07**	**0.73**	**0.58**	**306.42**	**138.33**
	CMN	56.95	0.18	0.87	1.85	314.21	138.75
	SMN	79.81	0.07	0.55	1.00	427.05	137.81
	CPI	72.72	0.13	1.74	0.44	422.31	138.33
	DPI	61.50	0.01	0.28	0.72	306.42	138.33
3	**ALL**	**148.00**	**0.31**	**2.27**	**0.70**	**444.07**	**86.67**
	CMN	168.00	0.40	5.20	0.80	504.29	92.50
	SMN	108.00	0.33	3.59	1.90	335.06	75.00
	CPI	148.67	0.21	1.63	1.40	451.70	86.67
	DPI	147.33	0.76	5.88	1.07	444.07	86.67
4	**ALL**	**351.89**	**1.17**	**6.67**	**3.89**	**567.48**	**66.16**
	CMN	335.20	1.58	7.69	4.75	621.36	69.38
	SMN	393.63	0.38	14.41	7.25	651.38	58.13
	CPI	390.86	1.21	11.03	6.93	692.40	66.16
	DPI	312.93	1.29	4.27	3.71	567.48	66.16

measures corresponding to the service times, service and travel times and served customers (Φ_{SST}, Φ_{SSTT} and Φ_{SC}) are almost balanced in the efficient solutions, as reflected by their fairness index values that are very small. The only measures that are not yet balanced turned out to be the ones associated to the profit and the waiting times (Φ_{P} and Φ_{CWT}). Therefore, the goal is to balance these measures using the objective functions defined in sections 8.3.1 and 8.3.2 maintaining the aim of maximising the overall profit collected.

Table 8.2 presents the comparison between the efficient and fair solutions when balancing the profits (on the left side) and the cumulative waiting times of the customers (on the right side). The table is divided into two distinct parts. The left one represents the part of the table where the objective balances the profit among the vehicles. In particular, the table reports the comparisons for two objective functions: (i) the Max-min (eq. (8.7)) and (ii) the Proportional (eq.(8.9)). The right section, instead, shows the comparison when balancing the cumulative waiting times of the customers. In this case, the table reports the comparisons for three objective functions: (i) the Min-max (eq.(8.10)), (ii) the Max-diff (eq.(8.11)) and (iii) the Sum-abs-diff (eq.(8.12)).

All the columns in the table report an average relative difference for a specific measure with respect to the efficient solutions. Specifically, the column Δ_{OBJ} represents the average relative difference (in percentage) for the objective function value (the *price of fairness*), the column $\Delta_{\Phi_{\text{P}}}$ shows the average relative difference (in percentage) for the fairness index related to the profit and the column $\Delta_{\Phi_{\text{CWT}}}$ indicates

Tab. 8.2: The comparison between the efficient and fair solutions concerning the profit collected (on the left) and the cumulative waiting times (on the right) for the DSTOP-ST-MIN-P. The values reported in table are relative differences with respect to the efficient solutions expressed in percentage.

		Max-min		**Proportional**		**Min-max**		**Max-diff**		**Sum-abs-diff**	
m		Δ_{OBJ}	Δ_{Φ_P}	Δ_{OBJ}	Δ_{Φ_P}	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$
2	**ALL**	**-3.51**	**-47.97**	**-4.25**	**18.13**	**-31.92**	**-95.96**	**-15.22**	**-99.68**	**-15.22**	**-99.68**
	CMN	-2.79	-63.39	-3.17	-14.14	-33.01	-95.81	-14.15	-99.63	-14.15	-99.63
	SMN	-4.50	-34.22	-5.75	46.91	-30.40	-97.34	-16.71	-99.79	-16.71	-99.79
	CPI	-4.87	-34.68	-4.94	35.98	-32.52	-96.87	-16.14	-99.79	-16.14	-99.79
	DPI	-2.10	-63.69	-3.54	-2.98	-31.29	-96.24	-14.27	-99.61	-14.27	-99.61
3	**ALL**	**-3.98**	**-23.65**	**-4.68**	**3.51**	**-9.14**	**-71.17**	**-13.35**	**-97.69**	**-13.57**	**-97.80**
	CMN	-2.58	-35.12	-5.02	-10.36	-9.06	-76.15	-13.73	-98.03	-14.83	-98.20
	SMN	-6.72	12.04	-4.02	46.67	-9.29	-57.15	-12.62	-96.75	-11.10	-96.65
	CPI	-3.52	-28.97	-6.48	-3.68	-8.75	-74.23	-14.87	-98.22	-14.44	-97.87
	DPI	-4.46	-18.28	-2.80	10.77	-9.54	-68.55	-11.78	-97.19	-12.66	-97.76
4	**ALL**	**-3.38**	**-40.47**	**-4.56**	**-16.58**	**-4.36**	**-36.93**	**-13.55**	**-95.31**	**-12.54**	**-93.39**
	CMN	-2.17	-47.87	-6.59	-23.99	-4.56	-45.89	-12.53	-95.43	-10.99	-94.32
	SMN	-6.78	-24.74	1.13	-0.83	-3.80	-36.75	-16.41	-96.59	-16.88	-93.38
	CPI	-3.11	-46.56	-5.69	-21.45	-4.53	-43.37	-13.03	-95.74	-11.17	-94.71
	DPI	-3.66	-32.87	-3.40	-10.50	-4.18	-42.96	-14.00	-95.81	-13.94	-93.23

the average relative difference (in percentage) for the fairness index related to the cumulative waiting times of the served customers.

It is important to note that a negative relative difference for the columns Δ_{Φ_P} and $\Delta_{\Phi_{CWT}}$ indicates that the corresponding measure is more balanced compared to the efficient solutions. Instead, a negative value for the Δ_{OBJ} column indicates that the overall profit collected in the efficient solutions is greater than in the fairness ones.

As seen in the left part of the Table 8.2, the Max-min fairness function appears to reduce the objective function value (Δ_{OBJ}) slightly less than the proportional fairness. As indicated by the Δ_{Φ_P} columns, Max-min fairness achieves a more balanced distribution of profit among teams compared to Proportional fairness. Specifically, for instances with two and three available vehicles, the relative differences in the proportional fairness columns are even positive meaning that the proportional fairness appears to deteriorate the balancing of this measure. Looking at the right part of the table, the Max-diff and Sum-abs-diff functions achieve the smallest reduction in the objective function value and the best balancing of the cumulative waiting times ($\Delta_{\Phi_{CWT}}$) for the served customers. This clearly demonstrates that both achieve a more balanced distribution of the cumulative waiting times of this measure. Finally, the instances with scattered mandatory nodes (SMN) appear to be more challenging in terms of balancing the profit collected across vehicles, particularly when using the Proportional fairness function.

Tables 8.3 and 8.4 follow the same structure introduced for the previously tables 8.1 and 8.2 but refer to the DSTOP-ST-MIN-PL. As shown in Table 8.3, the measures

Tab. 8.3: A summary of some measures concerning the efficient solutions for the DSTOP-ST-MIN-PL.

m		Φ_{P}	Φ_{SSTT}	Φ_{SST}	Φ_{SC}	Φ_{CWT}
2	**ALL**	**58.33**	**0.13**	**1.67**	**0.09**	**293.41**
	CMN	67.70	0.26	1.30	0.80	274.18
	SMN	51.12	0.04	1.96	0.46	308.21
	CPI	58.52	0.24	2.23	0.61	323.99
	DPI	58.13	0.02	1.11	0.78	262.83
	FLII	64.25	0.15	2.53	0.08	347.07
	NLII	37.00	0.07	1.41	0.10	100.26
3	**ALL**	**134.65**	**0.38**	**2.14**	**1.20**	**398.71**
	CMN	146.20	0.65	3.63	2.30	477.99
	SMN	123.10	0.29	0.66	0.40	319.44
	CPI	137.80	0.63	1.44	1.25	418.89
	DPI	131.50	0.95	3.97	1.65	378.53
	FLII	138.27	0.18	2.26	1.70	460.92
	NLII	123.80	0.97	2.03	1.90	212.09
4	**ALL**	**254.75**	**0.74**	**3.46**	**2.00**	**439.66**
	CMN	237.70	1.00	4.11	1.78	448.44
	SMN	297.38	0.15	8.34	5.06	417.73
	CPI	263.61	1.27	5.81	4.46	465.95
	DPI	245.89	1.50	3.07	1.68	413.37
	FLII	318.36	0.89	3.51	3.93	601.91
	NLII	191.14	0.72	3.57	0.36	277.42

corresponding to the columns Φ_{SSTT}, Φ_{SST} and Φ_{SC} are already balanced in the efficient solutions as for the DSTOP-ST-MIN-P. Also in this case, the only measures that remain unbalanced are the profit collected and the cumulative waiting times of the served customers.

Table 8.4 largely follows the same trend observed in the corresponding table for the DSTOP-ST-MIN-P. Focusing on the left part of the table, the Max-min fairness function confirms its superior performance over the proportional fairness function concerning the fairness index related to the profit collected. On the other hand, in the right part of the table, the Max-diff and Sum-abs-diff functions still outperform the Min-max function in balancing the cumulative waiting times of the served customers. Finally, concerning the reduction in the objective function, the Min-max function performs better in instances with three and four available vehicles but worse in those with only two available vehicles. No relevant considerations can be made for each feature generation method.

