Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de estudios de posgrado

A Branch and Cut Approach to The Orienteering Problem with Mandatory Visits and Conflicts

por

Marlene Pérez Franco

como requisito para obtener el grado de

maestría en ciencias de la ingeniería

con orientación en Sistemas

Julio, 2025

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de estudios de posgrado

A Branch and Cut Approach to The Orienteering Problem with Mandatory Visits and Conflicts

presentada por

Marlene Pérez Franco

como requisito para obtener el grado de

maestría en ciencias de la ingeniería

con orientación en Sistemas

Julio, 2025

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
Facultad de Ingeniería Mecánica y Eléctrica
Posgrado

Los miembros del Comité de Evaluación de Tesis recomendamos que la Tesis "A Branch and Cut Approach to The Orienteering Problem with Mandatory Visits and Conflicts", realizada por la estudiante Marlene Pérez Franco, con número de matrícula 2221298, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestría en Ciencias de la Ingeniería con Orientación en Sistemas.

El Comité de Evaluación de Tesis

Dr. Vincent André Lionel Boyer
Director

Dra. María Angélica Salazar Aguilar
Co-director

Dra. Pamela Jocelyn Palomo Martínez
Revisor

Dra. Diana Lucía Huerta Muñoz
Revisor

Dr. Igor Semionovich Litvinchev
Revisor

Vo.Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

Institución 190001

Programa 557543

Acta Núm. 4539

Ciudad Universitaria, a 05 de septiembre de 2025.

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

To my parents, Marlene and Raúl, my biggest thank you goes to you both. You have been my greatest strength and motivation throughout this period. You have always appreciated and encouraged me throughout my life, which has made me reach this point. This work is a testimony of the effort and love you have placed in me all my life. There are not enough words to thank you for the great things you have done for me. I love you infinitely.

To Lu, for always being there and wishing me the best. We have grown together and have faced every challenge along the way. Thanks for being there with me all the time. And to Nacho, your love walks with me every single day.

To Lili and Milo, for all of your love, your support, and trust. Thank you for sharing your wisdom and the good moments that we've shared. Your presence has been essential to me and I will always be grateful to have you in my life.

To Chely, for always being there for me, your love, and your support. Thank you for all our conversations and all of the advice you gave to me. I am very fortunate to have you in my life.

To Mary and Paco, your love, support, and help during the writing of this thesis were invaluable. Thank you for being a safe and happy place.

To Bri, for the moments of happiness, always being by my side, and bringing light. Thank you for your love, support, patience, and the ability to understand me; they were fundamental in this step.

To Ange, for being such a great friend and showing up at the right time. Your support and capacity to make me feel understood were key in this process. Thank you for your friendship and your warm. See you soon.

To Reno and Emi, for your beautiful friendship that made me feel at home. Thank you for your trust, support, and deep conversations.

To Fer, for being a great cousin and roommie. Thank you for all the laughs and support you gave me. Your presence always made things easier.

To Toño, my brother, for always being by my side. Thank you for the support, conversations, and growing together, always sharing the ups and downs of life. You are one of the most important people to me.

To my teachers, Drs. Angy, Vincent, Iris, César, Romeo, and Sara for teaching me many things, understanding me, and encouraging me all this time. Every one of you has significantly contributed to my development, and your knowledge has been indispensable to my training. Thank you for sharing your experience and motivating me to do my best.

To the School of Mechanical and Electrical Engineering (FIME) for providing me the opportunity to be part of this community. I am very proud to be in this faculty, which has empowered me to build myself professionally.

To the Autonomous University of Nuevo Leon (UANL), for each course and for each experience that allowed my growth. I am very proud to be a student at this university as I wouldn't be the person I am today, both academically and personally.

To SECIHTI for providing the scholarship that enabled me to focus on my studies and complete this research. I am grateful for your support, which has been crucial to this work.

Abstract

Marlene Pérez Franco.

Candidato para obtener el grado de Maestría en Ciencias de la Ingeniería con orientación en Sistemas.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Title of the study: A Branch and Cut Approach to The Orienteering Problem with Mandatory Visits and Conflicts.

Number of pages: 65.

This work proposed a Branch and Cut (B&C) method to solve the Orienteering Problem with Mandatory Visits and Conflicts (OPMVC), which is a combinatorial optimization problem that has extensive applicability in logistics, transportation, and network design, to mention a few. The goal is to find the route that maximizes the collected score, including all the mandatory and some optional vertices, without generating conflict between vertices or exceeding the time limit. The B&C algorithm is expected to outperform the performance of the best-known methods existing in the OPMVC literature in terms of computational efficiency and solution quality. Experimental analysis using benchmark instances from the literature confirms that the B&C algorithm performs effectively. The B&C method can cut down the computation times and offer optimum quality solutions at the same time by the use of undirected graphs in the formulation of the problem and the strategic integration of the relevant valid inequalities. In addition, this work has opened up prospects for extending the research to other routing problems with similar constraints, highlighting the need for introducing advanced optimization techniques into real-world problems.

Signature of the faculty adviser: 

Dr. Vincent André Lionel Boyer

Chapter 1

Introduction

This work focuses on the Orienteering Problem with Mandatory Visits and Conflicts (OPMVC). This problem is a variant of the Orienteering Problem (OP), it has mandatory visits, optional locations with associated scores, and constraints that forbid certain pairs of places from being visited on the same route. The goal is to maximize the score obtained by visiting some optional locations while ensuring that all mandatory places are visited within the given maximum time limit.

The OPMVC has diverse applications across our lives due to its nature of being a combinatorial problem. Delivery and logistics involve planning efficient routes to ensure that essential locations (such as distribution centers) are visited while maximizing additional deliveries within distance or schedule constraints. In supply chain management, the OPMVC is used to coordinate the flow of goods from suppliers to customers, ensuring visits to hubs, for example. In tourism, the OPMVC helps to design itineraries that include must-see attractions while maximizing the number of additional spots to visit in a limited travel time. In vehicle routing, the OPMVC can be used to plan routes for a delivery service. It requires delivering packages to specific points, where the customers have paid for urgent delivery and selecting additional clients who chose the free option. The aim is to serve all the required stops and attempt to make as many extra stops as possible in order to maximize the score. All within the space and time allowed in the vehicle and working hours.

This study contributes with an exact method for solving the OPMVC based on a Branch and Cut (B&C) framework. The innovation is that this is the first time all OPMVC instances of literature have been solved to optimality using an undirected formulation with valid inequalities added dynamically as the solving progresses.

1.1 Motivation

Although one metaheuristic and some exact methods have been proposed, not all of the OPMVC benchmark instances have been solved to optimality. Therefore, considering that other variants of the OP that have been successfully solved using the B&C method, there is an opportunity for this problem to be solved in the same way. The motivation for this thesis is to propose an exact method that can solve the OPMVC to optimality or at least improve the best-known solutions for this problem.

This would be beneficial not only in academic research but also in practical situations that share the same characteristics of the OPMVC.

1.2 Hypothesis

Reformulating the OPMVC using undirected graphs and solving it within a Branch and Cut framework is expected to improve computational efficiency by taking advantage of symmetry. This formulation is expected to accelerate convergence, leading to optimal or near-optimal solutions for benchmark instances. Consequently, this approach is anticipated to produce better solutions than the existing in the literature, in less time and provide a viable alternative for solving larger OPMVC instances, making it more practical for real-world applications. This assumption is based on the fact that existing variants of the Orienteering Problem (OP) have successfully been tackled with B&C techniques, especially by making use of cutting planes that enforce tightening the linear relaxation.

1.3 Objective

The primary objective of this thesis is to develop and evaluate an exact approach for solving the OPMVC using an undirected graph formulation, expecting to reduce computational complexity, accelerate convergence, and improve solution quality and efficiency compared to existing methods.

1.3.1 Specific Objectives

- Advance the state of the art in solving the OPMVC.

- Identify and adapt efficient cuts from the literature to enhance the solution process.
- Develop an effective separation method in the Branch and Cut framework to distinguish between applicable and non-applicable cuts.
- Disseminate the findings through publication in a peer-reviewed scientific journal.

1.4 Contribution

This study proposes an exact method to solve the OPMVC using a B&C algorithm. The contributions are the following.

- Formulation of an exact approach: The problem was reformulated to use undirected graphs, which reduced the number of variables. Consequently, B&C was approximately 90% faster than the benchmark MILP formulation and was able to solve instances with up to 262 vertices to optimality in less than five minutes.
- Identification of valid inequalities in the literature: Major inequalities valid in the literature were identified and incorporated. These are logical cuts, matching inequalities, infeasible path inequalities, and connectivity inequalities. It was determined that connectivity inequalities help to minimize the optimality gap and improve the overall performance of the algorithm.
- Separation Algorithm Development: Design and implement separation procedures to efficiently identify and apply valid cuts.
- Integration of CPLEX Dynamic Search: Incorporate CPLEX's dynamic search framework to enhance computational performance.
- Improve the state of the art: The proposed B&C was able to solve to optimality all 340 OPMVC benchmark instances. Additionally, it generated 69 new best-known solutions and improved 60 existing best-known solutions.
- Scientific dissemination and validation: Progress of this work were disseminated in the Latin-Iberoamerican Conference on Operations Research (CLAIO) in Guadalajara. A scientific article based on this work has been submitted and accepted. Pérez-Franco, M., Boyer, V., & Salazar-Aguilar, M. A. *The orienteering problem with mandatory visits and conflicts: A branch and cut approach.* Journal of the Operational Research Society TJOR. https://doi.org/10.1080/01605682.2025.253658. [forthcoming].

1.5 Methodology

1. Literature review: Examine the TSP, OP, relevant problem variants, and B&C solution techniques.
2. Identification and definition of the problem: Study the basic mathematical formulation of the OPMVC using a model based on directed graphs.
3. Reformulation of the problem: The problem is defined using an undirected graph with the aim of taking advantage of the symmetry of the model.
4. Identification of relevant cuts: Obtain information on cutting planes from similar problems in the literature.
5. Model implementation: Develop the problem formulation using an undirected graph representation.
6. Incorporation of Valid Inequalities: Introduce additional valid inequalities to strengthen the model.
7. Benchmark Validation: Solve benchmark instances to evaluate model performance.
8. Result Analysis: Assess computational performance and compare solutions with existing methods.

1.6 Thesis structure

The structure of this thesis is as follows. Chapter 2 provides a literature review introducing key concepts and previous research relevant to this study. Chapter 3 presents the formal description of the problem and the optimization model based on an undirected graph. Chapter 4 describes the B&C framework, including the primal heuristic, valid inequalities, and the separation algorithms. Chapter 5 discusses the experimental setup, computational results, and their interpretation. It also describes the hardware used, characteristics of the benchmark instances, and presents a comparative analysis through tables and graphs. Chapter 6 concludes the thesis by summarizing the main findings, reflecting on the research objectives, and providing directions for future research.

Chapter 2

Literature Review

This chapter serves as the foundation for the research by presenting previous research, concepts, and methodologies related to this study. Starting with the Traveling Salesman Problem (TSP) and covering its origin, historical development, and relevant formulations. Then, analyze how the TSP has been effectively solved through Branch and Cut (B&C) approaches. Next, the definition of the Orienteering Problem (OP) and its key characteristics, along with its contrast with TSP, are presented. Applicable cases are provided where the OP solutions have been obtained with exact B&C formulations. Then, variants of OP, which are structurally similar to the presented problem, are described. Finally, the OPMVC is introduced, containing a description of the problem definition, its connection to previous problems investigated, and relevant models, methods, and findings from the literature on which this thesis is based.

2.1 The Traveling Salesman Problem

In operations research and optimization, a routing problem is the process of finding the best way to move objects or resources from one place to another. The Traveling Salesman Problem (TSP) is a classic optimization problem that seeks to find the shortest route among a set of places, visiting each of them just once. The objective is to minimize the total distance traveled by a salesman between each pair of places while visiting all cities on the tour. The TSP was first investigated by William Rowam Hamilton and Thomas Penyngton Kirkman in the 1800s as stated in Matai *et al.* [2010]. Later, in 1930, the TSP was studied in Vienna by Menger and by Hassler, Whitney, and Merrill in Princeton, as reported in Cook *et al.* [1998].

It is a complex problem to solve due to its combinatorial nature. This is

because, with a number n of places to go, there are $1/2(n-1)!$ possible ways to visit them. Therefore, the more places exist, the more potential routes must be analyzed to choose the one with the shortest time.

One of the most significant works is the one of Dantzig *et al.* [1954]. This study aimed to visit 48 cities, one in each state of the USA, with Washington D.C. as the initial and final point. A linear programming approach using an undirected graph model was developed to achieve this. A highlight is the introduction of inequalities to avoid subtours in the solution. A subtour is a part of a route that forms a cycle of cities but does not include all the cities of the problem. Finally, this work solved the problem and found the optimal solution. Furthermore, it established a new benchmark in the field of the TSP.

The researchers continued exploring different optimization techniques to solve problems with more places. Grötschel and Holland [1985] implemented a cutting plane algorithm to solve to optimality a 1000-cities instance. However, it required 61 minutes and 9 seconds of computational time to solve it. Whereby, later in Grötschel and Holland [1991], authors employed polyhedral combinatorics to decrease the computational resources needed to solve the same instance. This study contributed to the state of the art by solving the TSP with exact methods using Linear Programming, cutting planes, heuristics, and the Branch and Bound (B&B) method. A polyhedral cutting planes method was proposed by Padberg and Rinaldi [1991]. Where each time a cut did not lead to an optimal solution, a strategy that kept producing cuts after branching was performed. The method was implemented in Fortran, and all tested instances were solved depending on the parameter and number of nearest neighbors (NN) on 1616, 849, 1016, and 1072 seconds.

Recent studies have worked on TSP problems by combining heuristics with exact methods. This combination is usually used to improve the initial solution quality. Research has also expanded to TSP variants, adding specific characteristics or constraints. In Nekovář *et al.* [2021], an interesting TSP variant is explored in the context of power transmission line inspections, where an inspection vehicle must make multiple trips to inspect all power lines due to travel constraints. The objective is to optimize the inspection routes and minimize operational costs. An Integer Linear Programming (ILP) and a Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic, which provides competitive solutions, were proposed.

In addition to exact methods, many algorithms, heuristics, and metaheuristics have been developed to find feasible solutions that are close to optimal. Some proposed techniques are the Ant Colony Optimization Algorithm (ACO), Genetic Algorithm (GA), Simulated Annealing (SA), and Particle Swarm Optimization Algorithm (PSO). Ha *et al.* [2020] focus on a variant of the TSP that works with drones

(TSP-D), with the objectives of minimizing the total operational cost and the time required for completion. They present a hybrid genetic algorithm search with dynamic population management and adaptive diversity control. This sophisticated search strategy ensures that the solutions remain diverse throughout the search and dynamically adjust the population. Their results improve many of the best-known solutions in the literature.

2.2 The Orienteering Problem

According to Feillet *et al.* [2005], the Orienteering Problem (OP) is one of the three profit-based problems of the TSP and was proposed by Tsiligirides [1984] and Golden *et al.* [1987]. The OP is also known as the Selective Traveling Salesman Problem (STSP) according to Laporte and Martello [1990]. It is of great interest due to its practical applications in the real world, such as in logistics and transportation, urban planning, telecommunications, supply chain, tourism, and many more. The OP is a routing problem whose goal is to determine which subset of places to visit and in which order, allowing to travel at most a certain distance or time and, for every visited place, a certain number of points to maximize the total collected score is determined.

To the best of our knowledge, the first contributions of exact methods for the OP were published in Laporte and Martello [1990], where an ILP model was proposed. Two versions of the algorithm were created. The first one relaxed the connectivity constraints, and the violated conditions were gradually added using a B&B method only when an integer solution was found. The connectivity constraints make sure that the selected vertices are in a single connected path. The second algorithm introduces the constraints as soon as it finds a group of connected vertices that are not part of a starting set. Later, in Ramesh *et al.* [1992], an algorithm was developed for solving a variant of OP, where the start and end points are the same. The algorithm applies Lagrangian relaxation through a spanning tree procedure in a B&B framework. It explores the characteristics of relaxation and includes strategies to improve performance. Results were obtained for instances with up to 150 places.

A B&C method was proposed in Gendreau *et al.* [1998a]. A variety of classes of valid inequalities were developed to improve the solution process, and then, they were integrated into the B&C algorithm. Also, two heuristics were created to have a construction process, followed by an optimization phase until no improvement can be found. The algorithm is capable of handling instances with up to 300 places. On the other hand, the B&C approach in Fischetti *et al.* [1998] includes exact and heuristic separation algorithms for the symmetric Selective Traveling Salesman

Problem (STSP). They use various types of valid inequalities and introduce several new classes like the conditional and cycle cover cuts. All the linear programs were solved in CPLEX, and the algorithm could optimally solve instances with up to 300 places. In Kobeaga *et al.* [2024], a B&C was developed. It was able to improve the values of the lower and upper bounds in the literature. One of the used techniques was the Integrated Constraint Handling, in which a separation algorithm simultaneously treats Subtour Elimination Constraints and Connectivity Constraints for the Cycle Problems. A variable pricing procedure was implemented to reduce the computational complexity, and a novel method for computing the global upper bound was used to improve the algorithm performance.

2.3 Variants of the Orienteering Problem

There are several variants of the OP, each with unique characteristics. The Team Orienteering Problem (TOP) is a combinatorial optimization problem in which multiple vehicles are used to maximize profit from visited places while operating under travel budgets. Every vehicle starts from a depot and visits a set of places. Every place has an associated score that can be collected if it is visited. In Xu *et al.* [2020], an approximation algorithm for the TOP is presented, whose approximation ratio is at least 0.32. This means that the algorithm guarantees a solution whose total profit is at least 32% of the optimal solution, even in the worst-case scenario. The results show that the benefits obtained by the proposed algorithms are approximately 12.5% to 17.5% higher than those of the existing algorithms.