To conclude the analysis, the comparisons presented in Tables 8.1 and 8.3 indicate that certain measures such as workload (defined as the sum of service and travel times or just service times) and the number of served customers are already bal-

Tab. 8.4: The comparison between the efficient and fair solutions concerning the profit collected (on the left) and the cumulative waiting times (on the right) for the DSTOP-ST-MIN-PL. The values reported in table are relative differences with respect to the efficient solutions expressed in percentage.

		Max-min		Proportional		Min-max		Max-diff		Sum-abs-diff	
m		Δ_{OBJ}	Δ_{Φ_P}	Δ_{OBJ}	Δ_{Φ_P}	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$
2	**ALL**	**-2.56**	**-51.99**	**-2.31**	**-17.74**	**-30.07**	**-94.67**	**-16.73**	**-99.55**	**-16.73**	**-99.55**
	CMN	-2.92	-56.79	-3.24	-20.46	-32.83	-94.45	-15.35	-99.36	-15.35	-99.36
	SMN	-2.21	-47.10	-1.41	-14.97	-27.41	-94.83	-18.05	-99.67	-18.05	-99.67
	CPI	-3.26	-38.41	-2.55	-2.60	-30.80	-95.04	-16.64	-99.60	-16.64	-99.60
	DPI	-1.84	-65.67	-2.05	-32.98	-29.31	-94.23	-16.83	-99.48	-16.83	-99.48
	FLII	-3.15	-50.67	-2.39	-19.11	-30.53	-95.20	-16.90	-99.63	-16.90	-99.63
	NLII	0.43	-60.27	-1.91	-9.19	-27.75	-88.18	-15.90	-98.45	-15.90	-98.45
3	**ALL**	**-2.80**	**-34.27**	**-5.61**	**-7.58**	**-8.08**	**-61.47**	**-17.21**	**-98.36**	**-13.41**	**-97.10**
	CMN	-2.24	-22.98	-7.21	-6.63	-8.35	-64.14	-18.55	-98.65	-12.94	-97.38
	SMN	-3.46	-47.68	-3.69	-8.69	-7.75	-57.47	-15.61	-97.92	-13.98	-96.66
	CPI	-2.83	-34.33	-5.92	-9.80	-9.99	-63.66	-17.61	-98.58	-14.54	-97.07
	DPI	-2.76	-34.22	-5.28	-5.25	-6.08	-59.04	-16.79	-98.11	-12.23	-97.12
	FLII	-2.97	-24.06	-5.15	-3.66	-8.46	-63.05	-17.93	-98.45	-14.05	-97.36
	NLII	-2.04	-68.50	-7.59	-20.68	-6.39	-51.16	-14.04	-97.75	-10.63	-95.36
4	**ALL**	**-4.54**	**-34.85**	**-5.12**	**-7.84**	**-6.14**	**-40.81**	**-16.80**	**-93.94**	**-10.36**	**-88.54**
	CMN	-2.39	-42.62	-6.17	-10.29	-5.85	-45.77	-16.60	-94.75	-10.17	-89.75
	SMN	-10.66	-19.31	-2.16	-2.94	-6.99	-27.49	-17.40	-91.74	-10.88	-85.31
	CPI	-4.31	-36.05	-5.91	-13.22	-5.44	-40.81	-18.12	-95.02	-10.53	-89.88
	DPI	-4.78	-33.55	-4.32	-2.06	-6.86	-40.82	-15.46	-92.71	-10.18	-87.03
	FLII	-6.42	-31.20	-5.21	-3.97	-6.32	-42.20	-18.68	-96.21	-11.20	-90.36
	NLII	-2.00	-40.92	-5.00	-14.28	-5.91	-37.80	-14.26	-89.01	-9.22	-84.60

anced in the efficient solutions for the TOP-ST-MIN-P. In contrast, other measures, specifically the profit collected by each team and the cumulative waiting times of served customers, remain unbalanced. To address this, the objective functions introduced in Sections 8.3.1 and 8.3.2 have proved how to effectively balance these measures, as evidenced in Tables 8.2 and 8.4. Specifically, the Max-min fairness function (eq.(8.7)) proved more effective than the Proportional fairness function (eq.(8.9)) in balancing profit among teams. Conversely, the Max-diff (eq.(8.11)) and Sum-abs-diff (eq.(8.12)) objective functions demonstrated to achieve a better balancing for the cumulative waiting times compared to the Min-max (eq.(8.11)) function.

8.4 Fairness in a dynamic operative context

The fairness over time is a crucial aspect in optimisation problems where decisions are made sequentially and impact multiple resources across different time intervals. Unlike the fairness in a static context, fairness over time aims to maintain balance throughout the planning horizon in order to prevent long-term disadvantages for any individual or group. This is particularly crucial in areas such as workforce scheduling and vehicle routing, where dynamically ensuring a fair allocation of resources over

time could be essential. This section extends the previous analysis to a dynamic operative context. The aim of this section is to investigates how partial instance information can affect both the solution quality and the balancing achieved for the fairness in a static context.

Fairness over time is addressed by considering the deterministic variant of the problem presented in Section 7.1. The dynamic setting, which includes all related events discussed in Section 7.1, is re-examined from a fairness perspective assuming deterministic travel and service times. An instance adaptation procedure, similar to the one described in Section 7.2, is applied in order to modify the TOP-ST-MIN benchmark set introduced in Chapter 4. The dynamic engine relies on a DES mechanism to handle incoming events and aims to adjust the current solution in a way that maximises total profit while balancing the cumulative waiting times of the customers. Each time an event is processed, an optimisation step is performed that consists in a single run of an adapted version of the VND described in Section 5.6.2. The algorithm progressively builds a (partial) feasible solution $(v_1, \ldots, v_m)$ composed by m routes. Each term v_i is a route that includes the source node and a sequence of served customers and, eventually, the destination node. Finally, as for the Branch & Regret approaches introduced in Section 7.3, it may occur that no action can be taken because certain mandatory nodes cannot be served due to previous decisions. For this reason, the constraints for mandatory nodes are not treated as strict requirements (which could frequently result in infeasible solutions) but they are imposed through a substantial penalty in the objective function allowing the algorithm more flexibility.

8.4.1 Quantitative analysis

This section presents a detailed quantitative analysis focused on assessing the impact of the fairness in dynamic contexts where new informations become available over time. The analysis is divided in two parts and is based on the quality of service setting with the goal of balancing the cumulative waiting time experienced by customers. In the first part, an analysis similar to that presented in Section 8.3.4 is conducted, but within a dynamic context. In the second part, the impact of dynamicity is assessed while considering fairness.

To this end, the corresponding fairness functions (eq.(8.10)), (eq.(8.11)), and (eq.(8.12)) previously employed in, will also be adopted. Also in this case, all the results are subdivided by feature generation methods as depicted in Figure 4.7 and partitioned by number of available vehicles (m). The experiments were carried out on a 64-bit Windows machine with an Intel(R) Core(TM) i9-12900H, 3.20GHz and 16 GB of RAM.

All the tables present the same structure and are organised into three parts, one for each fairness function considered. The column Δ_{OBJ} represents the average relative

Tab. 8.5: The comparison between the efficient and fair solutions in a dynamic context concerning the cumulative waiting times for the DSTOP-ST-MIN-P. The values reported for the columns Δ_{OBJ} and Δ_{CWT} are relative differences with respect to the efficient solutions expressed in percentage while the values reported for the columns Δ_{U} are absolute differences.

		Min-max			**Max-diff**			**Sum-abs-diff**		
m		Δ_{OBJ}	$\Delta_{\Phi_{\text{CWT}}}$	Θ_{U}	Δ_{OBJ}	$\Delta_{\Phi_{\text{CWT}}}$	Θ_{U}	Δ_{OBJ}	$\Delta_{\Phi_{\text{CWT}}}$	Θ_{U}
2	**ALL**	**-10.97**	**-30.32**	**0.86**	**-29.24**	**-19.53**	**2.00**	**-29.24**	**-19.53**	**2.00**
	CMN	-13.43	-34.40	0.90	-40.43	-52.55	2.85	-40.43	-52.55	2.85
	SMN	-7.11	-23.45	0.81	-11.69	36.20	0.94	-11.69	36.20	0.94
	CPI	-8.18	-29.18	0.72	-29.85	10.19	2.33	-29.85	10.19	2.33
	DPI	-14.09	-31.25	1.00	-28.57	-43.61	1.67	-28.57	-43.61	1.67
3	**ALL**	**-10.31**	**-18.69**	**0.50**	**-31.53**	**-38.47**	**2.03**	**-46.83**	**-45.94**	**3.43**
	CMN	-1.07	-16.54	-0.25	-32.21	-39.83	2.28	-51.89	-45.72	3.91
	SMN	-34.74	-25.86	2.00	-27.97	-30.77	1.48	-33.73	-43.27	2.48
	CPI	-13.17	-27.49	0.53	-47.63	-47.15	2.93	-47.90	-45.24	3.47
	DPI	-7.42	-11.07	0.47	-15.22	-30.95	1.13	-45.75	-46.54	3.40
4	**ALL**	**-19.35**	**-25.83**	**0.82**	**-29.29**	**-33.87**	**1.86**	**-30.07**	**-37.08**	**1.57**
	CMN	-10.13	-17.65	0.30	-28.44	-36.14	2.15	-26.44	-33.03	1.70
	SMN	-51.53	-60.30	2.13	-32.24	-24.27	1.13	-42.73	-54.18	1.25
	CPI	-27.44	-37.90	1.43	-7.47	8.12	1.21	-32.74	-24.70	2.07
	DPI	-12.55	-15.77	0.21	-47.67	-68.83	2.50	-27.83	-47.40	1.07

difference (in percentage) of the objective function value (the *price of fairness*). The column $\Delta_{\Phi_{\text{CWT}}}$ shows the average relative difference (in percentage) of the fairness index related to the cumulative waiting times of the served customers. Finally, the columns Θ_{U} represents the average absolute difference regarding the number of unvisited mandatory customers between fair and efficient solutions. All the average relative differences reported in tables 8.5 and 8.6 are computed with respect to the efficient solutions. It is important to note that a negative relative difference in the $\Delta_{\Phi_{\text{CWT}}}$ column indicates that, on average, the efficient solutions achieve a more balanced measure, while a positive value suggests the opposite. In contrast, a negative relative difference in the Δ_{OBJ} column means that the efficient solutions have a higher average objective function value. Lastly, a positive value in the Θ_{U} column indicates that, on average, the fair solutions visit more mandatory nodes than the efficient ones.

Table 8.5 presents the comparison between efficient and fair solutions in a dynamic context. As can be seen from the table, the Min-max fairness function, achieves the smallest reduction in the objective function value. In contrast, the Max-diff and Sum-abs-diff objective functions generally achieve better balance in cumulative waiting times, with the exception of instances involving two available vehicles ($m = 2$), where the Min-max fairness function performs more effectively.

Tab. 8.6: The comparison between the efficient and fair solutions in a dynamic context concerning the cumulative waiting times for the DSTOP-ST-MIN-PL. The values reported for the columns Δ_{OBJ} and Δ_{CWT} are relative differences with respect to the efficient solutions expressed in percentage while the values reported for the columns Δ_{U} are absolute differences.

		Min-max			Max-diff			Sum-abs-diff		
m		Δ_{OBJ}	$\Delta_{\Phi_{\text{CWT}}}$	Θ_{U}	Δ_{OBJ}	$\Delta_{\Phi_{\text{CWT}}}$	Θ_{U}	Δ_{OBJ}	$\Delta_{\Phi_{\text{CWT}}}$	Θ_{U}
2	**ALL**	**-6.75**	**-19.54**	**-0.54**	**-25.86**	**-13.09**	**-1.72**	**-25.86**	**-13.09**	**-1.72**
	CMN	-5.90	-20.01	-0.75	-35.24	-42.15	-2.70	-35.24	-42.15	-2.70
	SMN	-7.48	-19.07	-0.38	-17.78	15.66	-0.96	-17.78	15.66	-0.96
	CPI	-3.27	-26.52	-0.30	-29.82	1.94	-2.26	-29.82	1.94	-2.26
	DPI	-10.65	-11.88	-0.78	-21.42	-29.57	-1.17	-21.42	-29.57	-1.17
	FLI	-8.23	-25.47	-0.56	-27.07	-13.93	-1.69	-27.07	-13.93	-1.69
	NLI	-0.84	35.58	-0.50	-21.07	-5.24	-1.80	-21.07	-5.24	-1.80
3	**ALL**	**-16.99**	**-29.05**	**-0.78**	**-33.45**	**-43.80**	**-1.98**	**-46.14**	**-52.12**	**-2.88**
	CMN	-6.71	-20.17	-0.20	-38.89	-44.94	-2.90	-56.33	-50.32	-4.55
	SMN	-31.55	-41.20	-1.35	-25.72	-42.23	-1.05	-31.70	-54.58	-1.20
	CPI	-20.84	-36.09	-1.05	-47.54	-52.29	-2.95	-50.90	-61.17	-3.25
	DPI	-13.06	-22.52	-0.50	-19.09	-35.93	-1.00	-41.30	-43.73	-2.50
	FLI	-14.43	-26.32	-0.77	-34.74	-44.44	-2.23	-51.69	-54.95	-3.80
	NLI	-27.12	-39.46	-0.80	-28.32	-41.34	-1.20	-24.14	-41.29	-0.10
4	**ALL**	**-20.98**	**-10.12**	**-0.68**	**-27.91**	**-22.08**	**-1.55**	**-29.32**	**-29.07**	**-1.39**
	CMN	-12.88	0.95	-0.43	-22.93	-19.96	-1.73	-25.51	-27.26	-1.65
	SMN	-45.73	-53.28	-1.31	-43.15	-30.37	-1.13	-40.96	-36.14	-0.75
	CPI	-22.64	-25.48	-1.14	-15.04	7.69	-1.36	-30.67	-26.42	-1.75
	DPI	-19.49	4.60	-0.21	-39.47	-50.62	-1.75	-28.12	-31.61	-1.04
	FLI	-23.02	-19.23	-1.04	-32.50	-27.99	-2.07	-33.25	-31.49	-1.79
	NLI	-18.88	0.37	-0.32	-23.16	-15.28	-1.04	-25.25	-26.28	-1.00

Finally, in terms of the average number of unserved customers, the Min-max objective consistently produces fair solutions that visit more mandatory nodes than the other two functions. The Max-diff and the Sum-abs-diff exhibit similar performance, except for the instances with three available vehicles where the Max-diff function is able to produce fair solutions where one more mandatory customer is visited. Finally, it is worth noting that the only case in which the efficient solution visits more mandatory nodes occurs in instances with clustered mandatory nodes and three available vehicles ($m = 3$).

Table 8.6 reports the analogous results for the DSTOP-ST-MIN-PL. The trend remains largely unchanged with the Min-Max fairness function still performing slightly better on the objective function value and the other two fairness function that perform slightly better on balancing the cumulative waiting times. Similar considerations also follow for the average number of unvisited mandatory customers.

The second part of the analysis is aimed to evaluate the impact of the dynamicity considering the balancing of the cumulative waiting times as fairness metric. Ta-

Tab. 8.7: The comparison of the fair solutions between a static and dynamic context concerning the cumulative waiting times for the DSTOP-ST-MIN-P. The values reported for the columns Δ_{OBJ} and Δ_{CWT} are relative differences with respect to the efficient solutions expressed in percentage while the values reported for the columns Δ_U are absolute differences.

		Min-max			Max-diff			Sum-abs-diff		
m		Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Θ_U	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Θ_U	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Θ_U
2	**ALL**	**-57.99**	**-1070.29**	**3.39**	**-73.19**	**-15767.02**	**4.53**	**-73.19**	**-15767.02**	**4.53**
	CMN	-56.31	-1069.54	3.15	-76.54	-9420.42	5.10	-76.54	-9420.42	5.10
	SMN	-60.24	-1071.37	3.69	-68.41	-25995.53	3.81	-68.41	-25995.53	3.81
	CPI	-54.62	-895.56	2.89	-72.10	-22473.54	4.50	-72.10	-22473.54	4.50
	DPI	-61.41	-1270.98	3.89	-74.29	-10693.14	4.56	-74.29	-10693.14	4.56
3	**ALL**	**-71.11**	**-65.59**	**3.30**	**-76.87**	**-1464.47**	**4.83**	**-81.99**	**-1340.77**	**6.23**
	CMN	-65.10	-108.90	2.10	-74.79	-1722.46	4.63	-81.88	-1701.89	6.26
	SMN	-82.90	6.97	5.70	-80.40	-1045.35	5.18	-82.28	-811.34	6.18
	CPI	-72.51	-50.72	3.60	-82.23	-1494.75	6.00	-82.41	-1276.58	6.53
	DPI	-69.64	-77.99	3.00	-71.49	-1445.03	3.67	-81.57	-1402.90	5.93
4	**ALL**	**-75.43**	**26.48**	**4.14**	**-76.16**	**-780.79**	**5.18**	**-76.70**	**-494.72**	**4.89**
	CMN	-71.05	1.69	2.95	-74.85	-802.80	4.80	-74.60	-661.90	4.35
	SMN	-87.57	77.07	7.13	-80.00	-710.51	6.13	-83.00	-152.80	6.25
	CPI	-79.94	48.95	4.79	-71.92	-1082.11	4.57	-80.02	-562.52	5.43
	DPI	-70.84	-0.74	3.50	-80.54	-407.28	5.79	-73.21	-430.05	4.36

ble 8.7 reports such a comparison for the DSTOP-ST-MIN-P. All the average relative differences reported in tables 8.5 and 8.6 are computed with respect to the static operative context. It is important to note that a negative relative difference in the $\Delta_{\Phi_{CWT}}$ column indicates that, on average, the static operative context achieves a more balanced measure, while a positive value suggests the opposite. In contrast, a negative relative difference in the Δ_{OBJ} column means that the static operative context provides a higher average objective function value. Lastly, a positive value in the Θ_U column reports the average number of unvisited mandatory nodes in the dynamic context.