On the other hand, the Orienteering Problem with Time Windows (OPTW) includes specific time windows during which places must be visited. The goal is to maximize the total score collected by visiting different locations within time constraints. The research of Gama and Fernandes [2021] seeks to solve this problem by applying neural network models trained through reinforcement learning. The results show that the proposed model offers better solutions than the traditional heuristics proposed for dynamic and varied environments.

In Lin and Vincent [2017], the Team OPTW and Mandatory Visits (TOPTW-MV) is studied. This variant involves multiple vehicles, and aims to maximize the total score in all routes, considering vehicle capacities and a limited number of vehicles. Additionally, it incorporates mandatory visits and time constraints. A MILP model and a Multi-Start Simulated Annealing (MSA) heuristic are proposed to solve the problem. The results showed that the MSA algorithm obtained better solutions than the commercial solver Gurobi in solving small instances of the benchmark.

The Time Dependent OPTW and Service Time Dependent Profits (TDOPTW-STP) is studied in Khodadadian *et al.* [2022], and has variations of travel times based on daily times. The objective is to maximize profits by having minimum and maximum service times or time windows and a given time budget. A MILP formulation is presented, as well as a Variable Neighborhood Search (VNS) metaheuristic. Experimental results show that the VNS algorithm provides high-quality real-time solutions. It also provides more efficient time budget management, especially in more congested traffic networks.

In the Clustered Orienteering Problem (COP) places are grouped into clusters with an associated score, which is collected only if all of the places in the cluster are served. The objective is to maximize total profits within a time limit. In Wu *et al.* [2024] an evolutionary algorithm is proposed to solve this problem. This algorithm combines a backbone-based crossover operator, a destroy-and-repair mutation operator for search diversification, and a solution-based tabu search procedure. The proposed algorithm outperforms the state-of-the-art algorithms on 924 benchmark instances from the literature, and it sets new lower bounds in 14 instances.

Uncertainty in travel times or scores is considered in the Stochastic Orienteering Problem (SOP).The objective is to plan a route that maximizes the score obtained by visiting certain places in a graph, respecting a travel budget on edges with stochastic costs and a defined probability of failure. Thayer and Carpin [2021] introduced an adaptive method that builds a path tree that allows exploring multiple sequences of places. This method increases the expected reward and controls the computation time.

Having reviewed the background and variations of the OP, this paper focuses on a variant of the OP, referred to as the Orienteering Problem with Mandatory Visits and Conflicts (OPMVC). This variant of the classic model enhances it with new constraints to capture conditions in the logistics and planning industry.

2.4 The Orienteering Problem with Mandatory Visits and Conflicts

The OPMVC was first studied in Palomo-Martínez *et al.* [2017b] and considers additional constraints, such as mandatory visits and incompatibilities among places. The mandatory places must be visited as a part of the solution tour. On the other hand, optional places are not required to be in the solution, and visiting them depends on time constraints. Every optional place has a positive score which is

collected if and only if the place is visited, while mandatory places have a score of zero. Conflicts between the places appear when pairs of places cannot be visited on the same route, forcing them to choose which place to visit. The OPMVC also has a time budget, and the solution must ensure that the total time spent traversing the edges between visited places does not exceed this limit. The goal of the OPMVC is to maximize the total score obtained from the visited optional places, while visiting the mandatory ones and respecting time limit and conflicts between places.

The first method for solving the OPMVC was developed by Palomo-Martínez *et al.* [2017b], who proposed a hybrid algorithm that combines a reactive Greedy Randomized Adaptive Search Procedure (GRASP) with a general Variable Neighborhood Search (VNS). The algorithm performs a multi-start strategy, in which several initial solutions are generated and then improved using the general VNS method. The process continues until it reaches a stopping condition.Computational experiments validate the algorithm's performance, and it found optimal solutions for 128 out of 340 instances.

Later, in Palomo-Martínez *et al.* [2017a], the authors adapt five MILP formulations derived from the TSP with emphasis on subtour elimination constraints. An exhaustive comparison using the commercial solver CPLEX showed that the best performance was obtained with the Gavish and Graves subtour elimination constraints by achieving 71% more optimal solutions than the other approaches. They observed that increasing the number of conflicts between nodes is beneficial in finding more optimal solutions within a reasonable time. This benefit occurs because adding more constraints reduces the number of possible solutions that meet all constraints. However, not all instances were solved to optimality within a time limit of 1h.

Similar problems have been studied before. In Salazar-Aguilar *et al.* [2014], a Multi-District Team Orienteering Problem was presented. It involves scheduling mandatory and optional tasks across multiple districts within a specified planning horizon and exclusionary constraints. The authors proposed a Mixed-Integer formulation and an Adaptive Large Neighborhood Search (ALNS) algorithm. A Memetic Algorithm (MA) for the Orienteering Problem with Mandatory Visits and Exclusionary Constraints (OPMVEC) was presented in Lu *et al.* [2018]. The study combines a dedicated Tabu Search (TS) that accommodates feasible and infeasible solutions through constraint relaxation, a backbone-based crossover, and a randomized mutation procedure to avoid premature convergence.

The OPMVC is an interesting problem to study because of its numerous practical applications. It aims to identify routes where particular places must be reached, while some places are optional, and some are impossible to visit on the same route

because of restrictions or conflicts. An interesting application is in package delivery logistics. Electric vehicle delivery modality has recently increased in countries such as China, the United States, Estonia, and Spain. This brings new challenges and, one in particular, is focused on battery life as studied in Martins *et al.* [2021]. Both OPMVC and STSP-TDP aim to optimize routes considering constraints. Both problems focus on finding the best sequence of places to visit or deliver to maximize a score collected under certain constraints, such as vehicle capacity or available time. In addition, both involve selecting deliveries or stops to minimize costs or times. However, the main difference is the nature of the constraints and the scores. The STSP-TDP introduces time-dependent benefits, allowing the score of visiting a place to vary throughout the day. On the other hand, the OPMVC focuses on optimizing routes under the constraint of battery life without considering temporal changes in benefits.

In the fuel delivery problem, the objective is to supply the required places while minimizing the distance traveled between them, as studied in Allen *et al.* [2024]. The fuel delivery problem and OPMVC share the structure of optimizing routes under limited resource constraints. In both cases, some places must be visited (mandatory), and others can be visited optionally, depending on operational constraints such as limited resources. The main difference is that OPMVC seeks to maximize a score based on the optional nodes visited. At the same time, the fuel delivery problem focuses more on route efficiency and delivery at urgent places.

The Unmanned Aerial Vehicle (UAV, Peyman *et al.* [2024]) mission planning is a problem that involves defining the route that a drone should take. The objective is to complete the mission safely to optimize resources such as the battery, to adapt to the conditions, and to comply with the defined restrictions. Both UAV mission planning and OPMVC aim to optimize routes under limited resource constraints, such as battery (in the case of drones) and maximum time in the OPMVC. Both require selecting a subset of places to visit. The main difference is that the UAV faces navigation challenges in three dimensions and dynamic environments. At the same time, the OPMVC is more static and relies on two-dimensional routes and conflicts between vertices.

Another application is tourism planning, which aims to design itineraries that maximize visitor satisfaction. It requires selecting specific places of interest that a traveler wants to visit, but it is impossible to visit all of them. The objective is to choose a set of places to be visited by a single route to maximize the benefit associated with the interest that each site offers to the traveler. This problem is known as Selective Traveling Salesman Problem with Time-Dependent Profits (STSP-TDP) and is studied in Barrena *et al.* [2023]. The OPMVC presents similarities since both seek to optimize an itinerary subject to specific restrictions. Both problems involve

planning a tour to maximize an objective function while meeting certain limitations. However, the key difference is that the score associated with the places in the STSP-TDP is variable and depends on the visit time to each point, while in the OPMVC, the score related to the optional vertices is static.

Table (2.1) shows a summary of the problems related to OPMVC and their key characteristics.

Problem	**Key Characteristics**	**Similarities with OPMVC**	**References**
Traveling Salesman Problem (TSP)	• Seeks the shortest route between places. • Visits every place only once. • Minimizes total distance.	Fundamental base of OPMVC. MILP formulations of OPMVC were adapted from TSP literature.	Matai *et al.* [2010], Dantzig *et al.* [1954], Grötschel and Holland [1985],
Orienteering Problem (OP)	• Determines subset of places to visit. • Maximizes total score. • Time/distance budget constraint.	OPMVC is a direct variant of OP. Also known as Selective Traveling Salesman Problem (STSP).	Tsiligirides [1984], Golden *et al.* [1987], Laporte and Martello [1990]
Team Orienteering Problem (TOP)	• Extension of OP with multiple vehicles. • Maximizes score of visited places. • Individual travel budgets.	Variant of OP. The difference is that in the OPMVC is a single route and in the TOP there are multiple vehicles.	Xu *et al.* [2020]
Orienteering Problem with Time Windows (OPTW)	• Extension of OP. • Specific time windows per place. • Visit must begin within defined interval.	Variant of OP. OPMVC has total time limit, but not specific windows per place.	Gama and Fernandes [2021
Team Orienteering Problem with Time Windows and Mandatory Visits (TOPTW-MV)	• Combines multiple vehicles. • Time windows. • Mandatory visits.	Shares mandatory visits with OPMVC, but includes time windows and multiple vehicles.	Lin and Vincent [2017]
Time-Dependent Orienteering Problem with Time Windows (TDOPTW-STP)	• Scores and times vary by hour. • Maximizes benefits with time budget. • Time windows and service times.	Variant of OP. The difference is that OPMVC has static scores and TDOPTW-STP dynamic scores.	Khodadadian *et al.* [2022]
Clustered Orienteering Problem (COP)	• Places grouped in clusters. • Score obtained if all places in the cluster are served.	Variant of OP. OPMVC focuses on conflicts between individual vertices, not grouping.	Wu *et al.* [2024]
Stochastic Orienteering Problem (SOP)	• Considers uncertainty in travel times. • Uncertainty in place scores.	Variant of OP. OPMVC is deterministic and without random variables.	Thayer and Carpin [2021]
Multi-District Team Orienteering Problem (MDTOP)	• Mandatory and optional activities. • Multiple districts. • Planning horizon. • Exclusion constraints.	Contains exclusion constraints or conflicts like the OPMVC, but adds multiple districts.	Salazar-Aguilar *et al.* [2014]
Selective Traveling Salesman Problem with Time-Dependent Profits (STSP-TDP)	• Maximizes score by visiting places • Time-dependent variable scoring • Optimizes itinerary with constraints	Similar to the OPMVC in maximizes score. The difference is that the STSP-TDP has dynamic scores.	Barrena *et al.* [2023]
Unmanned Aerial Vehicle Mission Planning (UAV)	• Defines route for drone missions. • Optimizes resources like battery. • Adapts to conditions and restrictions. • Three-dimensional navigation.	Both select subset of places to visit. The differences are that the UAV faces 3D navigation and dynamic environments.	Peyman *et al.* [2024]
Fuel Delivery Problem	• Minimize distance traveled. • Mandatory and optional places. • Limited resource constraints.	Both have mandatory and optional visit places. The difference is that the fuel delivery focuses on route efficiency and urgent deliveries.	Allen *et al.* [2024]
Orienteering Problem with Mandatory Visits and Conflicts (OPMVC)	• Mandatory vertices that must be visited. • Optional vertices with score. • Conflicts between vertices (incompatibilities). • Time budget constraint. • Objective: maximize total score.	Main study problem. Combines mandatory visits and conflict constraints	Palomo-Martínez *et al.* [2017a], Palomo-Martínez *et al.* [2017b], Lu *et al.* [2018]

Continued on next page

Table 2.1 – continued from previous page

Problem	Key Characteristics	Connection with OP-MVC	Study Title / Author(s)

Table 2.1: Comparison of optimization problems related to OPMVC

As illustrated, the OPMVC shares similarities in structure and constraints with numerous problems, such as those encountered in tourism planning, drone missions, and fuel distribution. Nevertheless, it has specific features, such as mandatory visits and conflicts between vertices. To this day and to the best of our knowledge, no exact method has been able to solve all of the instances of the OPMVC from the literature to optimality. Therefore, this work contributes by developing a B&C algorithm to solve all of them to optimality.

Chapter 3

Formal Definition and Mathematical Formulation

This chapter presents the formal definition of the OPMVC, a complex combinatorial optimization problem. The problem is formulated using an undirected graph, defining the parameters, decision variables, objective function, and key constraints that ensure the feasibility of the generated routes. Finally, to facilitate understanding, an illustrative example is shown.

3.1 Proposed OPMVC

The key optimization model related to this work is the one proposed by Palomo-Martínez *et al.* [2017a]. In that work, five mixed-integer linear formulations from the TSP literature were adapted and used to represent the OPMVC, all of them based on directed graphs. The Gavish and Graves (GG) subtour elimination constraints outperformed all the studied ones, and they are added to the model. The complete model is in Appendix (A).

In this work, with the aim of reducing the number of variables, we derived a new formulation based on an undirected graph. We considered the work of Fischetti *et al.* [1998]. Therefore, the notation is similar. The undirected graph is defined by $G = (V, E)$, where V represents the set of vertices, and $E = \{(i, j) \in V^2 \mid i < j\}$ is the set of edges. In this formulation, each vertex $i \in V$ is assigned a score s_i, representing the profit obtained by visiting it. Two special vertices, o and d, represent the route's origin and destination depot. These vertices are always included in the solution. The binary variable x_e shows whether edge $e \in E$ will be traversed in the solution or not, and the binary variable y_i indicates if the vertex $i \in V$ is visited.

The route must respect a global time constraint, meaning the total traversal time of the selected edges cannot exceed a maximum value denoted as T_{max}. Each edge $e \in E$ has an associated travel time t_e. The set $M \subset V$ includes all mandatory vertices, and it always includes o and d. The used notations are defined in Table (3.1).

Notation	Definition
o, d	o and d represent the starting and ending vertices or depots, respectively.
T_{max}	The maximum total duration allowed for the route.
t_e	The edge $e \in E$ travel time.
s_i	The score assigned to each vertex $i \in V$, with $s_i = 0$ for mandatory vertices ($i \in M$).
$M \subset V$	The set of mandatory vertices, which includes o and d.
$V \setminus M$	The set of optional vertices, which are not mandatory.
$C \subset V^2$	C is the set of conflicting vertex pairs, where if $(i, j) \in C$, vertices i and j cannot be on the same route.
$E(S) = \{(i, j) \in E \mid i \in S \text{ and } j \in S\}$	The set of edges between vertices within the subset $S \subset V$.
$\delta(S) = \{(i, j) \in E \mid i \in S \text{ and } j \notin S\}$	The set of edges where one vertex is in S and the other is outside S.
$E(i), \delta(i)$	If $S = \{i\}$, $i \in V$, we write $E(i)$ and $\delta(i)$, instead of $E(\{i\})$ and $\delta(\{i\})$.
$V(T) = \{i \in V \mid T \cap \delta(i) \neq \emptyset\}$	The set of vertices V that have at least one edge in the set $T \subset E$.
$x(T) = \sum_{e \in T} x_e$, for $T \subset E$	Sum of the values of the variables x_e associated with all the edges in the subset $T \subset E$.
$y(S) = \sum_{i \in S} y_i$, for $S \subset V$	Sum of the values of y_i for all elements i that are in the subset S of V.

Table 3.1: Definition of parameters for the proposed OPMVC formulation

Then, considering the decision variables.

- $x_e = \begin{cases} 1, & \text{if edge } e \in E \text{ is traversed} \\ 0, & \text{otherwise} \end{cases}$
- $y_i = \begin{cases} 1, & \text{if vertex } i \in V \text{ is visited} \\ 0, & \text{otherwise} \end{cases}$

The formulation is as follows.

$$Maximize\ z = \sum_{i \in V} s_i y_i \tag{3.1}$$

$$s.t.\ \sum_{e \in E} t_e x_e \leq T_{max}, \tag{3.2}$$

$$x(\delta(o)) = x(\delta(d)) = 1, \tag{3.3}$$

$$x(\delta(i)) = 2y_i, \qquad i \in V \setminus \{o, d\}, \tag{3.4}$$

$$x(E(S)) \leq |S| - 1, \qquad S \subset V, \tag{3.5}$$

$$y_i = 1, \qquad i \in M, \tag{3.6}$$

$$y_i + y_j \leq 1, \qquad (i, j) \in C, \tag{3.7}$$

$$x_e \in \{0, 1\}, \qquad e \in E, \tag{3.8}$$

$$y_i \in \{0, 1\}, \qquad i \in V. \tag{3.9}$$

The objective function (3.1) maximizes the sum of the scores related to vertex i and multiplied by binary variable y_i, $i \in V$. Constraint (3.2) restricts the maximum route length. Constraint (3.3) ensures that one connection leaves the initial depot and only one reaches the final depot. The degree constraints (3.4) ensure that when a vertex i is visited (y_i= 1), it must have two incident edges (incoming and outgoing); otherwise($y_i = 0$), it should have no incident edges. The subtour elimination constraints are (3.5). The constraints (3.6) force to visit all the mandatory vertices. Conflicts among vertices are avoided by constraints (3.7). Finally, constraints (3.8) and (3.9) define the binary nature of the decision variables.

3.2 Illustrative Example of the OPMVC

This section presents an example of the OPMVC. The example illustrates how the model operates with mandatory and optional vertices, travel times, and conflicts. It also illustrates how the objective function and constraints intersect to produce a feasible, high-scoring route.

There are given six vertices, where:

- Vertex O is the initial depot (o).
- Vertex D is the final depot (d).
- Vertex 3 is mandatory and must be visited.
- Vertices 1, 2, and 4 are optional vertices.

- Vertices 2 and 4 have conflict between them, so they cannot be visited on the same route.
- $T_{\max}$ is 15.

The objective is to maximize the score by visiting mandatory and optional vertices, ensuring the total time does not exceed $T_{\max}$. Figure (3.1) illustrates an example of the OPMVC problem with six vertices: an initial depot (O), a final depot (D), a mandatory vertex (3) that must be visited, and three optional vertices (1, 2, 4) with their positive associated scores that are the numbers in bold, for example, vertex number 1 has a score of 45, vertex 2 of 46, and vertex 4 of 49. Vertices 2 and 4 conflict, so they cannot be included in the same route. The figure shows the connections between vertices with their respective time. The total route time must not exceed $T_{\max} = 15$, all mandatory vertices must be visited, and the optimal solution must maximize the score collected from the visited optional vertices, avoiding conflicts.

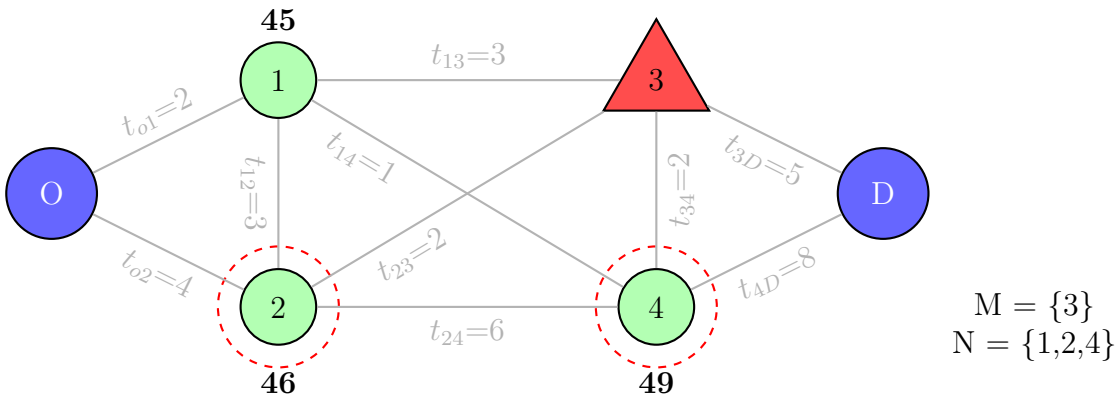

Figure 3.1: OPMVC example

When solving the example, we find that the best solution reaches a maximum score of 94 and it takes a 10 time units, within the allowed $T_{\max}$. This solution visits mandatory vertex 3, as well as the optional vertices 1 and 4, while avoiding the conflict between vertices 2 and 4. The optimal route is shown in Figure (3.2).

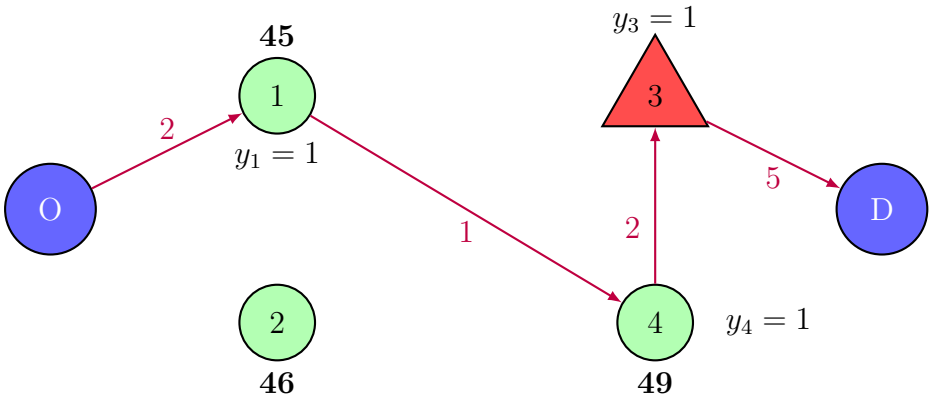

Figure 3.2: OPMVC solution

Chapter 4

Branch & Cut Algorithm

This chapter presents a detailed description of the proposed Branch-and-Cut method. First, a general flowchart illustrates the overall process of the method. Then, each step is described in detail followed by a primal heuristic which helps generate high-quality solutions in the early stages of the search. Following this, the valid inequalities are discussed. These constraints enhance the problem formulation and refine the feasible region. Finally, the separation algorithms used to identify and integrate these inequalities into the model are presented. Throughout this chapter, we systematically develop each of these components.

4.1 Branch and Cut Framework

Figure (4.1) summarizes the general Branch and Cut algorithm. It starts by solving the linear relaxation of the problem (step *LP*), this means we solve the problem without integer conditions and subtour elimination constraints. A solution X^* is derived from this relaxation, which can follow two paths depending on its nature. If X^* is integer and feasible, the algorithm terminates with an optimal solution (*Stop* step). Besides, if the solution is fractional, then the algorithm continues to the next step, which is the primal heuristic.

During the primal heuristic step (step *Primal heuristic* X^*), an attempt is to refine the fractional solution with the aim of getting a better approximation of the solution. The obtained fractional solution is then processed by the separation algorithms in the *Separation Algorithm*(X^*) step. Here, new constraints are identified (*New constraints found* step), and added to the LP (*Add constraints*). This process of adding constraints is continued until there are no more to add.

Then, the flow continues through the B&B process in search of the integer opti-

mal solution and terminates at the *Stop* step. The LP, with all the added constraints, is sent to the B&B phase (*LP + constraints to B&B*), where lazy constraints are used to eliminate integer solutions that are infeasible due to the presence of subtours.

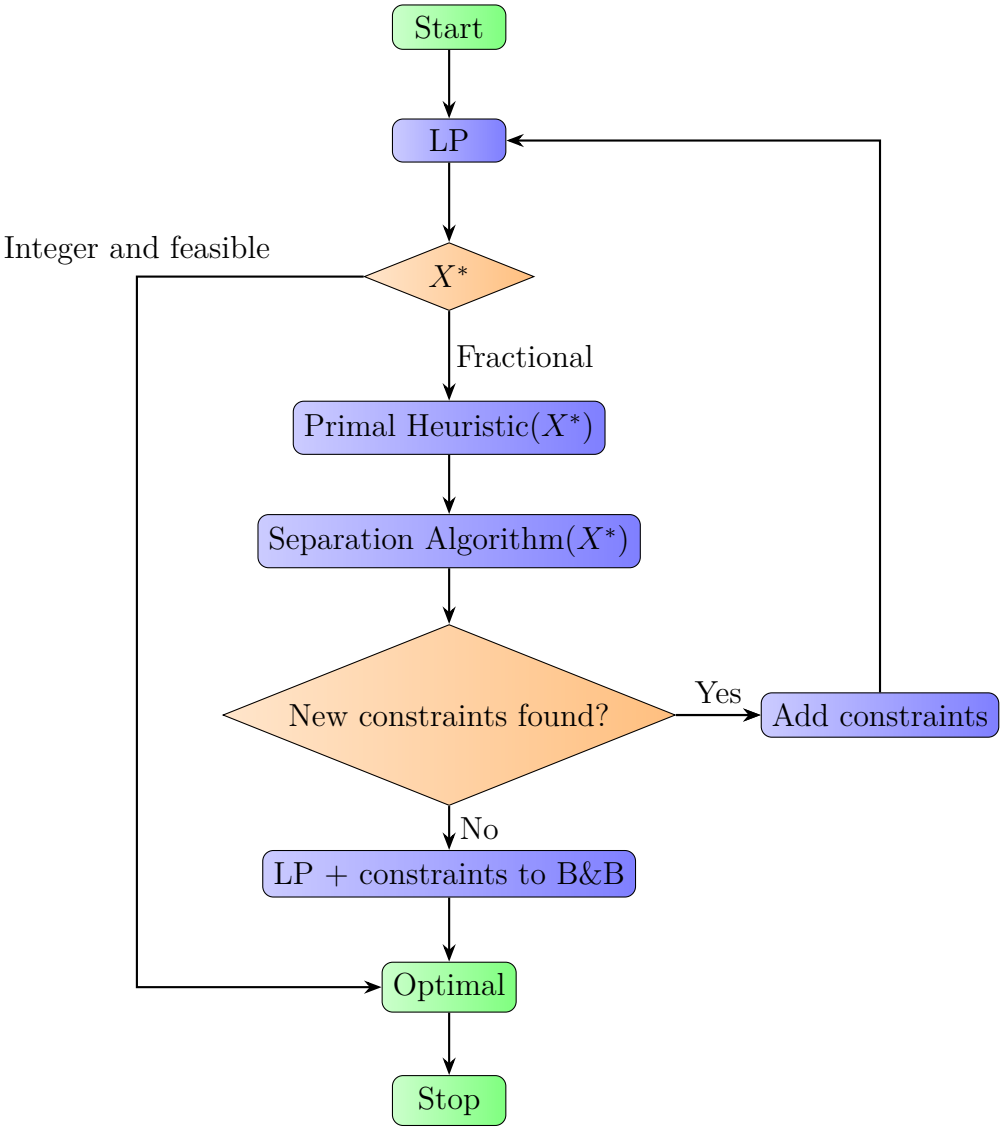

Figure 4.1: B&C Framework.

4.2 Valid inequalities

Valid inequalities are crucial in improving optimization models, specifically in B&C methods. Their objective is to refine the problem's linear relaxation, eliminating the infeasible solutions without excluding the feasible ones. The role of valid inequalities

is also to reduce the solution space, helping to converge faster to the optimal solution. In the B&C diagram process in Figure (4.1), valid inequalities are in the *Separation Algorithm* (X^*) step and added in *Add constraints*.

Valid inequalities are not generated arbitrarily. Separation algorithms are used to find the appropriate constraints. These algorithms are a set of techniques specifically designed to identify valid inequalities that are not present in the problem's initial linear relaxation. Depending on the nature of the problem and the type of inequality to be found, a particular separation algorithm is used. These algorithms operate at the step *Separation Algorithm* (X^*) in the B&C framework (Figure 4.1).

Each type of valid inequality used in this study is described below, along with the separation algorithm used to detect and incorporate them into the model.

4.2.1 Logical Constraints

These constraints impose basic assumptions derived directly from the logical structure of the problem. Their primary purpose is to guarantee that any valid solution must comply with these basic rules.

An example would be that if a vertex is part of a route, the vertex immediately preceding and following the vertex in the sequence must also be included in the path. Even though these constraints are often quite evident, their function is crucial in ensuring feasible solutions. These constraints quickly eliminate non-viable solutions early in the solving process. In mathematical terms, logical constraints are expressed as shown in expression (4.1). This mathematical expression implies that if $y_i = 0$ for some $i \in V$ then no edges can connect to vertex i, where $\delta(i)$ represents the set of edges incident to vertex i.

$$x_e \leq y_i, \quad i \in V,\ e \in \delta(i) \tag{4.1}$$

Although (4.2) relaxes the conditions of (4.1), its formulation ensures the validity of (4.1) in the practice of the separation algorithm.

$$x_e + x_{e'} \leq y_i + y_j, \qquad (i,j) \in C,\ e \in \delta(i),\ e' \in \delta(j) \tag{4.2}$$

Logical constraints are systematically reviewed to determine when they should be added to the problem formulation, ensuring that the generated solutions always respect the fundamental conditions of the logical structure.

Figure (4.2) illustrates the valid inequality (4.1) and shows a graph with vertices i, j, and k, where edges (i, j) ($x_{ij} = 0.5$) and (i, k) ($x_{ik} = 1$). This inequality ensures that if vertex i is not selected ($y_i = 0$), no edges incident to i can be activated ($x_{ij} = x_{ik} = 0$), thus eliminating infeasible solutions early in the *B&C* algorithm. For example, by setting $y_i = 0$, the graph is reduced to isolated vertices (j and k), demonstrating how these logical cuts restrict the search space to feasible solutions.

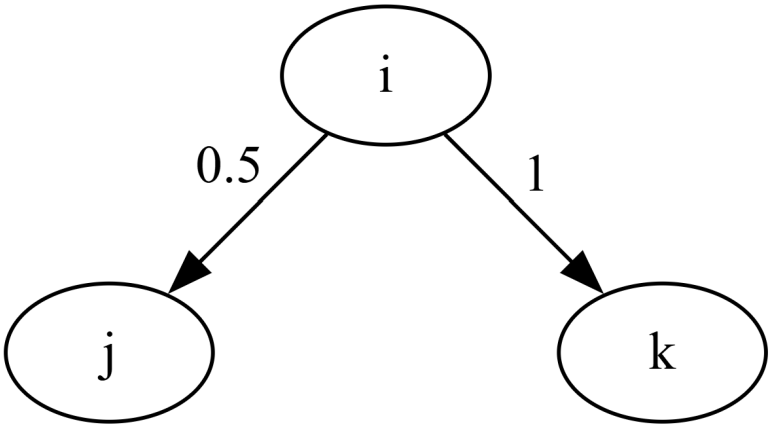

Figure 4.2: Logical inequality visualization

4.2.2 Matching Inequalities

Matching inequalities come from the concept of perfect matching in graph theory. In this context, a matching is a set of edges in a graph such that every vertex is connected to one other vertex through an edge, leaving no unmatched vertex. A perfect matching ensures that all vertices in the graph are connected, forming pairs of vertices with no isolated vertices.

The main goal of matching inequalities is to establish constraints that prevent invalid solutions, such as creating subtours. Subtours are small cycles within the solution that do not include all vertices; therefore, the conditions of the problem are violated. Matching inequalities are part of a family of restrictions called *Comb Inequalities*. Formally, they are defined as shown in expression (4.3).

$$x(E(H)) + x(T) \leq y(H) + \frac{|T| - 1}{2}, \qquad H \subset V,\ T \in \delta(H),\ |T|\ odd,\ \bigcup_{e \in T} V(e) = \emptyset \tag{4.3}$$

Here, H is the handle and represents a subset of vertices, while T is a set of edges called the teeth. Matching inequalities are used to enforce the linear relaxation of the optimization problem and prevent the appearance of infeasible solutions, such as the subtours mentioned above.

The impact of these inequalities on improving the problem solution has been widely demonstrated, not only in the original work of Fischetti *et al.* [1998], but also in more recent research, such as that of Kobeaga *et al.* [2024]. These investigations show how matching inequalities strengthen the problem formulation, accelerating convergence to the optimal solution.

Figure (4.3) shows an example of a comb inequality applied to a 48-vertices instance, where vertices 12, 15, and 32, represented by solid circles, form the *handle* H. The *teeth* T are the edges $T = \{(11, 12), (15, 39), (32, 45)\}$. Numerical values on the edges (such as 0.5, 0.190265, or 0.809735) correspond to the weights of the linear relaxation of the problem. The dashed lines indicate non-active edges in this specific inequality. In this example, the inequality (4.3) becomes $x(E(H)) + x(T) \leq y(H) + \frac{|T|-1}{2}$, where $x(E(H)) = 1.5$, $x(T) = 1 + 1 + 1 = 3$, and $y(H) = 3$. With $|T| = 3$, the inequality becomes $1.5 + 3 \leq 3 + \frac{3-1}{2}$, or $4.5 \leq 4$. Since $4.5 > 4$, this inequality is violated by the current fractional solution, demonstrating how matching inequalities can identify and eliminate non-optimal fractional solutions.

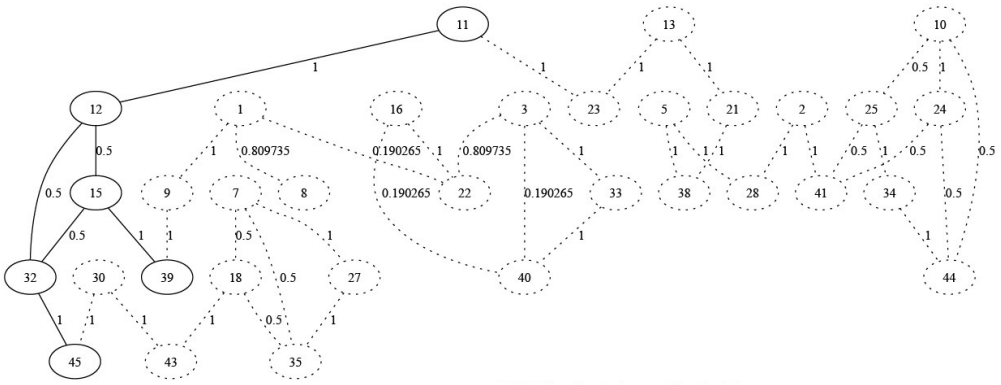

Figure 4.3: Matching inequality visualization

4.2.3 Infeasible Path Inequalities

Infeasible path inequalities are constraints employed in optimization to exclude paths or sequences of vertices from the solution set. These paths may be eliminated for violating particular problem constraints, like time and distance restrictions or other

logical criteria of the model. As a result, ignoring these invalid paths, the algorithm can be confident that it is focusing on feasible and potentially optimal solutions.

First, there are the edge cover inequalities, which exclude paths based on time constraints. If we have a subset $T \subset E$, where the sum of the travel time of all the edges is greater than the maximum allowed time t_{max}. The following constraint (4.4), proposed by Leifer and Rosenwein [1994], can be introduced.

$$x(T) \leq |T| - 1 \quad , \quad \sum_{e \in T} t_e > t_{\max} \tag{4.4}$$

Consider a scenario where you have a set of vertices V, and a subset $S \subset V$ that includes a pair of vertices (i, j) that are in conflict, i.e., $(i, j) \in C$, where C is the conflict set. Suppose these vertices are part of the same solution, meaning they are selected to be in the same path or route. According to the inequality (4.5), the number of edges in the subgraph induced by S, denoted $E(S)$, must satisfy the constraint $x(E(S)) \leq |S| - 2$. This means that in the subset S, the number of edges cannot exceed $|S| - 2$, where $|S|$ is the number of vertices in S.