As expected, the fairness in a static operative context achieves higher objective function values and balances the cumulative waiting times more effectively than the fairness over time. The Min-max fairness function achieves greater reductions in terms of objective function value, whereas the Max-diff and Sum-abs-diff functions are significantly more effective at balancing the cumulative waiting times. The only exception consists in the results concerning the instances characterised by four available vehicles ($m = 4$). In this case, the solutions obtained in the dynamic context achieve a slightly better balancing of cumulative waiting times compared to those obtained in the static context. This can be attributed to two main factors. First, all experiments are performed using a heuristic approach, specifically the VND. Second, in the dynamic context, mandatory nodes and incompatibilities are progressively added to better reflect real-world conditions. Finally, in terms of the average number of unvisited mandatory nodes, the Min-max function consistently yields solutions

Tab. 8.8: The comparison of the fair solutions between a static and dynamic context concerning the cumulative waiting times for the DSTOP-ST-MIN-PL. The values reported for the columns Δ_{OBJ} and Δ_{CWT} are relative differences with respect to the efficient solutions expressed in percentage while the values reported for the columns Δ_U are absolute differences.

		Min-max			Max-diff			Sum-abs-diff		
m		Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Θ_U	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Θ_U	Δ_{OBJ}	$\Delta_{\Phi_{CWT}}$	Θ_U
2	**ALL**	**-55.56**	**-767.16**	**3.28**	**-70.33**	**-10875.97**	**4.46**	**-70.33**	**-10875.97**	**4.46**
	CMN	-55.85	-912.00	3.15	-75.89	-6276.97	5.10	-75.89	-6276.97	5.10
	SMN	-55.30	-660.72	3.38	-64.81	-16965.47	3.96	-64.81	-16965.47	3.96
	CPI	-51.52	-704.95	2.78	-70.80	-13699.05	4.74	-70.80	-13699.05	4.74
	DPI	-59.65	-833.10	3.78	-69.84	-8186.13	4.17	-69.84	-8186.13	4.17
	FLI	-57.81	-768.24	3.39	-71.97	-13044.44	4.53	-71.97	-13044.44	4.53
	NLI	-44.67	-761.72	2.90	-62.17	-4490.80	4.20	-62.17	-4490.80	4.20
3	**ALL**	**-71.42**	**-25.74**	**3.83**	**-74.56**	**-2238.26**	**5.03**	**-80.31**	**-1025.67**	**5.93**
	CMN	-65.31	-46.57	2.10	-74.43	-2588.48	4.80	-82.90	-1150.05	6.45
	SMN	-78.67	0.53	5.55	-74.71	-1898.48	5.25	-77.18	-879.66	5.40
	CPI	-72.50	-9.94	4.20	-80.09	-2002.24	6.10	-82.03	-729.07	6.40
	DPI	-70.33	-41.27	3.45	-68.83	-2434.56	3.95	-78.56	-1359.52	5.45
	FLI	-70.97	-24.49	3.30	-75.31	-2140.78	4.77	-82.55	-966.01	6.33
	NLI	-73.32	-31.95	5.40	-71.43	-2675.49	5.80	-70.91	-1246.83	4.70
4	**ALL**	**-70.93**	**-5.08**	**4.09**	**-70.08**	**-789.06**	**4.96**	**-72.77**	**-328.40**	**4.80**
	CMN	-67.46	-40.72	3.13	-67.50	-1053.06	4.43	-70.84	-436.35	4.35
	SMN	-80.89	66.46	6.50	-77.46	-338.81	6.31	-78.31	-126.22	5.94
	CPI	-73.53	19.54	4.36	-66.42	-1281.94	4.57	-74.92	-364.82	4.96
	DPI	-68.23	-32.83	3.82	-73.69	-409.30	5.36	-70.59	-296.37	4.64
	FLI	-74.91	24.38	4.14	-74.66	-927.12	5.18	-77.05	-284.70	4.89
	NLI	-65.55	-64.48	4.04	-64.19	-685.65	4.75	-67.10	-387.74	4.71

that visit more mandatory nodes than the other two fairness functions. No relevant considerations can be made for each feature generation method. A similar trend is observed for the Table 8.8 that reports the same comparison but for the DSTOP-ST-MIN-PL.

In summary, with respect to the first part of the comparisons, the fairness over time approach successfully balances the cumulative waiting times of served customers in a dynamic context, as demonstrated in Tables 8.5 and 8.6, albeit at the expense of a reduction of the objective function value. On the other hand, in the second part of the comparisons, the system's dynamic nature impacts the solution quality (objective function of the TOP-ST-MIN-P) to some extent but has a much greater effect on the fairness measures, as shown in Tables 8.7 and 8.8.

8.5 Conclusions

This chapter has explored several fairness functions and indices in order to define and measure the impact of the fairness on solution quality. The chapter also investigated the application of fairness in the TOP-ST-MIN from the perspectives of different

stakeholders. From the viewpoint of service workers, the efficiency case already tends to balance team workloads. In contrast, from the perspectives of service providers and customers, balancing the profit collected by each team and the cumulative waiting times experienced by served customers emerges as a key concern.

Additionally, the discussion on the fairness over time provides considerable insights in order to evaluate how the resource distribution can evolve dynamically. In particular, the dynamic application of the fairness to the cumulative waiting times results in a significantly better balancing than the efficiency case. Finally, the chapter concludes by comparing fair solutions in static and dynamic contexts to evaluate the impact of dynamicity on fairness.

9 Conclusions

This thesis has primarily focused on extending the classical Team Orienteering Problem (TOP-ST-MIN) by introducing the Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (TOP-ST-MIN). This new variant generalises two real-world applications in healthcare: the Daily Swab Test Collection Problem (DSTCP) during a pandemic and the Ambulance Routing Problem (ARP) in a post-disaster scenario. Both problems involve complex resource allocation challenges in which the aim consists in optimising the use of available resources respecting constraints such as time limits, task prioritisation and critical patient needs. The TOP-ST-MIN captures these operational requirements integrating service times, mandatory nodes and incompatibilities in a single combinatorial optimisation problem.

A thorough literature review provided a detailed analysis of existing research inherent to different aspects of the TOP-ST-MIN. The review explored a range of methodologies developed for the TOP-ST-MIN under deterministic perspective with particular emphasis on the features of the problem that are: (i) service times at nodes, (ii) a set of mandatory nodes and (iii) a set of incompatibilities between nodes.

This thesis examines the TOP-ST-MIN by presenting new methods for generating problem instances. These methods are designed to be fully deterministic (except for the service times) to allow precise control over the characteristics of the generated instances. This structured methodology ensures that the generated instances are both challenging in order ot offer valuable insights into the computational impact of the problem's key characteristics. The instance generation process specifically targets the three key features characterising the TOP-ST-MIN, with the goal of evaluating how each contributes to the computational difficulty. This approach provides deeper insight into why algorithms may perform differently across instances, allowing for a more distinct comparison between methods.

A key contribution of this thesis is the development and evaluation of mathematical models and algorithms for solving the TOP-ST-MIN under a deterministic perspective. It introduced an exact method based on the cutting-plane methodology for finding optimal solutions and heuristic approaches designed to efficiently handle larger problem instances. The effectiveness of these algorithms was demonstrated through a comprehensive quantitative analysis on the newly generated instances providing a

comparative evaluation across different groups of instances based on the adopted feature generation method. In particular, the proposed heuristic approaches identified 48 additional feasible solutions for the TOP-ST-MIN-P and 99 for the TOP-ST-MIN-PL that were not found by the exact method. Although the Cuts-Separation Heuristic (CSH) method proved to be more effective than the Variable Neighbourhood Descent (VND), it proved to be slower. Overall, the results demonstrated that both methods, especially the CSH, are competitive with the best-performing algorithms for the classical TOP.

This thesis also investigated the TOP-ST-MIN under uncertainty considering stochastic and dynamic aspects, such as variability in travel and service times and uncertainty in the arrival of new customers requests. As a preliminary step, the Team Orienteering Problem (TOP) was extended in order to consider stochastic travel times. This allows to expand the TOP framework to real-world scenarios characterised by uncertainty. In this regard, the Stochastic Team Orienteering Problem (STOP) is introduced and investigated. Two mathematical formulations for the STOP are presented, along with two exact algorithms developed using the Integer L-shaped methodology.

The thesis also introduced the Dynamic and Stochastic Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (DSTOP-ST-MIN) which is a generalisation of the STOP with dynamic aspects where also the service times are modelled by random variables. To tackle the challenges introduced by dynamic environments such as new customer requests, changing node requirements and emerging new incompatibilities, the DSTOP-ST-MIN is addressed using the Branch & Regret paradigm. To this end, four innovative algorithms based on the Branch & Regret paradigm were developed. These algorithms employ a scenario-based approach that combines Discrete Event Simulation (DES) with optimisation allowing to build an adaptive engine based on real-time information. Extensive computational experiments showed the robustness of these methods proving their ability to adapt to evolving circumstances while maintaining high-quality solutions.

Fairness in resource allocation was also a key aspect of this work. Three distinct settings corresponding to three fairness perspectives were introduced, each reflecting a specific stakeholder's point of view in a static operative context. The first focuses on balancing the profit among teams and represents the perspective of the service provider. The second targets on the balancing of cumulative waiting times, serving as an indicator of the service quality from the customers standpoint. The third aims to balance the workload across teams and reflects the viewpoint of the service workers. The analysis is then extended to a dynamic operative context. In the static context, the computational results revealed that the workloads are inherently balanced in the efficiency case, which corresponds to the problem without integrating any fairness considerations. On the other hand, the collected profit and the total waiting times need to be balanced and the results show the capability of the objective functions

introduced to achieve a good balancing while maintaining the overall score collected. In the dynamic context, such balancing results are confirmed, though to a lesser extent.

To summarise, the thesis successfully addressed all the research questions outlined in Chapter 1 providing a detailed investigation of the TOP-ST-MIN from theoretical and practical points of view. The study proved the practical relevance and applicability of the TOP-ST-MIN in real-world contexts through the development of several mathematical formulations, different algorithmic approaches and extensive computational evaluations. The integration of the fairness, stochasticity and dynamic aspects further enriched the analysis making a significant contribution to the field of combinatorial optimisation.

Looking ahead, several promising directions for future research emerge. An interesting road could be the development of algorithms adopting the online optimisation framework, where decisions must be made sequentially as new data becomes available, without full knowledge of future events. Such methods could be directly compared to the Branch & Regret approaches introduced in Chapter 7, in order to evaluate their respective advantages and disadvantages in dynamic and uncertain environments. Moreover, the TOP-ST-MIN can also be extended to model more realistic environments considering multi-depot settings, vehicle capacities and time window constraints, which represent common requirements in logistics and transportation applications.

Another promising direction consists in the integration of learning-based methods, such as reinforcement learning which could improve performance in dynamic environments by using historical data. Finally, the concept of fairness over time, introduced in Chapter 8 could be further extended within a stochastic framework in order to extend the concept of fairness to uncertainty contexts.

Bibliography

[1] Brahim Aghezzaf and Hassan Fahim. “Solving the Capacitated Team Orienteering Problem with Time Windows Through Variable Neighborhood Search”. In: *International Review on Computers and Software (IRECOS)* 10 (2015), p. 1134 (cit. on p. 48).

[2] Ana María Anaya-Arenas, Angel Ruiz, and Jacques Renaud. “Importance of Fairness in Humanitarian Relief Distribution”. In: *Production Planning & Control* 29 (2018), pp. 1145–1157 (cit. on pp. 24, 58).

[3] Enrico Angelelli, Claudia Archetti, Carlo Filippi, and Michele Vindigni. “The Probabilistic Orienteering Problem”. In: *Computers & Operations Research* 81 (2017), pp. 269–281 (cit. on p. 53).

[4] Enrico Angelelli, Claudia Archetti, Carlo Filippi, and Michele Vindigni. “A Dynamic and Probabilistic Orienteering Problem”. In: *Computers & Operations Research* 136 (2021), p. 105454 (cit. on p. 56).

[5] Roberto Aringhieri, Sara Bigharaz, Alessandro Druetto, et al. “The daily swab test collection problem”. In: *Annals of Operations Research* 335.3 (2024), pp. 1449–1470 (cit. on pp. 7, 8, 12, 44, 48).

[6] Roberto Aringhieri, Sara Bigharaz, Davide Duma, and Alberto Guastalla. “Fairness in ambulance routing for post disaster management”. In: *Central European Journal of Operations Research* 30 (2022), pp. 189–211 (cit. on pp. 13, 44, 47, 58).

[7] Roberto Aringhieri, Sara Bigharaz, Davide Duma, and Alberto Guastalla. “Novel Applications of the Team Orienteering Problem in Health Care Logistics”. In: *Optimization in Artificial Intelligence and Data Sciences*. Ed. by Lavinia Amorosi, Paolo Dell’Olmo, and Isabella Lari. Cham: Springer International Publishing, 2022, pp. 235–245 (cit. on pp. 7, 13, 48).

[8] Roberto Aringhieri, Simone Bocca, Luigi Casciaro, and Davide Duma. “A Simulation and Online Optimization approach for the Real-Time Management of Ambulances”. In: *2018 Winter Simulation Conference (WSC)*. Vol. 2018-December. IEEE, 2019, pp. 2554–2565 (cit. on p. 37).

[9] Roberto Aringhieri, Roberto Cordone, Alberto Guastalla, and Andrea Grosso. “Constructive and Destructive Methods in Heuristic Search”. In: *Discrete Diversity and Dispersion Maximization: A Tutorial on Metaheuristic Optimization*. Ed. by Rafael Martí and Anna Martínez-Gavara. 2023, pp. 65–91 (cit. on p. 63).

[10] Roberto Aringhieri, Paolo Landa, Patrick Soriano, Elena Tanfani, and Angela Testi. “A Two-Level Metaheuristic for the Operating Room Scheduling and Assignment Problem”. In: *Computers & Operations Research* 54 (2015), pp. 21–34 (cit. on p. 95).

[11] Lucas Assunção and Geraldo Mateus. “A cutting-plane algorithm for the Steiner team orienteering problem”. In: *Computers & Industrial Engineering* 135 (2019), pp. 922–939 (cit. on pp. 44, 46, 48, 63, 83, 88, 89).

[12] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. “A Dynamic Vehicle Routing Problem with Multiple Delivery Routes”. In: *Annals of Operations Research* 199.1 (2012), pp. 103–112 (cit. on p. 55).

[13] Narayanaswamy Balakrishnan and Vladimir Nevzorov. “A Primer on Statistical Distributions”. In: (2003) (cit. on pp. 112, 113).

[14] Egon Balas. “The prize collecting traveling salesman problem”. In: *Networks* 19.6 (1989), pp. 621–636 (cit. on p. 3).

[15] Evripidis Bampis, Bruno Escoffier, and Sasa Mladenovic. “Fair resource allocation over time”. In: *AAMAS 2018 - 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 766–773 (cit. on p. 59).

[16] Dimitris Bertsimas, Vivek F. Farias, and Nikolaos Trichakis. “The Price of Fairness”. In: *Operations Research* 59.1 (2011), pp. 17–31 (cit. on pp. 57, 158).

[17] Dimitris Bertsimas, Vivek F. Farias, and Nikolaos Trichakis. “On the Efficiency-Fairness Trade-off”. In: *Management Science* 58.12 (2012), pp. 2234–2250 (cit. on p. 57).

[18] Nicola Bianchessi, Renata Mansini, and M. Grazia Speranza. “A branch-and-cut algorithm for the Team Orienteering Problem”. In: *International Transactions in Operational Research* 25.2 (2018), pp. 627–635 (cit. on pp. 74, 75, 125).

[19] Arjen Boin, Raphael Bossong, Vera-Karin Brazova, et al. “Civil Security and the European Union: A survey of European civil security systems and the role of the EU in building shared crisis management capacities”. In: *UI Papers* 2014 (2014) (cit. on p. 18).

[20] Yuri Boykov and Vladimir Kolmogorov. “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (2004), pp. 1124–1137 (cit. on p. 80).

[21] Steven E. Butt and Tom M. Cavalier. “A heuristic for the multiple tour maximum collection problem”. In: *Computers & Operations Research* 21.1 (1994), pp. 101–111 (cit. on pp. 5, 72).

[22] Federica Di Camillo, Alessandro Marrone, Stefano Silvestri, Paola Tessari, and Alessandro R. Ungaro. *The Italian Civil Security System*. Tech. rep. Istituto Affari Internazionali (IAI), 2014, pp. 83–88 (cit. on p. 18).

[23] Ann Campbell, Michel Gendreau, and Barrett Thomas. “The orienteering problem with stochastic travel and service times”. In: *Annals of Operations Research* 186 (2011), pp. 61–81 (cit. on pp. 6, 53).

[24] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. “A fast and effective heuristic for the orienteering problem”. In: *European Journal of Operational Research* 88.3 (1996), pp. 475–489 (cit. on pp. 31, 61, 68, 102, 108, 109, 131, 132).

[25] Nicos Christofides. “Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem”. In: *Operations Research Forum* 3 (1976) (cit. on p. 26).

[26] Victor Hugo Corrêa, Hang Dong, Manuel Iori, et al. “An iterated local search for a multi-period orienteering problem arising in a car patrolling application”. In: *Networks* 83.1 (2024), pp. 153–168 (cit. on pp. 44, 48).

[27] Jean-François Côté, Thiago Alves de Queiroz, Francesco Gallesi, and Manuel Iori. “A Branch-and-Regret Algorithm for the Same-Day Delivery Problem”. In: *Transportation Research Part E: Logistics and Transportation Review* 177 (2023), p. 103226 (cit. on p. 137).

[28] Duc-Cuong Dang, Rym Nesrine Guibadj, and Aziz Moukrim. “An effective PSO-inspired algorithm for the team orienteering problem”. In: *European Journal of Operational Research* 229.2 (2013), pp. 332–344 (cit. on pp. 108–110).

[29] George B. Dantzig, Delbert R. Fulkerson, and Selmer M. Johnson. “Solution of a Large-Scale Traveling-Salesman Problem”. In: *Journal of the Operations Research Society of America* 2.4 (1954), pp. 393–410 (cit. on pp. 2, 78, 100).

[30] George B. Dantzig and John H. Ramser. “The Truck Dispatching Problem”. In: *Management Science* 6.1 (1959), pp. 80–91 (cit. on p. 2).