To handle such cases, the following vertex conflict inequality is introduced (4.5).

$$x(E(S)) \leq |S| - 2, \quad S \subset V,\ S^2 \cap C \neq \emptyset \tag{4.5}$$

Where S is a subset of vertices and $S^2 \cap C \neq \emptyset$ i.e. there exists at least a pair of vertices in conflict in S.

This inequality is used to enforce the edge cover constraint because, when a conflict between vertices exists, it ensures that no possible path that goes through all the vertices in S can be formed.

Later, this process is taken one step further with profit cut inequalities. These inequalities are used when a path has a total profit less than a known lower bound $\underline{z}$. The path is considered infeasible because it would not provide enough value or profit to the optimization model. The following inequality, proposed by Gendreau *et al.* [1998a], expresses this idea in expression (4.6).

$$x(\delta(S)) \geq 2, \quad S \subset V,\ |S| \geq 2,\ \sum_{i \in S} s_i \leq \underline{z} \tag{4.6}$$

Where S is a subset of vertices and $\sum_{i \in S} s_i$ represents the total benefit associated with S. This inequality makes sure that those routes in which the total benefit is not greater than the lower bound $\underline{z}$ cannot be included.

To further strengthen this constraint, especially when the subset S contains source and destination vertices $o \in S$ and $d \in S$, the inequality can be modified to exclude all possible subpaths within S, as shown in expression (4.7).

$$x(E(S)) \leq y(S) - 2, \quad S \subset V,\ o \in S,\ d \in S,\ \sum_{i \in S} s_i \leq \underline{z} \tag{4.7}$$

This type of inequality is primarily beneficial to make sure there is no partial path in S that constitutes a valid solution, effectively excluding all subpaths that do not meet the minimum benefit requirements.

Figure (4.4) shows the concept of infeasible path cuts. A graph with solid vertices and weighted edges is shown on the left, where the red vertices (0 and 48) represent the initial and final depots. The numbers above the edges indicate the transition costs, while the values inside the vertices are the identification labels. On the right, a subgraph with dotted lines is shown representing infeasible paths according to the constraint (4.6), where the sum of the profits $\sum_{i \in S} s_i$ for these sets of vertices is less than the set lower bound $\underline{z}$. These paths are eliminated from the search space by adding cut inequalities that impose $x(\delta(S)) \geq 2$ for any set S that does not meet the minimum profit requirement.

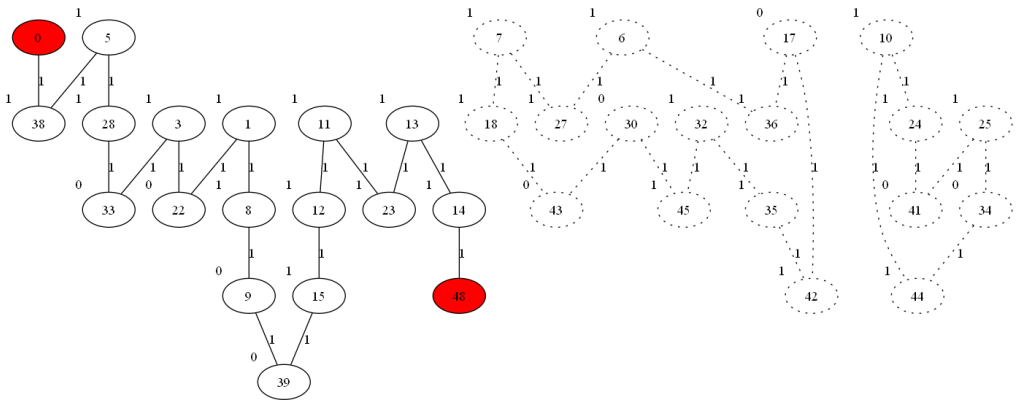

Figure 4.4: Infeasible path inequality visualization

4.2.4 Connectivity Inequalities

Connectivity inequalities are constraints that ensure that a single connected path will represent the optimal solution of an optimization problem. This is important to prevent the solution from being in disconnected subtours. In other words, they check constantly that each visited vertex is connected to another vertex through a path.

Gendreau *et al.* [1998a] presented these inequalities and are known as the Generalized Subtour Elimination Constraints (GSECs). The objective of GSECs is to make the solution a connected component. In the context of problems like the OPMVC, these constraints are essential to avoid the generated path that splits into many infeasible parts.

The connectivity inequalities for the OPMVC are presented by the expressions in Table (4.1).

Mathematical Expression	**Definition**
$x(\delta(S)) \geq 2y_i$ $S \subset V,\ o \in S,\ d \in S,$ (4.8) $i \in V \setminus S$	This inequality guarantees that if a vertex i outside S is included in the solution ($yi = 1$), there are at least two edges between S and the rest of the graph to maintain the route's feasibility.
$x(\delta(S)) \geq 2y_i,$ $S \subset V,\ o \in V \setminus S,$ (4.9) $d \in V \setminus S,\ i \in S$	This inequality ensures that the number of edges crossing the subset S is at least twice the variable associated with the vertices outside S.
$x(\delta(S)) \geq 2y_i + 2y_j - 2,$ $S \subset V,\ i \in S,$ $j \in V \setminus S,\ o \in S,\ d \in S$ (4.10)	This generalized form ensures that the edges of S are enough to maintain connectivity, also considering the relations between vertices i and j in the subsets S and $V \setminus S$. Proposed by Kobeaga *et al.* [2024].

Table 4.1: Connectivity inequalities

Constraint (4.11) ensures the connection between visited vertices, specifically when S has mandatory vertices. The Cycle Cover Cut was proposed by Fischetti *et al.* [1998] and its feasibility was proved by Gendreau *et al.* [1998b]. Its purpose is to remove solutions that do not meet connectivity given mandatory vertices, guaranteeing that the mandatory vertices are present in the route in a continuous and connected way.

$$x(E(S)) \leq y(S) - 1, \quad S \subset V,\ S \cap M \neq \emptyset \tag{4.11}$$

Figure (4.5) illustrates the application of the connectivity inequality (4.8). The red vertices (0 and 48) represent the source and destination depots located within

the set S. The white vertices (10, 25, 28, 33, 34, 41, and 44) are outside the set S (i.e., $i \in V \setminus S$). The gray vertices (5, 13, 21, and 38) are part of the set S. Vertex 28, highlighted with a double circle, represents a selected vertex in the solution, which triggers the constraint $x(\delta(S)) \geq 2y_{28}$. The black edges show connections that satisfy this restriction, ensuring that there are at least two edges between the set S and the rest of the graph when vertex 28 is included in the solution.

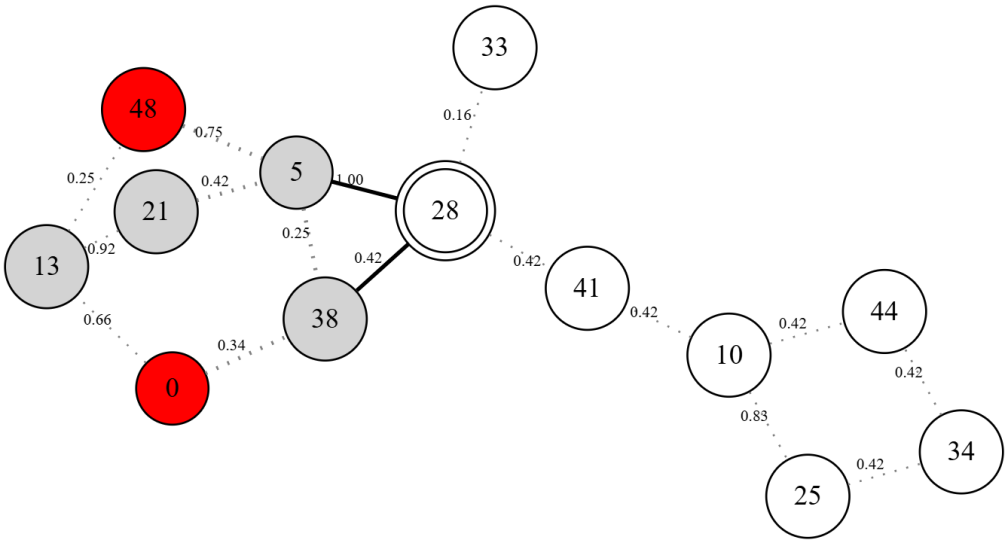

Figure 4.5: Connectivity inequality visualization

4.3 Separation Algorithms

4.3.1 Modified Kruskal Algorithm

The modified Kruskal Algorithm (4.3) seeks to identify matching inequalities from a fractional solution (x^*, y^*). Based on the support graph $G^* = (V^*, E^*)$, subsets of edges can be generated based on various thresholds θ defined on the set Θ. A non-cyclic forest is constructed for every threshold, which obeys a criterion resembling the classic Kruskal algorithm. Connected components are identified at each step, where adjacent edges with the highest fractional value are checked. When a large subset of edges T is found, a valid inequality of the form $x(E(H)) + x(T) \leq y(H) + \frac{|T|-1}{2}$ is obtained, and the model is enhanced by adding it as its constraint.

The modified Kruskal algorithm is adapted to identify and apply matching cuts efficiently, as described by Fischetti *et al.* [1998].

Let $G^* = (V^*, E^*)$ be the support graph derived from a fractional solution (x^*, y^*) where:

- $E^* = \{e \in E \mid x_e^* > 0\}$: Edges with nonzero fractional values.
- $V^* = V(E^*)$: Set of visited vertices.

Modified Kruskal Algorithm

Require: Graph $G^* = (V^*, E^*)$ associated with a fractional point (x^*, y^*) and a threshold set Θ

for $\theta \in \Theta$ **do**
  Set $G_\theta^* = (V^*, E_\theta^*)$ with $E_\theta^* = \{e \in E \mid x_e^* > \theta\}$
  Initialize an empty set $\mathcal{T}$
  Sort edges $e \in E_\theta^*$ in increasing order of x_e^*
  for $e \in E_\theta^*$ **do**
    if $\mathcal{T} \cup \{e\}$ does not form a cycle in $\mathcal{T}$ **then**
      Add e to $\mathcal{T}$
      Set H as the set of the connected components in $\mathcal{T}$ containing e
      Sort the edges $e \in \delta(H) = \{e_1, e_2, \ldots, e_{|H|}\}$ in decreasing order of x_e^*
      Find $T = \{e_1, e_2, \ldots, e_{|T|}\} \subset \delta(H)$ such that $|T| \geq 3$, and maximize $x_{e_1}^* + (x_{e_2}^* + x_{e_3}^* - 1) + \cdots + (x_{e_{|T|-1}}^* + x_{e_{|T|}}^* - 1)$
      for $e, f \in T^2$ with $e \neq f$ and $E(e) \cap E(f) \neq \emptyset$ **do**
        Remove e and f from T
        Remove $v \in E(e) \cap E(f)$ from H if $v \in H$, otherwise add v to H
      end for
      Add the cut $x(E(H)) + x(T) \leq y(H) + \frac{|T|-1}{2}$
    end if
  end for
end for

Algorithm 1: Modified Kruskal Algorithm

4.3.2 Random Path Construction Algorithm

Infeasible path inequalities are detected using a random path construction algorithm (2), which can find the routes that violate any previous restrictions. The algorithm is applied to a fractional solution (x^*, y^*). Starting from a vertex $s \in V^*$ and followed

by the selection of edges based on the fractional values x_e^*, a probabilistic path is constructed. At each iteration, a new adjacent edge that is not connected to depots o or d is selected; also, the total time accumulated so far is not greater than the allowed limit t_{max}. When no more edges can be added without violating the time constraint, the depots are joined to the built path in a way that minimizes the total path length.

Random Path Construction Algorithm

Require: Graph $G^* = (V^*, E^*)$ associated with a fractional point (x^*, y^*) and an initial vertex $s \in V^*$

$\forall e \in E$, set $p_e = \dfrac{x_e^*}{\sum_{f \in E} x_f^*}$

Select $e \in \delta(s) \setminus \delta(o) \cup \delta(d)$ with probability p_e

Set $\mathcal{P} \leftarrow \{e\}$

Set $E_{\mathcal{P}} = \{e \in \delta(i) \setminus \delta(o) \cup \delta(d) \mid i \in E(P) \text{ and } |\delta(i) \cap P| = 1\}$

while $E_{\mathcal{P}} \neq \emptyset$ and $\sum_{e \in \mathcal{P}} t_e \leq t_{max}$ **do**

  Select $e \in E_{\mathcal{P}}$ with probability p_e

  $\mathcal{P} \leftarrow \mathcal{P} \cup \{e\}$

  Update $E_{\mathcal{P}}$

end while

Add the depots o and d to $\mathcal{P}$ such that the length of $\mathcal{P}$ is minimized

return $\mathcal{P}$

Algorithm 2: Random Path Construction Algorithm

4.3.3 Max-Flow Algorithm

Connectivity inequalities are added to the model by solving subproblems with the Max-Flow Algorithm (3). This algorithm is used to determine the maximum amount of flow that can pass between vertices in a network, ensuring that there is connectivity, and eliminates solutions that lead to disconnected subtours.

Starting from the support graph $G^* = (V^*, E^*)$ associated with a fractional solution (x^*, y^*), the algorithm visits all the vertices $i \in V^*$ and solves a maximum flow problem originated at the vertex and terminating at the depots $\{o, d\}$. From the residual graph formed, the connected components containing vertex i and the depots are identified and denoted as S_i and S_{od}. When the total of the payoffs of vertices in S_{od} falls below a minimum level $\underline{z}$, a minimum inequality in the payoff is included.

Then, on each identified subset, several cases are considered: if it contains conflicting vertices, an inequality is added that limits the number of internal edges; whether at least one of the vertices of the subset is a mandatory vertex, a cut depending on the variable y is added; otherwise, a subtour removal inequality is added. Finally, to ensure that every visited vertex is correctly connected, an inequality is added that requires the number of incident edges to exceed 2.

```
Connectivity inequalities separation algorithm
Require: Graph G* = (V*, E*) associated with a fractional point
  (x*, y*) and a lower bound z
  for i ∈ V* do
     Solve the Max-Flow problem from i to {o, d} in G*
     Let G̃* be the final residual graph
     Set S_i as the set of connected components in G̃* containing
  i
     Set S_od as the set of connected components in G̃* containing
  o and d
     if Σ_{i∈S_od} s_i ≤ z then
        Add the cut x(E(S_od)) ≤ y(S_od) − 2  (4.7)
     end if
     for S ∈ {S_i, S_od} do
        if S contains at least two vertices with conflict then
           Add the cut x(E(S)) ≤ |S| − 2  (4.5)
        end if
        if S ∩ M ≠ ∅ then
           Add the cut x(E(S)) ≤ y(S) − 1  (4.11)
        else
           Add the cut x(E(S)) ≤ |S| − 1  (3.5)
        end if
        Add the cut x(δ(S)) ≥ 2y_i  (4.9)
     end for
  end for
```

Algorithm 3: Connectivity inequalities separation algorithm

4.4 Primal Heuristic

The primal heuristic is used to generate high-quality feasible solutions in the early stages of the search process. This heuristic has three main parts, as shown in

Figure (4.6): path construction, reparation, and improvement phase. The heuristic is used at the root vertex in each iteration of the cutting phase and whenever a new candidate solution is identified.

The approach starts with the random construction of a path, as described in Algorithm (2), using the fractional solution as input. When the initial route has been generated, the repair phase eliminates the conflicts or violations of the solution. At this stage, it identifies issues, whether the vertices selected have a conflict with other vertices or whether the time limit is exceeded. Conflicted vertices are ordered according to the value of their score, and vertices with the lowest scores are deleted. Then, if the total duration of the route is higher than the allowed maximum time, the optional vertices with the lowest score are removed from the route to reduce the total time until the route is feasible in terms of duration.

The final phase of the heuristic is the improvement phase, which seeks to increase the quality of the produced solution in the previous steps, ensuring that it does not violate any of the constraints. During this phase, mandatory vertices that are not included in the route are added. Mandatory vertices are selected from a group of candidates and are inserted into the route using two methods. The first is the best-insertion method that seeks to insert the vertex at the best possible position in the path. If it is impossible, the swap-insertion technique is used, where an optional vertex already in the path is swapped with the required mandatory vertex to be inserted. Then, the optional vertices try to be inserted using a best-insertion strategy to try to raise the value of the objective function. The vertex insertion is carried out only if it is within the available time limit.

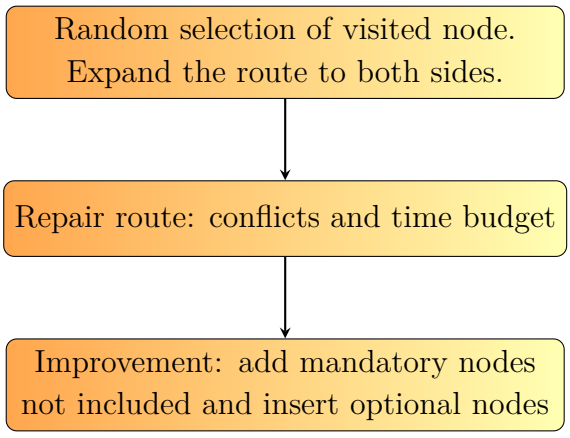

Figure 4.6: Primal Heuristic Diagram

Chapter 5

Experimentation and Results

This chapter presents the results of tests on benchmark instances from the literature. It first defines the experimental setup, detailing the solution process, hardware and software used, and instance characteristics. The results section compares the proposed B&C approach with the MILP model using Gavish and Graves (GG) subtour elimination constraints (Palomo-Martínez *et al.* [2017a]), and a Hybrid Variable Neighborhood Search (HVNS) by Palomo-Martínez *et al.* [2017b]. Key performance metrics include computation time, the number of optimal solutions found, best solutions improved, and cuts added.