[31] Mauro Dell’Amico, Francesco Maffioli, and Peter Värbrand. “On prize-collecting tours and the asymmetric travelling salesman problem”. In: *International Transactions in Operational Research* 2.3 (1995), pp. 297–308 (cit. on p. 3).

[32] Abraham Duarte, Jesús Sánchez-Oro, Nenad Mladenović, and Raca Todosijević. “Variable Neighborhood Descent”. In: *Handbook of Heuristics*. Ed. by Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende. Springer Cham, 2018, pp. 341–367 (cit. on p. 92).

[33] Joydeep Dutta, Partha Barma, Anupam Mukherjee, Samarjit Kar, and Tanmay De. “A multi-objective open set orienteering problem”. In: *Neural Computing and Applications* 32 (2020), pp. 13953–13969 (cit. on pp. 44, 48).

[34] Racha El-Hajj, Duc-Cuong Dang, and Aziz Moukrim. “Solving the Team Orienteering Problem with Cutting Planes”. In: *Computers & Operations Research* 74 (2016), pp. 21–30 (cit. on p. 123).

[35] Gunes Erdogan and Gilbert Laporte. “The orienteering problem with variable profits”. In: *Networks* 61.2 (2013), pp. 104–116 (cit. on p. 45).

[36] Lanah Evers, Kristiaan Glorie, Suzanne van der Ster, Ana Isabel Barros, and Herman Monsuur. “A Two-Stage Approach to the Orienteering Problem with Stochastic Weights”. In: *Computers & Operations Research* 43 (2014), pp. 248–260 (cit. on p. 53).

[37] Luca Ferretti, Chris Wymant, Michelle Kendall, et al. “Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing”. In: *Science* 368.6491 (2020) (cit. on pp. 15, 193).

[38] Matteo Fischetti, Juan José Salazar González, and Paolo Toth. “Solving the Orienteering Problem through Branch-and-Cut”. In: *INFORMS Journal on Computing* 10.2 (1998), pp. 133–148 (cit. on p. 78).

[39] Eduardo Feo Flushing, Luca Gambardella, and Gianni Di Caro. “Spatially-Distributed Missions With Heterogeneous Multi-Robot Teams”. In: *IEEE Access* 9 (2021), pp. 67327–67348 (cit. on pp. 44, 48, 50).

[40] Maria Furtado, Pedro Munari, and Reinaldo Morabito. “Pickup and delivery problem with time windows: A new compact two-index formulation”. In: *Operations Research Letters* 45.4 (2017), pp. 334–341 (cit. on p. 76).

[41] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. “A survey on algorithmic approaches for solving tourist trip design problems”. In: *Journal of Heuristics* 20 (2014), pp. 291–328 (cit. on p. 6).

[42] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. “Efficient Metaheuristics for the Mixed Team Orienteering Problem with Time Windows”. In: *Algorithms* 9 (2016), p. 6 (cit. on p. 49).

[43] Bezalel Gavish and Stephen Graves. *The Traveling Salesman Problem and Related Problems*. Tech. rep. Massachusetts Institute of Technology, Operations Research Center, 1978 (cit. on pp. 75, 114, 126, 143).

[44] Michel Gendreau, Alain Hertz, and Gilbert Laporte. “A Tabu Search Heuristic for the Vehicle Routing Problem”. In: *Management Science* 40.10 (1994), pp. 1276–1290 (cit. on p. 29).

[45] Bruce Golden, L. Levy, and Rakesh Vohra. “The orienteering problem”. In: *Nav Res Logist* 34 (1987), pp. 307–318 (cit. on pp. 4, 145).

[46] Bruce Golden, Xingyin Wang, and Edward Wasil. “The Evolution of the Vehicle Routing Problem”. In: *The Evolution of the Vehicle Routing Problem: A Survey of VRP Research and Practice from 2005 to 2022*. 2023, pp. 1–64 (cit. on p. 1).

[47] Felix Gündling and Tim Witzel. “Time-Dependent Tourist Tour Planning with Adjustable Profits”. In: *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Vol. 85. Open Access Series in Informatics. 2020, pp. 1–14 (cit. on pp. 44, 48).

[48] Farouk Hammami, Monia Rekik, and Leandro C. Coelho. “A hybrid adaptive large neighborhood search heuristic for the team orienteering problem”. In: *Computers & Operations Research* 123 (2020), p. 105034 (cit. on pp. 97, 108).

[49] Saïd Hanafi, Renata Mansini, and Roberto Zanotti. “The multi-visit team orienteering problem with precedence constraints”. In: *European Journal of Operational Research* 282.2 (2020), pp. 515–529 (cit. on pp. 44, 48, 50, 83, 88, 89).

[50] William E. Hart, Charles Laird, John Watson, et al. *Pyomo - Optimization Modeling in Python*. Springer, 2017 (cit. on p. 33).

[51] Kenneth A. Hawick and Heath A. James. “Enumerating Circuits and Loops in Graphs with Self-Arcs and Multiple-Arcs”. In: *International Conference on Foundations of Computer Science*. 2008 (cit. on pp. 80, 124, 129).

[52] M. Held and Richard Karp. “The Traveling Salesman Problem and Minimum Spanning Trees”. In: *Operations Research* 18.6 (1970), pp. 1138–1162 (cit. on pp. 77, 121).

[53] Keld Helsgaun. “An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic”. In: *European Journal of Operational Research* 126.1 (2000), pp. 106–130 (cit. on pp. 77, 121).

[54] Fateme Heris, Seyed Ghannadpour, Morteza Bagheri, and Fatemeh Zandieh. “A new accessibility based team orienteering approach for urban tourism routes optimization (A Real Life Case)”. In: *Computers & Operations Research* 138 (2021), p. 105620 (cit. on pp. 4, 44, 48).

[55] Fredrick S. Hillier. “Chance-Constrained Programming with 0-1 or Bounded Continuous Decision Variables”. In: *Management Science* 14.1 (1967), pp. 34–57 (cit. on p. 127).

[56] Lars Magnus Hvattum, Arne Løkketangen, and Gilbert Laporte. “A Branch-and-Regret Heuristic for Stochastic and Dynamic Vehicle Routing Problems”. In: *Networks* 49.4 (2007), pp. 330–340 (cit. on p. 137).

[57] Huan Jin and Barrett Thomas. “Team orienteering with uncertain rewards and service times with an application to phlebotomist intrahospital routing”. In: *Networks* 73.4 (2019), pp. 453–465 (cit. on pp. 44, 48).

[58] Ehud Kalai and Meir Smorodinsky. “Other Solutions to Nash’s Bargaining Problem”. In: *Econometrica* 43.3 (1975), pp. 513–518 (cit. on p. 158).

[59] Marisa G. Kantor and Moshe B. Rosenwein. “The Orienteering Problem with Time Windows”. In: *Journal of the Operational Research Society* 43.6 (1992), pp. 629–635 (cit. on p. 5).

[60] Richard M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103 (cit. on p. 72).

[61] Özlem Karsu and Alec Morton. “Inequity Averse Optimization in Operational Research”. In: *European Journal of Operational Research* 245.2 (2015), pp. 343–359 (cit. on p. 57).

[62] Liangjun Ke, Laipeng Zhai, Jing Li, and Felix T.S. Chan. “Pareto Mimic Algorithm: An Approach to the Team Orienteering Problem”. In: *Omega* 61 (2016), pp. 155–166 (cit. on p. 108).

[63] Frank P. Kelly, Aman K. Maulloo, and David K.H. Tan. “Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability”. In: *Journal of the Operational Research Society* 49 (1998) (cit. on p. 158).

[64] Vincent Knight, Paul Harper, and L. Smith. “Ambulance allocation for maximal survival with heterogeneous outcome measures”. In: *Omega* 40 (2012) (cit. on p. 24).

[65] Serhan Kotiloglu, T. Lappas, Konstantinos Pelechrinis, and Panagiotis Repoussis. “Personalized multi-period tour recommendations”. In: *Tourism Management* 62 (2017), pp. 76–88 (cit. on pp. 44, 46, 48).

[66] Harold W. Kuhn. “The Hungarian Method for the Assignment Problem”. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97 (cit. on p. 141).

[67] Ching-Chung Kuo, Fred Glover, and Krishna S. Dhir. “Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming”. In: *Decision Sciences* 24.6 (1993), pp. 1171–1185 (cit. on p. 63).

[68] Nacima Labadie, Renata Mansini, Jan Melechovský, and Roberto Calvo. “The Team Orienteering Problem with Time Windows: An LP-based Granular Variable Neighborhood Search”. In: *European Journal of Operational Research* 220 (2012), pp. 15–27 (cit. on p. 49).

[69] Gilbert Laporte and François V. Louveaux. “The Integer L-Shaped Method for Stochastic Integer Programs with Complete Recourse”. In: *Operations Research Letters* 13.3 (1993), pp. 133–142 (cit. on pp. 111, 115).

[70] Jiaojiao Li, Jianghan Zhu, Guansheng Peng, et al. “Branch-Price-and-Cut Algorithms for the Team Orienteering Problem with Interval-Varying Profits”. In: *European Journal of Operational Research* 319.3 (2024), pp. 793–807 (cit. on pp. 44, 45, 48).

[71] Jin Li, Qinmin Wu, Xueqian Li, and Daoli Zhu. “Study on the Time-Dependent Orienteering Problem”. In: *2010 International Conference on E-Product E-Service and E-Entertainment*. 2010, pp. 1–4 (cit. on p. 6).