5.1 Computational Experiments

A PowerShell script was used to execute the C++ implementation on 340 benchmark instances with a 3600-second time limit. The instances, sourced from Palomo-Martínez *et al.* [2017a], are classified into six groups based on Table (5.1). Each vertex may be incompatible with up to three others, and instance sizes range from 21 to 262 vertices.

The OPMVC instances are based on those of the OP and slightly modified to include the problem's characteristics. OP instances in the literature were created based on multiple problems. For example, an instance involves visiting one city in every state of the USA, starting and ending in Washington, aiming to find the shortest path. This modification involves randomly adding a percentage of mandatory vertices, introducing conflicts among optional vertices, and adjusting the distance limit.

The computational experiments were conducted on a HP Z230 workstation with an Intel Xeon(R) CPU E3-1245 v3 (3.40 GHz) processor and 16 GB of RAM, operating under Windows 10 Pro.

Class	Mandatory vertices	Free conflict vertices	Number of instances
1	10%	< 50	62
2	20%	< 50	55
3	30%	< 50	53
4	10%	> 50	62
5	20%	> 50	55
6	30%	> 50	53

Table 5.1: Characteristics of the benchmark instance classes

5.2 Results

The primary objective of the experiment is to compare the performance of the proposed B&C algorithm in solving the OPMVC on all benchmark instances from the literature. Specifically, we evaluate the algorithm's capacity to find optimal and feasible solutions, resolution times, and the number of improved solutions. Additionally, these metrics are analyzed at the root vertex of the B&C process. All the instances were run separately within a time limit of 1 hour for the B&C and HVNS methods and 3 hours for the GG method. To ensure fairness, the three approaches were executed in the same computational environment.

Table (5.2) presents the average feasibility and optimality rates achieved by the proposed B&C approach and benchmark methods. The B&C method consistently achieves 100% feasibility and optimality across all problem instances. In contrast, the GG method exhibits variability, with feasibility rates between 74.2% and 96.2% and optimality rates from 72.6% to 76.4%. The HVNS ensures full feasibility but shows significant variation in optimality, ranging from 45.3% to 90.6%. These results highlight the robustness of the B&C approach in consistently achieving both feasibility and optimality compared to alternative methods. The optimality rates of GG and HVNS are calculated in terms of the proportion of instances in which the methods yield the same optimal solution as the proposed B&C, which is presumed to be optimal.

Class	B&C		GG		HVNS	
	Feas	**Op**	**Feas**	**Op**	**Feas**	**Op**
1	**100.0%**	**100.0%**	77.4%	72.6%	**100.0%**	66.1%
2	**100.0%**	**100.0%**	89.1%	72.7%	**100.0%**	69.1%
3	**100.0%**	**100.0%**	**96.2%**	75.5%	**100.0%**	**90.6%**
4	**100.0%**	**100.0%**	74.2%	72.6%	**100.0%**	56.5%
5	**100.0%**	**100.0%**	81.8%	**76.4%**	**100.0%**	47.3%
6	**100.0%**	**100.0%**	79.2%	75.5%	**100.0%**	45.3%

Table 5.2: Average feasibility and optimality rates

Table (5.3) presents the average CPU times per instance class. The results indicate that the B&C method consistently requires less computational time than the GG method and, in certain cases, outperforms the HVNS. On average, B&C is approximately 90% faster than GG but about four times slower than the HVNS approach. Notably, B&C solves benchmark instances with up to 262 vertices in under five minutes. The table also highlights that for some classes, such as Class 1 and Class 3, B&C requires only a few seconds, whereas GG may take over an hour. The HVNS remains relatively efficient but is occasionally outperformed by B&C in terms of CPU time.

Class	B&C			GG			HVNS		
	min	**avg**	**max**	**min**	**avg**	**max**	**min**	**avg**	**max**
1	< 0.1	17.2	250.0	< 0.1	978.2	3600.0	< 0.1	4.4	37.0
2	< 0.1	4.6	94.0	< 0.1	751.0	3600.0	< 0.1	1.9	19.0
3	< 0.1	1.0	29.0	< 0.1	534.4	3600.0	< 0.1	0.5	5.6
4	< 0.1	**24.8**	284.0	< 0.1	1145.0	3600.0	< 0.1	**14.2**	**193.6**
5	< 0.1	11.1	156.0	< 0.1	970.1	3600.0	< 0.1	10.6	115.4
6	< 0.1	13.2	**286.0**	< 0.1	**1034.4**	3600.0	< 0.1	6.7	59.6

Table 5.3: CPU Times in seconds

Table (5.4) shows the number and the respective percentage of instances where the B&C method improved upon the solutions obtained by GG and the HVNS. Figure (5.1) plots the percentage of instances where the B&C method improved upon the solutions obtained by GG and the HVNS. The results show that algorithm B&C significantly improves over GG and the HVNS, with a particularly notable impact

in classes 4, 5, and 6. Compared to GG, B&C improves between 11.3% and 27.4% of the solutions, while compared to the HVNS, it achieves improvements greater than 31.0% in five of the six classes, reaching up to 54.7% in class 6. Overall, B&C outperforms both methods in 17.7% of cases for class 1, 12.7% for class 2, and 5.7% for class 3. Meanwhile, for the last classes, the percentage of improvement ranged between 20% and 25.8%. The fact that Class 3 has lower percentages of improvement does not necessarily mean that it is more challenging; on the contrary, many optimal solutions have already been found using the GG and HVNS methods, and B&C only completes the task of finding the few that were missing.

Class	**B&C vs GG**		**B&C vs HVNS**		**Overall**	
	#	%	#	%	#	%
1	14	22.6	21	37.9	11	17.7
2	8	14.5	17	31.0	7	12.7
3	6	11.3	5	9.4	3	5.7
4	17	27.4	27	43.5	16	25.8
5	11	20.0	29	52.7	11	20.0
6	13	24.5	29	54.7	13	24.5

Table 5.4: Number (#) and percentage (%) of solutions improved by B&C

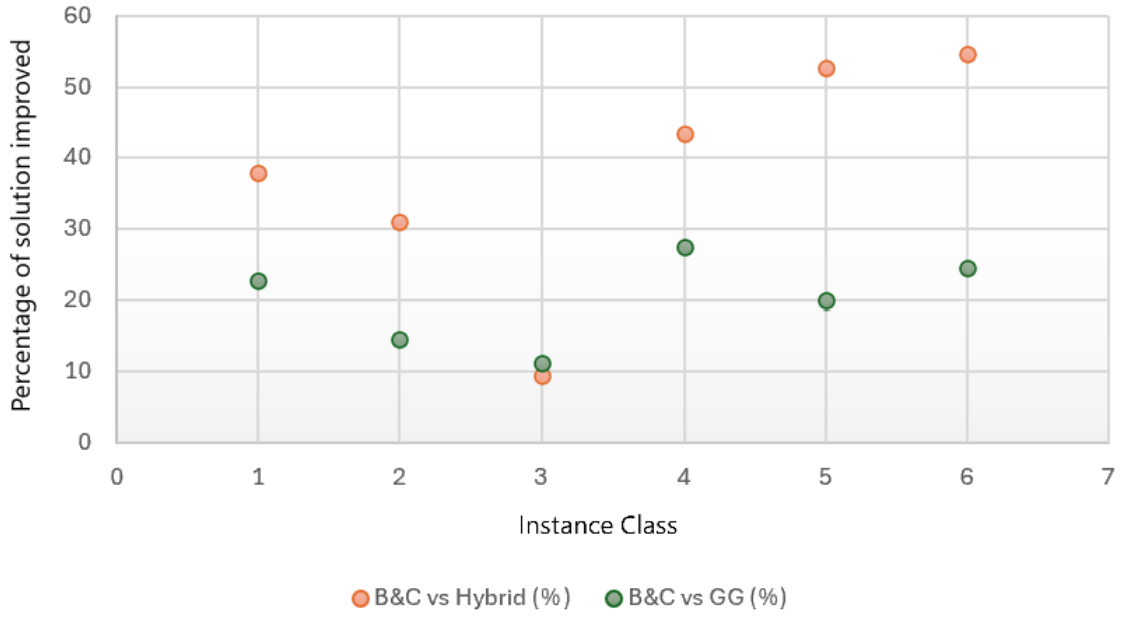

Figure 5.1: Comparison of B&C Improvement over GG and HVNS

5.2.1 Results at root vertex

In this experiment, we examine the performance at the root vertex. It demonstrates the effectiveness of the first relaxation and cutting step. This evaluation helps estimate how close the root vertex solution is to the optimal value in the end, as well as the effort required by the branching process.

Table (5.5) shows the percentage solution quality at the root vertex after the cutting phase. The results for the root vertex of classes 2, 3, and 6 are 100% feasible. Class 3 is the most efficient, with 84.9% optimality and an average CPU time of 0.5 s. CPU times are generally low, with averages ranging from 0.5 to 4.7 seconds. Values of up to 62 seconds are observed in class 1. The gap in class 4 is 29.5% and a maximum of 152%. It is calculated relative to the optimal solution obtained by the full B&C as shown in the Equation (5.1).

$$\text{Gap} = \left(\frac{Z_{\text{B\&C}} - Z_{\text{B\&C root}}}{Z_{\text{B\&C}}} \right) \times 100 \tag{5.1}$$

	Solution quality		**GAP (%)**			**CPU (s)**		
Class	**Feas**	**Op**	**min**	**avg**	**max**	**min**	**avg**	**max**
1	98.4%	29.0%	0.0	11.0	92.0	0.0	4.7	62.0
2	100.0%	56.4%	0.0	10.7	71.0	0.0	1.1	25.0
3	98.1%	84.9%	0.0	25.1	77.0	0.0	0.5	19.0
4	100.0%	6.5%	0.0	29.5	152.0	0.0	4.4	53.0
5	100.0%	16.4%	0.0	20.0	83.0	0.0	3.0	58.0
6	100.0%	18.9%	0.0	20.5	109.0	0.0	2.6	35.0

Table 5.5: Results obtained at the root vertex

Figure (5.2) shows the relationship in the literature instances regarding the percentage of mandatory vertices, the percentage of vertices in conflicts, and the amount of time it takes to solve them. To visualize and explore this relationship, a box plot was created showing the computational time it took for the B&C method to solve the instances of each class according to their characteristics. The figure shows that conflicts between vertices are essential to computational performance. Classes 1, 2, and 3 in orange have more than 50% of vertices in conflict and show low CPU times, from 1s to a maximum of 25s. This indicates that conflicts restrict the space of feasible solutions and accelerate solution resolution. On the other hand, classes 4, 5, and 6 in blue that have less than 50% of vertices in conflict show greater times, up to 60 s in class 5. This occurs when combined with 20 or 30% mandatory vertices. This

highlights that, while conflicts simplify the problem by reducing options, mandatory vertices only have a significant impact when conflicts are minimal.

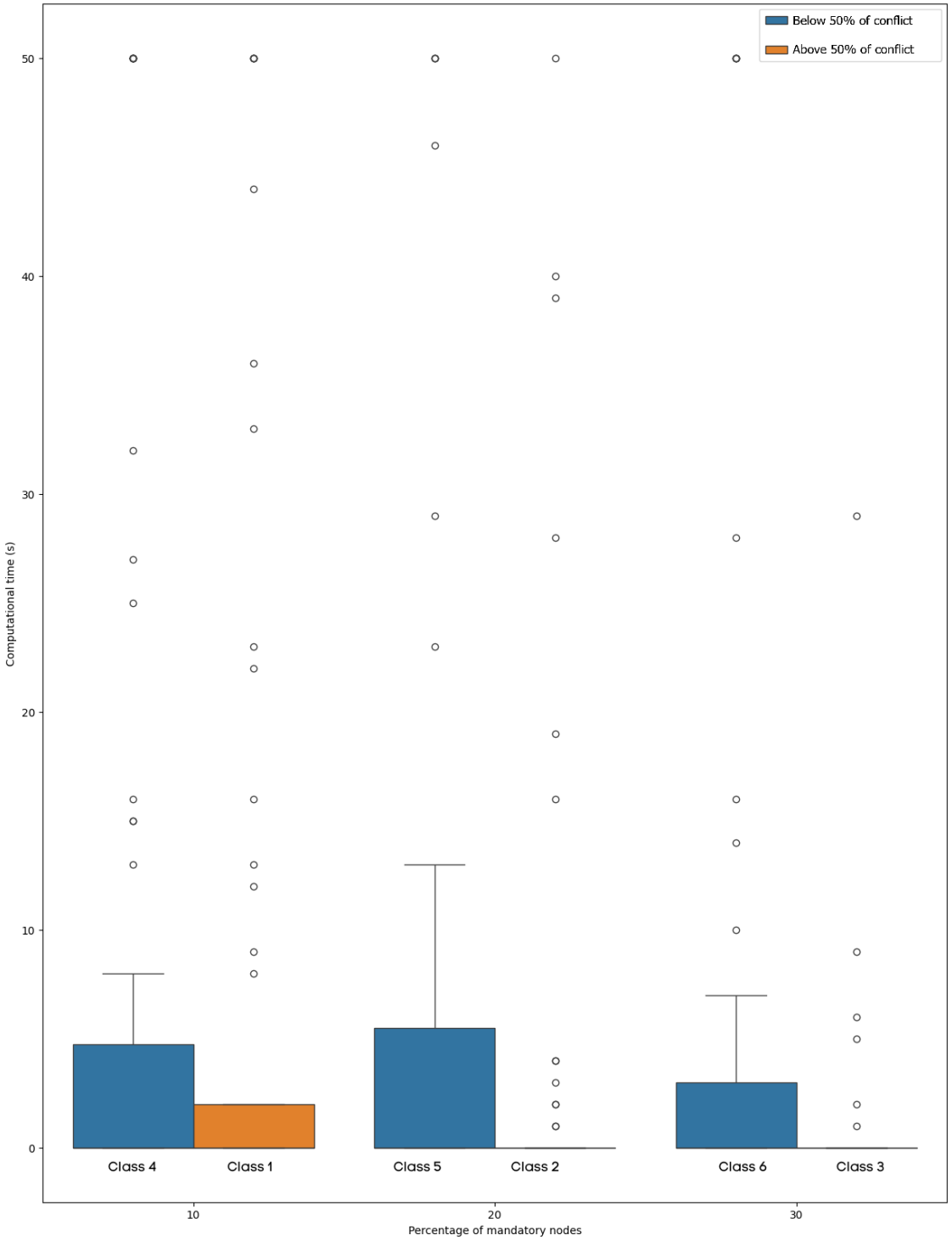

Figure 5.2: Relation between percentage of mandatory vertices and conflict free vertices

5.3 Primal Heuristic Performance

We also performed the experiment of solving all instances at the root vertex. We evaluate the results obtained by the B&C method in the root vertex and compare them with those from the complete HVNS method, the analysis was conducted on the 340 instances. Table (5.6) presents a summary of the percentage of instances for which feasible and optimal solutions were obtained by each method, categorized by class. It also shows the average time each method took to reach the solution. It is important to note that for two instances, one in class 1 and another in class 3, the B&C method fails to find a solution because the heuristic does not identify a feasible solution that satisfies the mandatory vertex constraints. Lastly, the average gap between the objective function results of HVNS and B&C at the root vertex is shown in the last column. This average gap is based on instances where both methods found a solution and it is calculated as shown in the Equation (5.2).

$$\text{Gap} = \left(\frac{Z_{HVNS} - Z_{B\&C\,root}}{Z_{HVNS}} \right) \times 100 \tag{5.2}$$

Class	B&C root vertex			HVNS			Gap
	Feas	Op	Avg time	Feas	Op	Avg time	
1	98.4%	29.0%	4.8	100.0%	66.1%	4.4	10.0%
2	100.0%	56.4%	1.1	100.0%	69.1%	1.9	5.0%
3	98.1%	84.9%	0.5	100.0%	90.6%	0.5	1.2%
4	100.0%	6.5%	4.4	100.0%	56.5%	14.2	1.3%
5	100.0%	16.4%	3.0	100.0%	47.3%	10.6	9.0%
6	100.0%	18.9%	2.1	100.0%	45.3%	6.7	8.0%

Table 5.6: Average Feasible and Optimal Solutions comparing B&C and HVNS

The results show that, while both methods yield feasible solutions in nearly all cases, HVNS consistently outperforms B&C in terms of finding optimal solutions. For most problem classes, HVNS achieves a higher percentage of optimal solutions, especially in classes 4, 5, and 6, where the gap between the two methods is most pronounced. This suggests that HVNS is more effective in exploring the solution space and identifying optimal solutions, while B&C with the root vertex limit tends to find good feasible solutions faster but misses the optimal ones in some cases.

The gap in optimal solutions between the two methods is relatively small in classes 2 and 3, where both B&C and HVNS achieve similar results in terms of optimality.

HVNS consistently has slightly higher average CPU times compared to B&C, but its maximum CPU times can be significantly larger, particularly in classes 4, 5, and 6, as shown in Table (5.7). For example, the maximum CPU time for HVNS in class 4 is 193.6 seconds, which is notably higher than the maximum for B&C in the same class (53 seconds).

Overall, B&C is more efficient in terms of CPU time, particularly for simpler problem classes (1-3), where it requires very little computational effort. HVNS, while requiring more time, may provide better-quality solutions, as seen in the previous analysis of optimality. This trade-off between computation time and solution quality makes B&C a suitable choice for problems where speed is crucial, while HVNS may be preferred when higher-quality solutions are needed, despite the increased computational cost.