[72] Zhixue Liao and Weimin Zheng. “Using a heuristic algorithm to design a personalized day tour route in a time-dependent stochastic environment”. In: *Tourism Management* 68 (2018), pp. 284–300 (cit. on pp. 44, 48).

[73] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. “Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency”. In: *Knowledge and Information Systems* 54 (2018), pp. 375–406 (cit. on pp. 45, 48, 51).

[74] Shen Lin. “Computer Solutions of the Traveling Salesman Problem”. In: *The Bell System Technical Journal* 44.10 (1965), pp. 2245–2269 (cit. on p. 94).

[75] Shen Lin and Brian W. Kernighan. “An Effective Heuristic Algorithm for the Traveling-Salesman Problem”. In: *Operations Research* 21.2 (1973), pp. 498–516 (cit. on p. 26).

[76] Shih-Wei Lin and Vincent Yu. “Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing”. In: *Computers & Industrial Engineering* 114 (2017), pp. 195–205 (cit. on pp. 44, 47).

[77] Shih-Wei Lin and Vincent F. Yu. “Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing”. In: *Computers & Industrial Engineering* 114 (2017), pp. 195–205 (cit. on p. 48).

[78] Andrea Lodi, Philippe Olivier, Gilles Pesant, and Sriram Sankaranarayanan. “Fairness over time in dynamic resource allocation with an application in healthcare”. In: 203.1–2 (2022), 285–318 (cit. on p. 60).

[79] Yongliang Lu, Una Benlic, and Qinghua Wu. “A memetic algorithm for the Orienteering Problem with Mandatory Visits and Exclusionary Constraints”. In: *European Journal of Operational Research* 268 (2018), pp. 54–69 (cit. on pp. 44, 47).

[80] Yongliang Lu, Una Benlic, and Qinghua Wu. “A memetic algorithm for the Orienteering Problem with Mandatory Visits and Exclusionary Constraints”. In: *European Journal of Operational Research* 268.1 (2018), pp. 54–69 (cit. on pp. 48, 49).

[81] J. MacQueen. “Some methods for classification and analysis of multivariate observations”. In: 1967 (cit. on pp. 25, 63).

[82] Silvano Martello, David Pisinger, and Paolo Toth. “Dynamic Programming and Strong Bounds for the 0-1 Knapsack Problem”. In: *Management Science* 45.3 (1999), pp. 414–424 (cit. on p. 26).

[83] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. 1990 (cit. on p. 26).

[84] Claude E. Miller, Albert W. Tucker, and Richard A. Zemlin. “Integer Programming Formulation of Traveling Salesman Problems”. In: *Journal of the ACM* 7.4 (1960), pp. 326–329 (cit. on p. 17).

[85] Nicola Morandi, Roel Leus, and Hande Yaman. “The Orienteering Problem with Drones”. In: *Transportation Science* 58 (1 2023), pp. 240–256 (cit. on pp. 44, 48, 52).

[86] Valentina Morandi. “Bridging the User Equilibrium and the System Optimum in Static Traffic Assignment: How the Cooperation Among Drivers Can Solve the Congestion Problem in City Networks”. PhD thesis. 2021, pp. 89–119 (cit. on p. 59).

[87] Airam Expósito Márquez, Simona Mancini, Julio Brito, and José Moreno-Pérez. “A fuzzy GRASP for the tourist trip design with clustered POIs”. In: *Expert Systems with Applications* 127 (2019) (cit. on p. 48).

[88] Maurizio Naldi, Gaia Nicosia, Andrea Pacifici, and Ulrich Pferschy. “Maximin Fairness in Project Budget Allocation”. In: *Electronic Notes in Discrete Mathematics* 55 (2016), pp. 65–68 (cit. on p. 57).

[89] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. “On spectral clustering: analysis and an algorithm”. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. 2001, pp. 849–856 (cit. on p. 25).

[90] Gaia Nicosia, Andrea Pacifici, and Ulrich Pferschy. “Price of Fairness for allocating a bounded resource”. In: *European Journal of Operational Research* 257.3 (2017), pp. 933–943 (cit. on pp. 24, 38, 58, 158).

[91] Cynthia Porras Nodarse, Boris Pérez-Cañedo, David Pelta, and Jose Verdegay. “A Critical Analysis of a Tourist Trip Design Problem with Time-Dependent Recommendation Factors and Waiting Times”. In: *Electronics* 11 (2022), pp. 1–11 (cit. on p. 44).

[92] Christos Orlis, Demetrio Laganá, Wout Dullaert, and Daniele Vigo. “Distribution with Quality of Service Considerations: The Capacitated Routing Problem with Profits and Service Level Requirements”. In: *Omega* 93 (2020), p. 102034 (cit. on p. 4).

[93] Pamela Palomo-Martínez, Angélica Salazar-Aguilar, and Victor Albornoz. “Formulations for the orienteering problem with additional constraints”. In: *Annals of Operations Research* 258 (2017), pp. 503–545 (cit. on p. 44).

[94] Pamela Palomo-Martínez, Angélica Salazar-Aguilar, Gilbert Laporte, and André Langevin. “A hybrid variable neighborhood search for the Orienteering Problem with mandatory visits and exclusionary constraints”. In: *Computers & Operations Research* 78 (2015), pp. 408–419 (cit. on pp. 9, 44, 47, 48, 71).

[95] Lucas Parada, Robin Legault, Jean-François Côté, and Michel Gendreau. “A Disaggregated Integer L-Shaped Method for the Stochastic Vehicle Routing Problem”. In: (2022) (cit. on pp. 115, 120).

[96] Artur Alves Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. “A Generic Exact Solver for Vehicle Routing and Related Problems”. In: *Mathematical Programming* 183 (2020), pp. 483–523 (cit. on p. 89).

[97] John Rawls. *A Theory of Justice: Original Edition*. Harvard University Press, 1971 (cit. on p. 158).

[98] Lorena Reyes-Rubiano, Carlos Ospina-Trujillo, Javier Faulin, et al. “The team orienteering problem with stochastic service times and driving-range limitations: a simheuristic approach”. In: *2018 Winter Simulation Conference* (2018), pp. 3025–3035 (cit. on pp. 44, 48).

[99] Lorena Reyes-Rubiano, Carlos Ospina-Trujillo, Javier Faulin, et al. “The Team Orienteering Problem with Stochastic Service Times and Driving-Range Limitations: A Simheuristic Approach”. In: 2018, pp. 3025–3035 (cit. on p. 54).

[100] Ulrike Ritzinger, Jakob Puchinger, and Richard F. Hartl. “A Survey on Dynamic and Stochastic Vehicle Routing Problems”. In: *International Journal of Production Research* 54.1 (2016), pp. 215–231 (cit. on p. 55).

[101] José Ruiz-Meza, Julio Brito, and Jairo Montoya-Torres. “A GRASP to solve the multi-constraints multi-modal team orienteering problem with time windows for groups with heterogeneous preferences”. In: *Computers & Industrial Engineering* 162 (2021), p. 107776 (cit. on pp. 44, 48).

[102] José Ruiz-Meza and Jairo Montoya-Torres. “Tourist trip design with heterogeneous preferences, transport mode selection and environmental considerations”. In: *Annals of Operations Research* 305 (2021), pp. 227–249 (cit. on p. 44).

[103] José Ruiz-Meza and Jairo R. Montoya-Torres. “A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines”. In: *Operations Research Perspectives* 9 (2022), p. 100228 (cit. on p. 4).

[104] Martin W. P. Savelsbergh and Marc Sol. “The General Pickup and Delivery Problem”. In: *Transportation Science* 29.1 (1995), pp. 17–29 (cit. on p. 2).

[105] Andreas Schulz and Nicolas Stier-Moses. “Efficiency and Fairness of System-Optimal Routing with User Constraints”. In: *Networks* 48 (2006), pp. 223–234 (cit. on p. 56).

[106] Laura Sebastia and Eliseo Marzal. “Extensions of the Tourist Travel Design Problem for different travel styles”. In: *Procedia Computer Science* 176 (2020), pp. 339–348 (cit. on p. 44).

[107] Hayk Sedrakyan and Nairi Sedrakyan. *A Useful Inequality*. 2018, pp. 107–125 (cit. on p. 128).

[108] Marius M. Solomon. “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints”. In: *Operations Research* 35.2 (1987), pp. 254–265 (cit. on p. 2).

[109] Foteini Stavropoulou, Panagiotis Repoussis, and Christos Tarantilis. “The Vehicle Routing Problem with Profits and Consistency Constraints”. In: *European Journal of Operational Research* 274 (2019), pp. 340–356 (cit. on pp. 44, 46, 48).

[110] Peng Sun, Lucas P. Veelenturf, Said Dabia, and Tom Van Woensel. “The time-dependent capacitated profitable tour problem with time windows and precedence constraints”. In: *European Journal of Operational Research* 264.3 (2018), pp. 1058–1073 (cit. on pp. 44, 48, 51).

[111] Kadri Sylejmani, Jürgen Dorn, and Nysret Musliu. “Planning the trip itinerary for tourist groups”. In: *Information Technology & Tourism* 17 (2017) (cit. on p. 48).