Class	B&C root vertex			HVNS		
	min	avg	max	min	avg	max
1	0	4.8	62	<0.1	4.4	37.0
2	0	1.1	25	<0.1	1.9	19.0
3	0	0.5	19	<0.1	0.5	5.6
4	0	4.4	53	<0.1	14.2	193.6
5	0	3.0	58	<0.1	10.6	115.4
6	0	2.1	24	<0.1	6.7	59.6

Table 5.7: CPU Times (in seconds) comparing B&C and HVNS

5.3.1 Families of cuts

Experiments were conducted where only one cut family was tested and the other three were not included. Also, one family was removed, and three were left in the tests. Two variations of the experiment will be conducted. The first variation involves the value in the root vertex and the second variation involves running the full B&C procedure. The families of cuts are considered as follows:

1. Logical Cuts: (4.1) and (4.2).
2. Matching Cuts: (4.3).
3. Infeasible Path Cuts: (4.5) and (4.7).

4. Connectivity Cuts: (4.8), (4.9) and (4.11).

The condensed time and gap averages of the 6 classes are shown in Table (5.8) and contain the experiments on the 340 literature instances. The labels *W/* and *W/o* indicate with or without, respectively, and are used to show the inclusion or exclusion of a particular family of cuts in each experiment. The gap reported corresponds to the value provided by the solver. The complete results are in the Appendix C.

Method	**Root Vertex**		**B&C**	
	Avg Time (s)	**Avg Gap (%)**	**Avg Time (s)**	**Avg Gap (%)**
W/ All Cuts	2.679	18.621	11.744	0.000
W/ Logical Cuts	2.015	32.403	690.415	3.115
W/ Matching Cuts	1.856	33.468	696.232	3.129
W/ Infeasible Path Cuts	2.312	27.887	374.885	1.412
W/ Connectivity Cuts	2.941	17.563	10.053	0.000
W/o Logical Cuts	2.544	18.080	9.776	0.000
W/o Matching Cuts	3.147	18.242	9.491	0.000
W/o Infeasible Path Cuts	2.900	18.835	10.544	0.000
W/o Connectivity Cuts	2.800	25.680	365.617	1.432

Table 5.8: Average time and gap in Root Vertex and B&C of family cuts tests

Matching Cuts are the fastest in the initial root vertex phase at 1.856 seconds, although they have the most significant gap of 33.5%. On the other hand, Connectivity Cuts and the method with all active cuts achieve the smallest gaps of 17.6% and 18.6%, respectively, but with longer times of 2.941 and 2.679 seconds. This suggests that Logical or Matching Cuts speed up the initial solution at the expense of less optimal solutions.

When analyzing the full B&C, the results change. Methods that include Matching or Logical Cuts have execution times of approximately 700 seconds despite maintaining relatively small gaps of 3.1%. In contrast, turning off these cuts reduces the time to less than 10 seconds with a gap of 0.0%. It is essential to highlight the impact of Connectivity Cuts: when disabling them, the time increases to 365.617 seconds, increasing the gap and no longer being 0.0%. This proves that they are essential for the efficiency of the algorithm.

To further analyze the impact of each cut family, the results from Table (5.8) were then plotted by number of vertices. Figure (5.3) shows the CPU time as a function of the number of vertices for the B&C method testing only one cut family. All variants maintain low and uniform processing times for small to medium-sized

instances of 21–102 vertices, with some outliers that do not exceed 600 seconds. However, an inflection point is observed from 122 vertices, where tests with Matching, Infeasible Path, Logical, and Connectivity cuts increase in computational time, reaching the maximum limit of 3600 seconds for instances of 152 vertices and above. This contrasts with the full B&C implementation, which maintains low times even on the largest instances, reaching 200 seconds with 263 vertices. On vertices with 77 and 102 vertices, outliers are distributed up to 1600 seconds, while for the basic B&C implementation, they remain below 400 seconds. This suggests that the performance of algorithms with additional cuts may depend on the specific characteristics of each instance.

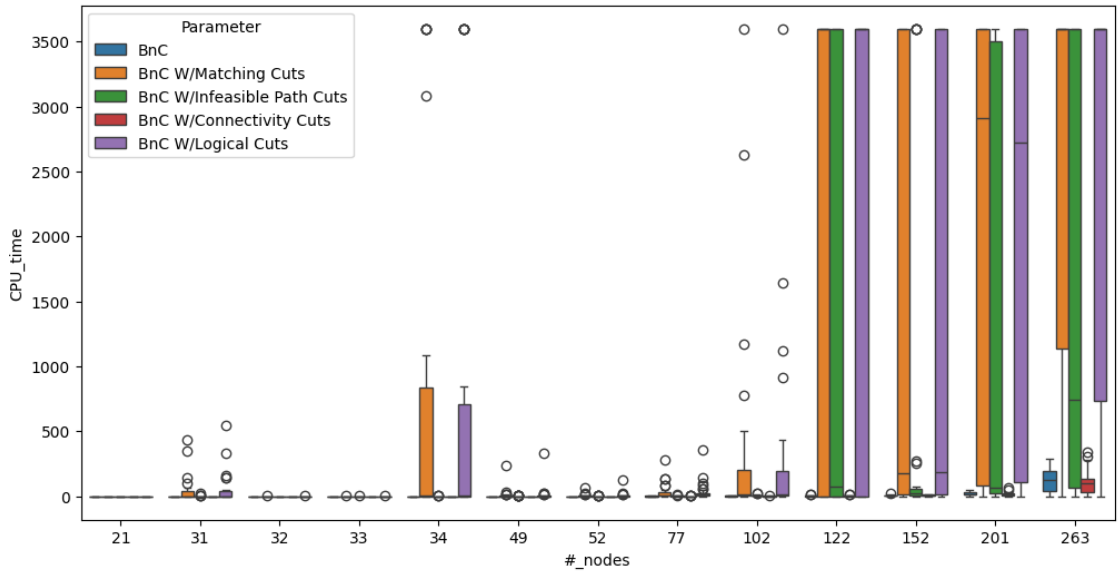

Figure 5.3: CPU time by number of vertices in B&C method with a single family

5.4 Comparison of Results with a Memetic Algorithm (MA)

The current section provides a comparative overview of the results obtained in this study and those presented by Lu *et al.* [2018], which used the same set of 340 benchmark instances to solve the same problem. Lu *et al.* suggested a solution approach based on a Memetic Algorithm (MA). Additionally, their study investigates a mixed approach that combines the MA and the Integer Programming (IP) model by Gavish and Graves (OPMVEC-GG), where the MA is used to produce an initial solution to the CPLEX solver.

Since the experiments were performed in different computational environments and there is no access to the source codes used by Lu *et al.*, it is not possible

to make a fair comparison of the reported CPU times between our exact B&C method, standalone MA, and the hybrid MA+OPMVEC-GG approach. We decided not to make that comparison and focus the analysis on the quality of the solutions obtained, through the percentage of optimal solutions found with each method.

Table (5.9) shows the percentage of the optimal solutions found by each method per classes. B&C method has 100 percent optimality in all the six classes. In comparision, the heuristic-based approaches have a lower rates of optimality. It is important to notice that the hybrid variant is better than the standalone MA in each class with the exception of Class 3, in which they perform equally well.

Class	**B&C (%)**	**MA (%)**	**MA + OPMVEC-GG (%)**
1	100.0	85.5	85.5
2	100.0	92.7	94.6
3	100.0	98.1	98.1
4	100.0	77.4	80.6
5	100.0	81.8	85.5
6	100.0	79.2	84.9

Table 5.9: Percentage of optimal solutions in each class

Chapter 6

Conclusion

This thesis introduces a novel Branch-and-Cut approach for solving the Orienteering Problem with Mandatory Visits and Conflicts (OPMVC), outperforming traditional Mixed Integer Linear Programming (MILP) methods by achieving optimal solutions for all the literature benchmark instances. Using undirected graphs, the proposed methodology reduces computational complexity and accelerates convergence, setting a new benchmark in OPMVC optimization. In particular, the B&C approach consistently achieves 100% feasibility and optimality across all benchmark instances, contributes with 69 new optimal solutions, and improves 60 previously reported best-known solutions.

The performance of the proposed B&C was evaluated by comparing the results obtained with the best approaches from the literature, i.e. a MILP approach based on a directed graph formulation and a Hybrid Variable Neighborhood Search. It is approximately 90% faster on average than the MILP approach. Although the hybrid VNS was up to four times faster than B&C, B&C solves instances of up to 262 vertices in less than five minutes to optimality.

The analysis of the experimental results also provides valuable insights into the influence of instance characteristics. Mandatory vertices significantly impacted time only when vertex conflicts were minimal (Classes 4, 5, and 6 with 20% or 30% mandatory vertices). The tests with the cut families reveal that connectivity inequalities are essential for the algorithm's efficiency and for achieving a 0% optimality gap. Our results show that all cuts in the proposed B&C framework were key to achieving optimality in all instances.

In summary, this work develops and validates a robust and scalable B&C approach that successfully solved all OPMVC benchmark instances available in the literature to optimality within a reasonable time limit. The undirected graph-based formulation, the integration of relevant valid inequalities, and the development of an effective separation method establish a new benchmark for OPMVC optimization,

marking a substantial advance in the state-of-the-art for this complex combinatorial problem.

6.1 Future work

Future research should focus on developing a new benchmark that includes larger instances with a higher number of vertices. This would enable a thorough assessment of the proposed method scalability and robustness when addressing more complex and realistic challenges.

A key aspect to explore is the impact of conflict density on solution quality. Understanding this relationship could lead to refinements in the mathematical formulation and enhancements in the search strategy. Additionally, incorporating alternative heuristic algorithms may improve initial solution quality, accelerating convergence within the optimization process.

Another promising direction is the implementation of parallel computing techniques to enhance computational efficiency. Given the intensive nature of graph optimization, parallelizing the algorithm could exploit modern hardware architectures to significantly reduce execution times, particularly for large-scale instances.

Finally, investigating Branch and Price as an alternative approach for solving large instances could provide a more effective way to manage problem complexity. By dynamically generating only the most relevant columns, this technique may improve solution quality while maintaining computational feasibility.

Appendix A

Gavish and Graves Model for the OPMVC

This appendix presents the complete mathematical formulation for the OPMVC based on the model proposed by Palomo-Martínez *et al.* [2017a]. This model uses a directed graph $G = (N, A)$, with A is a set of directed arcs and N is a set of nodes. It incorporates the subtour elimination constraints based on the ones by Gavish and Graves Gavish and Graves [1978]. The formulation is shown below.

$$x_{ij} = \begin{cases} 1 & \text{If node j is visited after visiting node i, (i,j)} \in \text{A;} \\ 0 & \text{otherwise.} \end{cases} \tag{A.1}$$

$$y_i = \begin{cases} 1 & \text{If the i node is visited, i} \in \text{N;} \\ 0 & \text{otherwise.} \end{cases} \tag{A.2}$$

The model with the subtour elimination constraints is:

$$Maximize\ z = \sum_{i \in N} s_i y_i \tag{A.3}$$

$$s.t. \sum_{i \in N:(1,i) \in A} x_{1i} = 1, \tag{A.4}$$

$$\sum_{i \in N:(i,n) \in A} x_{in} = 1, \tag{A.5}$$

$$y_k = 1, \qquad k \in M, \tag{A.6}$$

$$\sum_{j \in N:(j,i) \in A} x_{ji} = y_i, \qquad \text{i} \in N \backslash \{1\}, \tag{A.7}$$

$$\sum_{j \in N:(i,j) \in A} x_{ij} = y_i, \qquad \text{i} \in N \backslash \{n\}, \tag{A.8}$$

$$y_i + y_j \leq 1, \qquad i \in N, \text{j} \in C_i, C_i \neq \emptyset, \tag{A.9}$$

$$\sum_{(i,j) \in A} t_{ij} x_{ij} \leq T_{max} \tag{A.10}$$

$$0 \leq g_{ij} \leq (n-1)x_{ij}, \qquad (i,j) \in A,\ j \neq 1, \tag{A.11}$$

$$\sum_{j=1}^{n-1} g_{ji} - \sum_{j=2}^{n-1} g_{ij} = y_i, \qquad i \in N \setminus \{1, n\} \tag{A.12}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in A, \tag{A.13}$$

$$y_i \in \{0,1\}, \qquad e(i) \in N, \tag{A.14}$$

Where:

- s_i is the score obtained from visiting node i.
- $C_i \subseteq V$ is the set of nodes in conflict with node i.
- t_{ij} is the travel time required to go from node i to node j.
- T_{max} is the maximum total time allowed for the route.
- g_{ij} is the flow of a single commodity traversing arc (i,j).

Constraints (A.4) and (A.5) guarantee that the route starts at node 1 and ends at node n. Constraints (A.6) stipulate that no mandatory node can be left out of the route. Constraints (A.7) and (A.8) set the flow conservation constraint. Constraints (A.9) ensure no conflicts between the nodes in the route. Constraints (A.10) guarantee the total time taken to complete the route is within the allowed time. The (A.13) and (A.14) constraints are related to the nature of the decision variables.
The subtour elimination constraints (A.11) and (A.12) are based on the single commodity-flow model of Gavish and Graves [1978], let g_{ij} be the flow of a single commodity traversing arc (i,j). They ensure that $n-1$ flow units leave the source node and that each visited node consumes only one flow unit. If the variable g_{ij} is greater than zero, it can be considered as the number of arcs from node j to the destination node on the optimal tour. Expression (A.12) ensures that a node consumes a flow unit only if it is visited.

Appendix B

Detailed results

This appendix presents detailed results for each instance of the six classes using our B&C method, the MILP GG method, and the HVNS. The B&C and HVNS methods took a maximum of 1 hour, while the GG took a maximum of 3 hours

Each table corresponds to each of the six classes and is specified in every description. The first column has the name of the instance. Then, there are three sections that correspond to each method, each with three columns. Column z is the value of the objective function achieved. The Gap column is the percentage difference between the obtained solution and the optimal value. Finally, the CPU column indicates the computation time, in seconds, that the method took to resolve the instance.

Table B.1: Results for Class I

Inst.	**B&C**			**GG**			**HVNS**	
	z	**Gap(%)**	**CPU(s)**	z	**Gap(%)**	**CPU(s)**	z	**CPU(s)**
att48A	15	0.00	0.00	15	0.00	12.00	15	0.38
att48B	17	0.00	0.00	17	0.00	6.00	17	0.19
att48C	10	0.00	0.00	10	0.00	65.00	8	0.12
att48D	13	0.00	0.00	13	0.00	7.00	13	0.24
att48E	15	0.00	0.00	15	0.00	5.00	14	0.26
cmt121A	535	0.00	0.00	535	0.00	2078.00	535	1.24
cmt121B	416	0.00	16.00	396	0.29	10799.00	416	3.92
cmt121C	501	0.00	13.00	497	0.05	10799.00	498	2.76
cmt121D	530	0.00	2.00	530	0.00	760.00	530	1.42
cmt151A	815	0.00	8.00	815	0.00	6593.00	807	8.83
cmt151B	872	0.00	0.00	872	0.00	2096.00	872	2.05
cmt151C	435	0.00	12.00	435	0.00	1538.00	432	4.59
cmt151D	571	0.00	22.00	571	0.01	10799.00	568	8.74
cmt151E	673	0.00	9.00	673	0.00	1379.00	666	6.20

Continued on next page

Table B.1 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
cmt200A	598	0.00	33.00	n/a	n/a	10799.00	591	9.69
cmt200B	1352	0.00	1.00	n/a	n/a	10799.00	1352	6.10
cmt200C	751	0.00	23.00	n/a	n/a	10799.00	750	16.72
cmt200D	919	0.00	44.00	n/a	n/a	10799.00	902	15.64
cmt200E	1028	0.00	36.00	n/a	n/a	10799.00	1008	20.13
eil30A	6375	0.00	0.00	6375	0.00	0.00	6375	0.09
eil30B	5125	0.00	0.00	5125	0.00	5.00	5125	0.07
eil30C	5775	0.00	0.00	5775	0.00	4.00	5775	0.10
eil30D	6275	0.00	0.00	6275	0.00	14.00	6275	0.10
eil33A	5230	0.00	0.00	5230	0.00	8.00	5230	0.13
eil33B	14380	0.00	0.00	14380	0.00	2.00	14380	0.12
eil33C	7430	0.00	0.00	7430	0.00	13.00	7430	0.14
eil33D	11630	0.00	0.00	11630	0.00	6.00	11630	0.15
eil33E	12830	0.00	0.00	12830	0.00	1.00	12830	0.15
eil51A	245	0.00	0.00	245	0.00	13.00	245	0.37
eil51B	287	0.00	0.00	287	0.00	6.00	287	0.25
eil51C	122	0.00	0.00	122	0.00	10.00	111	0.14
eil51D	150	0.00	0.00	150	0.00	15.00	148	0.33
eil51E	177	0.00	0.00	177	0.00	12.00	173	0.50
eil76A	520	0.00	1.00	520	0.00	35.00	518	1.43
eil76B	599	0.00	0.00	599	0.00	13.00	599	0.44
eil76C	232	0.00	1.00	232	0.00	61.00	232	0.84
eil76D	312	0.00	1.00	312	0.00	43.00	312	1.02
eil76E	367	0.00	0.00	367	0.00	44.00	367	0.82
eil101A	570	0.00	2.00	570	0.00	237.00	562	1.77
eil101B	612	0.00	0.00	612	0.00	131.00	612	0.93
eil101C	281	0.00	1.00	281	0.00	301.00	281	1.13
eil101D	367	0.00	1.00	367	0.00	449.00	367	1.53
eil101E	414	0.00	2.00	414	0.00	883.00	407	1.74
gil262A	4413	0.00	238.00	n/a	n/a	10799.00	4278	36.50
gil262B	4980	0.00	1.00	n/a	n/a	10799.00	4980	9.33
gil262C	2556	0.00	250.00	n/a	n/a	10799.00	2499	33.55
gil262D	3215	0.00	202.00	n/a	n/a	10799.00	3143	33.12
gil262E	3692	0.00	150.00	n/a	n/a	10799.00	3529	36.98
op21A	165	0.00	0.00	165	0.00	0.00	165	0.04
op21B	135	0.00	0.00	135	0.00	1.00	135	0.02
op21C	150	0.00	0.00	150	0.00	0.00	150	0.03
op21D	155	0.00	0.00	155	0.00	1.00	155	0.04
op32A	85	0.00	0.00	85	0.00	0.00	85	0.12
op32B	115	0.00	0.00	115	0.00	0.00	115	0.07