[112] René Séguin. “Problemes Stochastiques de Tournées de Véhicules, Unpublished Doctoral Thesis, Université de Montréal, Montréal”. In: (1994) (cit. on p. 115).

[113] Luca Talarico, Frank Meisel, and Kenneth Sörensen. “Ambulance routing for disaster response with patient groups”. In: *Computers & Operations Research* (2014) (cit. on pp. 18, 24, 37, 41).

[114] Seyed M. R. Iravani Taylan and Mark S. Daskin. “The Orienteering Problem with Stochastic Profits”. In: *IIE Transactions* 40.4 (2008), pp. 406–421 (cit. on p. 52).

[115] Kendall Taylor, Kwan Hui Lim, and Jeffrey Chan. “Travel Itinerary Recommendations with Must-see Points-of-Interest”. In: *Companion of the The Web Conference 2018* (2018), pp. 1198–1205 (cit. on pp. 44, 46, 48).

[116] Thomas C. Thayer and Stefano Carpin. “A Fast Algorithm for Stochastic Orienteering with Chance Constraints”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 7961–7968 (cit. on p. 54).

[117] Fletcher Thompson and Roberto Galeazzi. “Robust mission planning for Autonomous Marine Vehicle fleets”. In: *Robotics and Autonomous Systems* 124 (2019), p. 103404 (cit. on pp. 44, 48).

[118] T. Tsiligirides. “Heuristic Methods Applied to Orienteering”. In: *The Journal of the Operational Research Society* 35.9 (1984), pp. 797–809 (cit. on p. 31).

[119] Marlin W. Ulmer, Dirk C. Mattfeld, and Felix Köster. “Budgeting Time for Dynamic Vehicle Routing with Stochastic Customer Requests”. In: *Transportation Science* 52.1 (2018), pp. 20–37 (cit. on p. 55).

[120] Christine L. Valenzuela and Antonia J. Jones. “Estimating the Held-Karp lower bound for the geometric TSP”. In: *European Journal of Operational Research* 102.1 (1997), pp. 157–175 (cit. on p. 32).

[121] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. “The orienteering problem: A survey”. In: *European Journal of Operational Research* 209 (2011), pp. 1–10 (cit. on pp. 14, 74, 96, 97).

[122] Cedric Verbeeck, Pieter Vansteenwegen, and El-Houssaine Aghezzaf. “Solving the Stochastic Time-Dependent Orienteering Problem with Time Windows”. In: *European Journal of Operational Research* 255.3 (2016), pp. 699–718 (cit. on p. 53).

[123] Ton Volgenant and Roy Jonker. “A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation”. In: *European Journal of Operational Research* 9.1 (1982), pp. 83–89 (cit. on pp. 77, 121).

[124] Duc Minh Vu, Yannick Kergosien, Jorge Mendoza, and Pierre Desport. “Branch-and-Check Approaches for the Tourist Trip Design Problem with Rich Constraints”. In: *Computers & Operations Research* 138 (2022), p. 105566 (cit. on pp. 10, 43, 44, 47, 48).

[125] Nathan D. Wallace, He Kong, Andrew John Hill, and Salah Sukkarieh. “The Orienteering Problem with Replenishment”. In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)* (2020), pp. 973–978 (cit. on p. 44).

[126] Wenjie Wang, Hoong Chuin Lau, and Shih-Fen Cheng. “Exact and heuristic approaches for the multi-agent orienteering problem with capacity constraints”. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2017, pp. 1–7 (cit. on pp. 44, 48, 50).

[127] Warisa Wisittipanich and Chanagan Boonya. “Multi-objective Tourist Trip Design Problem in Chiang Mai City”. In: *IOP Conference Series: Materials Science and Engineering* 895 (2020), p. 012014 (cit. on pp. 44, 48).

[128] Qinxiao Yu, Yossiri Adulyasak, Louis-Martin Rousseau, Ning Zhu, and Shoufeng Ma. “Team Orienteering with Time-Varying Profit”. In: *INFORMS Journal on Computing* 34.1 (2022), pp. 262–280 (cit. on pp. 44, 48, 51).

[129] Qinxiao Yu, Kan Fang, Ning Zhu, and Shoufeng Ma. “A matheuristic approach to the orienteering problem with service time dependent profits”. In: *European Journal of Operational Research* 273.2 (2019), pp. 488–503 (cit. on pp. 44, 45, 48, 50).

[130] Vincent F. Yu, Nabila Yuraisyah Salsabila, Shih-Wei Lin, and Aldy Gunawan. “Simulated annealing with reinforcement learning for the set team orienteering problem with time windows”. In: *Expert Systems with Applications* 238 (2024), p. 121996 (cit. on pp. 48, 49).

[131] Shu Zhang, Jeffrey W. Ohlmann, and Barrett W. Thomas. “Dynamic Orienteering on a Network of Queues”. In: *Transportation Science* 52.3 (2018), pp. 691–706 (cit. on p. 54).

[132] Weimin Zheng, Haipeng Ji, Congren Lin, Wenhui Wang, and Bilian Yu. “Using a heuristic approach to design personalized urban tourism itineraries with hotel selection”. In: *Tourism Management* 76 (2020), p. 10956 (cit. on pp. 44, 45, 48).

[133] Weimin Zheng, Zhixue Liao, and Jing Qin. “Using a four-step heuristic algorithm to design personalized day tour route within a tourist attraction”. In: *Tourism Management* 62 (2017), pp. 335–349 (cit. on p. 48).

[134] Taylan İlhan, Seyed M. R. Iravani, and Mark S. Daskin. “The orienteering problem with stochastic profits”. In: *IIE Transactions* 40.4 (2008), pp. 406–421 (cit. on p. 6).

Glossary

ARP Ambulance Routing Problem.

CMN Clustered Mandatory Nodes.

CPA Cutting-Plane Algorithm.

CPI Clusters-based Physical Incompatibilities.

CSH Cuts-Separation Heuristic.

DES Discrete Event Simulation.

DPI Degree-based Physical Incompatibilities.

DSTCP Daily Swab Test Collection Problem.

DSTOP-ST-MIN Dynamic and Stochastic Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes.

DSTOP-ST-MIN-P Dynamic and Stochastic Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (only Physical incompatibilities).

DSTOP-ST-MIN-PL Dynamic and Stochastic Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (both Physical and Logical incompatibilities).

FLI Farthest Logical Incompatibilities.

ILA Integer L-shaped Algorithm.

NLI Nearest Logical Incompatibilities.

OP Orienteering Problem.

SMN Scattered Mandatory Nodes.

STOP Stochastic Team Orienteering Problem.

TOP Team Orienteering Problem.

TOP-ST-MIN Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes.

TOP-ST-MIN-P Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (only Physical incompatibilities).

TOP-ST-MIN-PL Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes (both Physical and Logical incompatibilities).

TTDP Tourist Trip Design Problem.

VND Varaible Neighbourhood Descent.

List of Figures

List of Tables

List of Publications

Publications related to the thesis

A. Guastalla, R. Aringhieri and P. Hosteins. *Heuristic approaches for a new variant of the Team Orienteering Problem*. arXiv (2025) [math.OC] DOI: 10.48550/arXiv.2507.06012. Under revision.

A. Guastalla, R. Aringhieri and P. Hosteins. *The Team Orienteering Problem with Service Times and Mandatory & Incompatible Nodes*. Computers & Operations Research vol, pages, 2025.

R. Aringhieri, S. Bigharaz, A. Druetto, D. Duma, A. Grosso and A. Guastalla. *The daily swab test collection problem*. Annals of Operations Research 335, 1449-1470, 2024.

R. Aringhieri, S. Bigharaz, D. Duma and A. Guastalla. *Novel applications of the team orienteering problem in health care logistics*. Optimization in Artificial Intelligence and Data Sciences, 235-245, 2022.

R. Aringhieri, S. Bigharaz, D. Duma and A. Guastalla. *Fairness in ambulance routing for post disaster management*. Central European Journal of Operations Research 30, 189–211.

Other publications during the PhD period

M. Di Cunzolo, M. Ronzani, R. Aringhieri, C. Di Francescomarino, C. Ghidini, A. Guastalla and E. Sulis. *Robust solutions via optimisation and predictive process monitoring for the scheduling of the interventional radiology procedures*. International Transactions in Operational Research 32, 2189-2214, 2024.

R. Aringhieri, R. Cordone, A. Guastalla and A. Grosso. *Constructive and Destructive Methods in Heuristic Search*. Springer Optimization and Its Applications 204, 65-91, 2023.

M. Di Cunzolo, A. Guastalla, R. Aringhieri, E. Sulis, I. A. Amantea, M. Ronzani, C. Di Francescomarino, C. Ghidini, P. Fonio and M. Grosso. *Combining Process Mining and Optimization: A Scheduling Application in Healthcare*. Business Process Management Workshops 460, 197-209, 2023.

A. Guastalla; E. Sulis; R. Aringhieri; S. Branchi; C. Di Francescomarino and C. Ghidini. *Workshift Scheduling Using Optimization and Process Mining Techniques: An Application in Healthcare*. 2022 Winter Simulation Conference (WSC), 1116-1127.

Colophon

This thesis was typeset with LaTeX 2_ε. It uses a slightly adapted version of the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.