Continued on next page

Table B.1 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
op32C	35	0.00	0.00	35	0.00	1.00	35	0.02
op32D	60	0.00	0.00	60	0.00	1.00	60	0.05
op32E	75	0.00	0.00	75	0.00	2.00	75	0.08
op33A	260	0.00	0.00	260	0.00	0.00	260	0.10
op33B	330	0.00	0.00	330	0.00	1.00	330	0.08
op33C	110	0.00	0.00	110	0.00	1.00	110	0.03
op33D	160	0.00	0.00	160	0.00	0.00	160	0.05
op33E	180	0.00	0.00	180	0.00	0.00	180	0.06

Table B.2: Results for Class II

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
att48A	12	0.00	0.00	12	0.00	4.00	12	0.07
att48B	11	0.00	0.00	11	0.00	100.00	11	0.12
att48C	12	0.00	0.00	12	0.00	14.00	12	0.07
att48D	12	0.00	0.00	12	0.00	4.00	12	0.08
cmt121A	330	0.00	0.00	330	0.00	3120.00	329	1.03
cmt121B	305	0.00	4.00	301	0.07	10799.00	300	1.70
cmt121C	330	0.00	0.00	330	0.00	1795.00	327	0.84
cmt121D	330	0.00	0.00	330	0.00	964.00	330	0.69
cmt151A	462	0.00	4.00	462	0.00	261.00	454	3.17
cmt151B	561	0.00	0.00	561	0.00	296.00	561	1.06
cmt151C	433	0.00	2.00	433	0.00	195.00	415	2.32
cmt151D	496	0.00	3.00	496	0.00	1535.00	496	2.75
cmt151E	541	0.00	1.00	541	0.00	162.00	540	2.42
cmt200A	825	0.00	2.00	825	0.00	3385.00	825	3.26
cmt200B	486	0.00	16.00	486	0.00	4133.00	481	10.96
cmt200C	625	0.00	28.00	n/a	n/a	10799.00	600	7.50
cmt200D	732	0.00	19.00	699	0.06	10799.00	706	14.80
eil30A	2925	0.00	0.00	2925	0.00	0.00	2925	0.04
eil30B	1625	0.00	0.00	1625	0.00	1.00	1625	0.02
eil30C	1625	0.00	0.00	1625	0.00	2.00	1625	0.02
eil30D	2475	0.00	0.00	2475	0.00	1.00	2475	0.03
eil33A	8480	0.00	0.00	8480	0.00	2.00	8480	0.06
eil33B	8480	0.00	0.00	8480	0.00	1.00	8480	0.06
eil33C	8480	0.00	0.00	8480	0.00	4.00	8480	0.05
eil33D	9430	0.00	0.00	9430	0.00	2.00	9430	0.06

Continued on next page

Table B.2 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
eil51A	168	0.00	0.00	168	0.00	3.00	168	0.13
eil51B	148	0.00	0.00	148	0.00	6.00	148	0.17
eil51C	168	0.00	0.00	168	0.00	6.00	168	0.15
eil51D	168	0.00	0.00	168	0.00	5.00	168	0.13
eil76A	377	0.00	0.00	377	0.00	80.00	377	0.36
eil76B	409	0.00	0.00	409	0.00	11.00	409	0.25
eil76C	201	0.00	0.00	201	0.00	33.00	201	0.44
eil76D	294	0.00	0.00	294	0.00	20.00	293	0.37
eil76E	345	0.00	0.00	345	0.00	29.00	339	0.42
eil101A	386	0.00	1.00	386	0.00	163.00	383	1.47
eil101B	409	0.00	0.00	409	0.00	27.00	409	0.51
eil101C	260	0.00	0.00	260	0.00	69.00	260	1.37
eil101D	326	0.00	0.00	326	0.00	92.00	311	1.34
eil101E	370	0.00	0.00	370	0.00	114.00	370	1.26
gil262A	3003	0.00	0.00	n/a	n/a	10799.00	3003	4.57
gil262B	2418	0.00	94.00	n/a	n/a	10799.00	2294	18.89
gil262C	2861	0.00	40.00	n/a	n/a	10799.00	2788	11.66
gil262D	3000	0.00	39.00	n/a	n/a	10799.00	2967	8.69
op21A	60	0.00	0.00	60	0.00	0.00	60	0.01
op21B	125	0.00	0.00	125	0.00	0.00	125	0.01
op21C	60	0.00	0.00	60	0.00	0.00	60	0.01
op21D	75	0.00	0.00	75	0.00	0.00	60	0.01
op21E	80	0.00	0.00	80	0.00	1.00	80	0.02
op32A	80	0.00	0.00	80	0.00	1.00	80	0.05
op32B	70	0.00	0.00	70	0.00	2.00	70	0.05
op32C	80	0.00	0.00	80	0.00	0.00	80	0.04
op33A	210	0.00	0.00	210	0.00	0.00	210	0.05
op33B	170	0.00	0.00	170	0.00	1.00	170	0.03
op33C	210	0.00	0.00	210	0.00	0.00	210	0.05
op33D	220	0.00	0.00	220	0.00	0.00	220	0.04

Table B.3: Results for Class III

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
att48A	4	0.00	0.00	4	0.00	6.00	4	0.08
att48B	7	0.00	0.00	7	0.00	4.00	7	0.08
att48C	4	0.00	0.00	4	0.00	6.00	4	0.05

Continued on next page

Table B.3 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
att48D	5	0.00	0.00	5	0.00	4.00	5	0.07
att48E	5	0.00	0.00	5	0.00	4.00	5	0.06
cmt121A	152	0.00	0.00	152	0.00	131.00	152	0.37
cmt121B	152	0.00	0.00	152	0.00	145.00	152	0.35
cmt121C	152	0.00	0.00	152	0.00	175.00	152	0.32
cmt121D	152	0.00	0.00	152	0.00	165.00	152	0.35
cmt151A	199	0.00	0.00	199	0.00	344.00	199	0.50
cmt151B	179	0.00	2.00	179	0.00	299.00	179	0.75
cmt151C	199	0.00	0.00	199	0.00	1302.00	199	0.60
cmt151D	199	0.00	0.00	199	0.00	253.00	199	0.55
cmt200A	463	0.00	0.00	463	0.00	7415.00	463	1.40
cmt200B	391	0.00	6.00	391	0.00	1010.00	354	3.06
cmt200C	452	0.00	9.00	452	0.00	10799.00	431	2.59
cmt200D	463	0.00	1.00	463	0.00	5827.00	463	1.66
eil30A	950	0.00	0.00	950	0.00	2.00	950	0.01
eil30B	1950	0.00	0.00	1950	0.00	0.00	1950	0.02
eil30C	2075	0.00	0.00	2075	0.00	0.00	2075	0.02
eil30D	2075	0.00	0.00	2075	0.00	0.00	2075	0.02
eil33A	5530	0.00	0.00	5530	0.00	0.00	5530	0.03
eil33B	5530	0.00	0.00	5530	0.00	0.00	5530	0.02
eil33C	5530	0.00	0.00	5530	0.00	0.00	5530	0.02
eil51A	99	0.00	0.00	99	0.00	3.00	99	0.06
eil51B	90	0.00	0.00	90	0.00	7.00	90	0.08
eil51C	99	0.00	0.00	99	0.00	1.00	99	0.08
eil51D	99	0.00	0.00	99	0.00	1.00	99	0.06
eil76A	167	0.00	0.00	167	0.00	8.00	167	0.15
eil76B	202	0.00	0.00	202	0.00	10.00	202	0.13
eil76C	142	0.00	0.00	142	0.00	16.00	142	0.15
eil76D	193	0.00	0.00	193	0.00	16.00	193	0.25
eil76E	202	0.00	0.00	202	0.00	36.00	202	0.12
eil101A	175	0.00	0.00	175	0.00	22.00	175	0.19
eil101B	155	0.00	0.00	155	0.00	24.00	143	0.32
eil101C	175	0.00	0.00	175	0.00	47.00	175	0.32
eil101D	175	0.00	0.00	175	0.00	56.00	175	0.19
gil262A	1369	0.00	0.00	n/a	n/a	10799.00	1369	2.06
gil262B	1220	0.00	29.00	1220	0.01	10799.00	1036	3.53
gil262C	1369	0.00	5.00	1366	0.00	10799.00	1355	5.62
gil262D	1369	0.00	0.00	1369	0.00	2632.00	1369	2.24
op21A	45	0.00	0.00	45	0.00	0.00	45	0.01
op21B	30	0.00	0.00	30	0.00	0.00	30	0.01

Continued on next page

Table B.3 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
op21C	45	0.00	0.00	45	0.00	0.00	45	0.01
op21D	45	0.00	0.00	45	0.00	0.00	45	0.01
op32A	45	0.00	0.00	45	0.00	0.00	45	0.02
op32B	45	0.00	0.00	45	0.00	0.00	45	0.02
op32C	45	0.00	0.00	45	0.00	0.00	45	0.02
op32D	45	0.00	0.00	45	0.00	0.00	45	0.02
op33A	130	0.00	0.00	130	0.00	0.00	130	0.01
op33B	80	0.00	0.00	80	0.00	0.00	80	0.01
op33C	120	0.00	0.00	120	0.00	1.00	120	0.01
op33D	130	0.00	0.00	130	0.00	0.00	130	0.01

Table B.4: Results for Class IV

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
att48A	21	0.00	0.00	21	0.00	7.00	20	0.63
att48B	29	0.00	0.00	29	0.00	17.00	28	0.90
att48C	13	0.00	0.00	13	0.00	175.00	13	0.34
att48D	17	0.00	0.00	17	0.00	155.00	17	0.58
att48E	20	0.00	0.00	20	0.00	21.00	19	0.55
cmt121A	814	0.00	5.00	802	0.14	10799.00	804	8.37
cmt121B	513	0.00	4.00	509	0.53	10799.00	510	11.50
cmt121C	669	0.00	3.00	669	0.27	10799.00	655	10.45
cmt121D	774	0.00	3.00	757	0.19	10799.00	768	11.65
cmt151A	1087	0.00	7.00	1087	0.00	1257.00	1054	22.74
cmt151B	1483	0.00	8.00	n/a	n/a	10799.00	1451	21.87
cmt151C	477	0.00	15.00	n/a	n/a	10799.00	471	5.83
cmt151D	647	0.00	16.00	641	0.04	10799.00	627	14.71
cmt151E	799	0.00	15.00	799	0.00	3517.00	770	14.80
cmt200A	723	0.00	32.00	n/a	n/a	10799.00	723	20.73
cmt200B	2171	0.00	50.00	n/a	n/a	10799.00	2093	30.33
cmt200C	908	0.00	27.00	n/a	n/a	10799.00	894	21.98
cmt200D	1151	0.00	25.00	n/a	n/a	10799.00	1085	38.87
cmt200E	1324	0.00	13.00	n/a	n/a	10799.00	1291	45.84
eil30A	9375	0.00	0.00	9375	0.00	39.00	9375	0.39
eil30B	5450	0.00	0.00	5450	0.00	365.00	5450	0.14
eil30C	7300	0.00	0.00	7300	0.00	132.00	7300	0.25
eil30D	8575	0.00	0.00	8575	0.00	30.00	8575	0.25

Continued on next page

Table B.4 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
eil33A	6950	0.00	0.00	6950	0.00	74.00	6950	0.19
eil33B	19730	0.00	0.00	19730	0.00	13.00	19730	0.34
eil33C	10630	0.00	0.00	10630	0.00	18.00	10630	0.28
eil33D	14180	0.00	0.00	14180	0.00	10.00	14180	0.31
eil33E	17430	0.00	0.00	17430	0.00	6.00	17430	0.37
eil51A	328	0.00	0.00	328	0.00	39.00	325	0.90
eil51B	509	0.00	0.00	509	0.00	30.00	509	1.53
eil51C	168	0.00	0.00	168	0.00	19.00	168	0.24
eil51D	201	0.00	0.00	201	0.00	14.00	198	0.43
eil51E	232	0.00	0.00	232	0.00	23.00	232	0.62
eil76A	629	0.00	0.00	629	0.00	46.00	623	2.79
eil76B	923	0.00	1.00	923	0.00	133.00	913	1.69
eil76C	274	0.00	1.00	274	0.00	794.00	274	1.63
eil76D	366	0.00	1.00	366	0.00	92.00	366	2.37
eil76E	442	0.00	0.00	442	0.00	35.00	442	2.12
eil101A	789	0.00	2.00	789	0.00	294.00	764	7.87
eil101B	1071	0.00	1.00	1071	0.00	2209.00	1058	6.43
eil101C	326	0.00	2.00	326	0.00	328.00	326	1.38
eil101D	422	0.00	2.00	422	0.00	1895.00	422	2.60
eil101E	491	0.00	3.00	491	0.00	2457.00	491	5.95
gil262A	5668	0.00	260.00	n/a	n/a	10799.00	5423	67.09
gil262B	8007	0.00	284.00	n/a	n/a	10799.00	7657	55.50
gil262C	3002	0.00	196.00	n/a	n/a	10799.00	2894	19.15
gil262D	3883	0.00	164.00	n/a	n/a	10799.00	3752	136.74
gil262E	4475	0.00	189.00	n/a	n/a	10799.00	4279	193.60
op21A	260	0.00	0.00	260	0.00	1.00	260	0.09
op21B	180	0.00	0.00	180	0.00	0.00	180	0.05
op21C	215	0.00	0.00	215	0.00	1.00	215	0.09
op21D	260	0.00	0.00	260	0.00	0.00	260	0.09
op32A	110	0.00	0.00	110	0.00	1.00	110	0.21
op32B	180	0.00	0.00	180	0.00	1.00	180	0.13
op32C	35	0.00	0.00	35	0.00	5.00	35	0.02
op32D	75	0.00	0.00	75	0.00	2.00	75	0.10
op32E	90	0.00	0.00	90	0.00	1.00	90	0.18
op33A	310	0.00	0.00	310	0.00	3.00	310	0.22
op33B	460	0.00	0.00	460	0.00	4.00	460	0.17
op33C	120	0.00	0.00	120	0.00	4.00	120	0.04
op33D	180	0.00	0.00	180	0.00	2.00	180	0.08
op33E	220	0.00	0.00	220	0.00	3.00	220	0.12

Table B.5: Results for Class V

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
att48A	24	0.00	0.00	24	0.00	15.00	24	0.69
att48B	19	0.00	0.00	19	0.00	18.00	19	0.38
att48C	23	0.00	0.00	23	0.00	7.00	22	0.51
att48D	25	0.00	0.00	25	0.00	27.00	25	0.56
cmt121A	640	0.00	5.00	n/a	n/a	10799.00	627	14.38
cmt121B	482	0.00	6.00	n/a	n/a	10799.00	465	11.73
cmt121C	636	0.00	7.00	630	0.16	10799.00	612	6.22
cmt121D	711	0.00	4.00	672	0.15	10799.00	696	7.73
cmt151A	681	0.00	8.00	681	0.00	793.00	657	9.12
cmt151B	1207	0.00	6.00	1207	0.00	2504.00	1184	9.91
cmt151C	610	0.00	7.00	610	0.00	605.00	597	6.74
cmt151D	773	0.00	7.00	773	0.00	1253.00	755	10.01
cmt151E	914	0.00	2.00	914	0.00	861.00	889	12.94
cmt200A	1678	0.00	23.00	1678	0.00	10799.00	1634	65.59
cmt200B	760	0.00	29.00	n/a	n/a	10799.00	685	25.46
cmt200C	1053	0.00	13.00	1053	0.00	2552.00	939	30.35
cmt200D	1262	0.00	13.00	1262	0.00	8399.00	1151	36.69
eil30A	6225	0.00	0.00	6225	0.00	15.00	6225	0.22
eil30B	3350	0.00	0.00	3350	0.00	16.00	3350	0.10
eil30C	4750	0.00	0.00	4750	0.00	17.00	4750	0.12
eil30D	5075	0.00	0.00	5075	0.00	88.00	5075	0.21
eil33A	14230	0.00	0.00	14230	0.00	6.00	14230	0.22
eil33B	12880	0.00	0.00	12880	0.00	9.00	12880	0.20
eil33C	14930	0.00	0.00	14930	0.00	8.00	14930	0.19
eil33D	15680	0.00	0.00	15680	0.00	7.00	15680	0.15
eil51A	402	0.00	0.00	402	0.00	11.00	402	0.52
eil51B	236	0.00	0.00	236	0.00	46.00	235	0.48
eil51C	302	0.00	0.00	302	0.00	49.00	298	0.73
eil51D	357	0.00	0.00	357	0.00	22.00	340	0.69
eil76A	481	0.00	0.00	481	0.00	274.00	481	3.01
eil76B	754	0.00	0.00	754	0.00	37.00	747	3.50
eil76C	228	0.00	0.00	228	0.00	156.00	228	0.76
eil76D	341	0.00	0.00	341	0.00	78.00	333	1.86
eil76E	411	0.00	0.00	411	0.00	152.00	403	1.69
eil101A	559	0.00	2.00	559	0.00	489.00	538	6.79
eil101B	857	0.00	2.00	857	0.00	1138.00	848	3.81
eil101C	296	0.00	1.00	296	0.00	82.00	291	2.44

Continued on next page

Table B.5 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
eil101D	418	0.00	1.00	418	0.00	175.00	403	3.06
eil101E	521	0.00	1.00	521	0.00	327.00	495	3.96
gil262A	6363	0.00	46.00	n/a	n/a	10799.00	6183	41.17
gil262B	3766	0.00	119.00	n/a	n/a	10799.00	3128	35.84
gil262C	4875	0.00	130.00	n/a	n/a	10799.00	4476	115.39
gil262D	5593	0.00	156.00	n/a	n/a	10799.00	5215	86.78
op21A	135	0.00	0.00	135	0.00	0.00	135	0.01
op21B	235	0.00	0.00	235	0.00	1.00	235	0.04
op21C	135	0.00	0.00	135	0.00	0.00	135	0.01
op21D	135	0.00	0.00	135	0.00	0.00	135	0.01
op21E	165	0.00	0.00	165	0.00	0.00	165	0.02
op32A	150	0.00	0.00	150	0.00	3.00	150	0.14
op32B	110	0.00	0.00	110	0.00	2.00	110	0.15
op32C	140	0.00	0.00	140	0.00	2.00	140	0.15
op33A	400	0.00	0.00	400	0.00	3.00	400	0.18
op33B	270	0.00	0.00	270	0.00	5.00	270	0.06
op33C	410	0.00	0.00	410	0.00	0.00	410	0.17
op33D	440	0.00	0.00	440	0.00	4.00	440	0.14

Table B.6: Results for Class VI

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
att48A	11	0.00	0.00	11	0.00	10.00	11	0.25
att48B	19	0.00	0.00	19	0.00	14.00	18	0.43
att48C	12	0.00	0.00	12	0.00	7.00	11	0.22
att48D	14	0.00	0.00	14	0.00	13.00	14	0.34
att48E	16	0.00	0.00	16	0.00	6.00	15	0.55
cmt121A	503	0.00	3.00	503	0.14	10799.00	491	6.53
cmt121B	458	0.00	2.00	456	0.25	10799.00	446	6.31
cmt121C	557	0.00	3.00	510	0.24	10799.00	547	5.94
cmt121D	618	0.00	3.00	573	0.10	10799.00	611	3.06
cmt151A	873	0.00	6.00	873	0.00	4120.00	842	10.38
cmt151B	549	0.00	10.00	549	0.00	7921.00	517	14.34
cmt151C	740	0.00	6.00	740	0.00	2371.00	689	8.91
cmt151D	852	0.00	6.00	852	0.00	730.00	816	12.00
cmt200A	1445	0.00	28.00	1430	0.01	10799.00	1387	27.97
cmt200B	955	0.00	16.00	n/a	n/a	10799.00	932	13.68

Continued on next page

Table B.6 – Continued from previous page

Inst.	B&C			GG			HVNS	
	z	Gap(%)	CPU(s)	z	Gap(%)	CPU(s)	z	CPU(s)
cmt200C	1224	0.00	7.00	1224	0.00	1574.00	1148	26.47
cmt200D	1386	0.00	14.00	1385	0.00	10799.00	1298	27.46
eil30A	1550	0.00	0.00	1550	0.00	251.00	1550	0.02
eil30B	4775	0.00	0.00	4775	0.00	8.00	4775	0.12
eil30C	5550	0.00	0.00	5550	0.00	4.00	5550	0.09
eil30D	5850	0.00	0.00	5850	0.00	2.00	5725	0.04
eil33A	10600	0.00	0.00	10600	0.00	8.00	10600	0.15
eil33B	13330	0.00	0.00	13330	0.00	3.00	13330	0.11
eil33C	14030	0.00	0.00	14030	0.00	12.00	14030	0.05
eil51A	329	0.00	0.00	329	0.00	9.00	329	0.66
eil51B	212	0.00	0.00	212	0.00	13.00	212	0.45
eil51C	289	0.00	0.00	289	0.00	9.00	281	0.42
eil51D	344	0.00	0.00	344	0.00	11.00	343	0.50
eil76A	233	0.00	0.00	233	0.00	79.00	233	1.23
eil76B	613	0.00	1.00	613	0.00	180.00	602	1.85
eil76C	211	0.00	0.00	211	0.00	50.00	211	0.92
eil76D	305	0.00	1.00	305	0.00	322.00	304	1.64
eil76E	386	0.00	2.00	386	0.00	213.00	372	1.39
eil101A	633	0.00	1.00	633	0.00	839.00	620	2.94
eil101B	344	0.00	3.00	344	0.00	570.00	335	1.40
eil101C	474	0.00	2.00	474	0.00	405.00	466	7.20
eil101D	555	0.00	1.00	555	0.00	296.00	541	4.91
gil262A	4784	0.00	133.00	n/a	n/a	10799.00	4516	51.51
gil262B	3063	0.00	286.00	n/a	n/a	10799.00	2486	23.35
gil262C	4102	0.00	56.00	n/a	n/a	10799.00	3807	29.68
gil262D	4686	0.00	112.00	n/a	n/a	10799.00	4393	59.59
op21A	155	0.00	0.00	155	0.00	1.00	155	0.03
op21B	75	0.00	0.00	75	0.00	1.00	75	0.02
op21C	110	0.00	0.00	110	0.00	1.00	110	0.04
op21D	160	0.00	0.00	160	0.00	1.00	160	0.04
op32A	115	0.00	0.00	115	0.00	1.00	115	0.13
op32B	65	0.00	0.00	65	0.00	4.00	65	0.04
op32C	105	0.00	0.00	105	0.00	2.00	105	0.15
op32D	130	0.00	0.00	130	0.00	0.00	130	0.11
op33A	350	0.00	0.00	350	0.00	4.00	350	0.09
op33B	130	0.00	0.00	130	0.00	4.00	130	0.03
op33C	240	0.00	0.00	240	0.00	6.00	240	0.11
op33D	300	0.00	0.00	300	0.00	2.00	300	0.11

Appendix C

Detailed results of families of cuts tests

Tables are shown by class, showing the results of the methods solved without certain families of cuts. The first column has the name of the exercise, the second has the total amount of time to solve the instances in seconds, the third column has the average time, the third has the percentage of average gap, and the fourth is the percentage of feasible instances.

Table C.1: Performance on root without cuts families on Class 1.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible instances**
Root	297	4.8	37.8	98.4
W/o Logical Cuts	341	5.5	23.4	100
W/o Matching Cuts	377	6.1	36.7	98.4
W/o Infeasible Path Cuts	291	4.7	25.6	100
W/o Connectivity Cuts	120	1.9	35.8	98.4
Root w/all	291	4.7	20.8	98.4
Root w/o any	95	1.5	43.4	100
W/ Logical Cuts	90	1.5	44.4	98.4
W/ Matching Cuts	83	1.3	40.5	98.4
W/ Infeasible Path Cuts	123	2.0	38.5	100
W/ Connectivity Cuts	340	5.5	21.6	100
W/o Logical Cuts	244	3.9	25.1	100
W/o Matching Cuts	310	5.0	25.4	100
W/o Infeasible Path Cuts	293	4.7	23.7	100
W/o Connectivity Cuts	122	2.0	32.6	98.4

Table C.2: Performance on B&C without cuts families on Class 1.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
B&C	1069	17.2	0.0	100
W/o Logical Cuts	895	14.4	0.0	100
W/o Matching Cuts	765	12.3	0.0	100
W/o Infeasible Path Cuts	894	14.4	0.0	100
W/o Connectivity Cuts	42805	690.4	0.0	100
W/ Logical Cuts	48665	785	3.9	100
W/ Matching Cuts	48693	785.4	3.8	100
W/ Infeasible Path Cuts	43155	696	2.4	100
W/ Connectivity Cuts	798	12.9	0.0	100

Table C.3: Performance on root without cuts families on Class 2.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
Root	58	1.1	10.7	100
W/o Logical Cuts	60	1.1	17.2	100
W/o Matching Cuts	53	0.1	13.1	100
W/o Infeasible Path Cuts	67	1.2	13.8	100
W/o Connectivity Cuts	78	1.4	17.2	96.4
Root w/all	60	1.1	15.8	100
Root w/o any	54	1.0	24.9	96.4
W/ Logical Cuts	49	0.9	23.5	92.7
W/ Matching Cuts	48	0.9	23.7	96.4
W/ Infeasible Path Cuts	72	1.3	18.0	96.4
W/ Connectivity Cuts	46	0.8	12.5	100
W/o Logical Cuts	62	1.1	9.7	100
W/o Matching Cuts	60	1.1	13.1	100
W/o Infeasible Path Cuts	48	0.9	12.8	100
W/o Connectivity Cuts	84	1.5	17.3	98.2

Table C.4: Performance on B&C without cuts families on Class 2.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
B&C	253	4.6	0.0	100
W/o Logical Cuts	206	3.7	0.0	100
W/o Matching Cuts	263	4.8	0.0	100
W/o Infeasible Path Cuts	214	3.9	0.0	100
W/o Connectivity Cuts	4048	74.1	0.0	100
W/ Logical Cuts	12558	228.3	0.3	100
W/ Matching Cuts	12759	232	0.4	100
W/ Infeasible Path Cuts	4142	75.3	0.0	100
W/ Connectivity Cuts	184	3.3	0.0	100

Table C.5: Performance on root without cuts families on Class 3.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
Root	28.0	0.5	6.3	98.1
W/o Logical Cuts	20.0	0.4	23.4	98.1
W/o Matching Cuts	20.0	0.4	9.7	100.0
W/o Infeasible Path Cuts	20.0	0.4	8.0	100.0
W/o Connectivity Cuts	25.0	0.5	16.2	92.5
Root w/all	18.0	0.4	8.0	100.0
Root w/o any	24.0	0.5	17.4	94.3
W/ Logical Cuts	23.0	0.4	10.7	92.5
W/ Matching Cuts	27.0	0.5	19.5	100.0
W/ Infeasible Path Cuts	27.0	0.5	17.3	96.2
W/ Connectivity Cuts	23.0	0.4	8.7	98.1
W/o Logical Cuts	28.0	0.5	26.7	98.1
W/o Matching Cuts	27.0	0.5	24.0	98.1
W/o Infeasible Path Cuts	25.0	0.4	27.0	98.1
W/o Connectivity Cuts	32.0	0.6	13.5	98.1

Table C.6: Performance on B&C without cuts families on Class 3.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
B&C	52.0	1.0	0.0	100.0
W/o Logical Cuts	51.0	0.1	0.0	100.0
W/o Matching Cuts	43.0	0.8	0.0	100.0
W/o Infeasible Path Cuts	48.0	0.9	0.0	100.0
W/o Connectivity Cuts	89.0	1.68	0.0	100.0
W/ Logical Cuts	1972.0	37.2	0.0	100.0
W/ Matching Cuts	2703.0	51.0	0.0	100.0
W/ Infeasible Path Cuts	78.0	1.5	0.0	100.0
W/ Connectivity Cuts	40.0	0.8	0.0	100.0

Table C.7: Performance on root without cuts families on Class 4.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
Root	231.0	4.4	29.5	100.0
W/o Logical Cuts	291.0	4.4	22.6	100.0
W/o Matching Cuts	275.0	4.4	23.0	100.0
W/o Infeasible Path Cuts	303.0	4.9	24.4	100.0
W/o Connectivity Cuts	243.0	3.9	36.7	95.2
W/ Logical Cuts	221.0	3.6	22.7	100.0
W/ Matching Cuts	326.0	5.3	24.7	100.0
W/ Infeasible Path Cuts	316.0	5.1	23.1	100.0
W/ Connectivity Cuts	332.0	5.4	22.0	100.0

Table C.8: Performance on B&C without cuts families on Class 4.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
B&C	1329.0	21.4	0.0	100.0
W/o Logical Cuts	1033.0	16.7	0.0	100.0
W/o Matching Cuts	965.0	15.6	0.0	100.0
W/o Infeasible Path Cuts	1175.0	19.0	0.0	100.0
W/o Connectivity Cuts	52113.0	840.5	0.0	100.0
W/ Logical Cuts	80876.0	1304.5	9.1	100.0
W/ Matching Cuts	81797.0	1319.3	9.1	100.0
W/ Infeasible Path Cuts	51720.0	834.2	3.9	100.0
W/ Connectivity Cuts	1255.0	20.2	0.0	100.0

Table C.9: Performance on root without cuts families on Class 5.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
Root	157.0	3.0	20.0	100.0
W/o Logical Cuts	156.0	2.8	19.0	100.0
W/o Matching Cuts	147.0	2.7	19.5	100.0
W/o Infeasible Path Cuts	188.0	3.4	20.8	100.0
W/o Connectivity Cuts	198.0	3.6	27.3	96.4
W/ Logical Cuts	192.0	3.5	21.1	98.1
W/ Matching Cuts	132.0	2.4	18.2	100.0
W/ Infeasible Path Cuts	180.0	3.3	20.1	100.0
W/ Connectivity Cuts	180.0	3.3	18.8	100.0

Table C.10: Performance on B&C without cuts families on Class 5.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
B&C	588.0	10.7	0.0	100.0
W/o Logical Cuts	451.0	8.2	0.0	100.0
W/o Matching Cuts	476.0	8.7	0.0	100.0
W/o Infeasible Path Cuts	409.0	7.4	0.0	100.0
W/o Connectivity Cuts	19441.0	353.5	0.0	100.0
W/ Logical Cuts	46209.0	840.2	2.8	100.0
W/ Matching Cuts	46463.0	844.8	2.9	100.0
W/ Infeasible Path Cuts	19500.0	354.5	1.2	100.0
W/ Connectivity Cuts	450.0	8.2	0.0	100.0

Table C.11: Performance on root without cuts families on Class 6.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
Root	137.0	2.6	20.5	94.3
W/o Logical Cuts	116.0	2.2	37.3	98.1
W/o Matching Cuts	127.0	2.4	38.2	98.1
W/o Infeasible Path Cuts	74.0	1.4	20.4	100.0
W/o Connectivity Cuts	214.0	4.0	28.0	90.6
W/ Logical Cuts	123.0	2.3	19.7	98.1
W/ Matching Cuts	131.0	2.5	18.0	96.2
W/ Infeasible Path Cuts	70.0	1.3	37.8	100.0
W/ Connectivity Cuts	79.0	1.5	19.1	100.0

Table C.12: Performance on B&C without cuts families on Class 6.

Method	**Total time (in s)**	**Avg time (in s)**	**% Avg gap**	**% Feasible**
B&C	702.0	13.2	0.0	100.0
W/o Logical Cuts	688.0	13.0	0.0	100.0
W/o Matching Cuts	715.0	13.5	0.0	100.0
W/o Infeasible Path Cuts	845.0	16.0	0.0	100.0
W/o Connectivity Cuts	9018.0	150.2	0.0	100.0
W/ Logical Cuts	44461.0	838.9	1.5	100.0
W/ Matching Cuts	44304.0	835.9	1.5	100.0
W/ Infeasible Path Cuts	8866.0	167.3	0.2	100.0
W/ Connectivity Cuts	691.0	13.0	0.0	100.0

Bibliography

Allen, S., J. Heller, K. L. Gentry Lamb, J. Stewart and B. Lemay (2024), «Vehicle Routing Problem Approach for Improving Fuel Delivery Scheduling to Austere Test Sites», *Industrial and Systems Engineering Review*, **12**(2), 109–114.

Barrena, E., D. Canca, L. C. Coelho and G. Laporte (2023), «Analysis of the selective traveling salesman problem with time-dependent profits», *Top*, **31**(1), 165–193.

Cook, W., W. Cunninghm, W. Pulleyblank and A. Schrijver (1998), *Combinatorial Optimization*, Willey - Interscience.

Dantzig, G., R. Fulkerson and S. Johnson (1954), «Solution of a large-scale traveling-salesman problem», *Journal of the operations research society of America*, **2**(4), 393–410.

Feillet, D., P. Dejax and M. Gendreau (2005), «Traveling Salesman Problems with Profits», *Transportation Science*, **39**, 188–205.

Fischetti, M., J. J. Salazar González and P. Toth (1998), «Solving the Orienteering Problem through Branch-and-Cut», *INFORMS Journal on Computing*, **10**, 133–148.

Gama, R. and H. L. Fernandes (2021), «A reinforcement learning approach to the orienteering problem with time windows», *Computers & Operations Research*, **133**, 105 357.

Gavish, B. and S. C. Graves (1978), «The Travelling Salesman Problem and Related Problems», *Working Paper OR 078-78*, Massachusetts Institute of Technology, Operations Research Center, research supported in part by the Office of Naval Research under Contract N00014-75-C-0556.

Gendreau, M., G. Laporte and F. Semet (1998a), «A branch-and-cut algorithm for the undirected selective traveling salesman problem», *Networks*, **32**, 263–273.

Gendreau, M., G. Laporte and F. Semet (1998b), «A branch-and-cut algorithm for the undirected selective traveling salesman problem», *Networks (N. Y.)*, **32**(4), 263–273.

Golden, B., L. Levy and R. Vohra (1987), «The orienteering problem», *Nav Res Logist*, **34**, 307–318.

GRÖTSCHEL, M. and O. HOLLAND (1985), «Solving matching problems with linear programming», *Mathematical Programming*, **33**, 243–259.

GRÖTSCHEL, M. and O. HOLLAND (1991), «Solution of large-scale symmetric travelling salesman problems», *Mathematical Programming*, **51**(1), 141–202.

HA, Q. M., Y. DEVILLE, Q. D. PHAM and M. H. HÀ (2020), «A hybrid genetic algorithm for the traveling salesman problem with drone», *Journal of Heuristics*, **26**, 219–247.

KHODADADIAN, M., A. DIVSALAR, C. VERBEECK, A. GUNAWAN and P. VANSTEENWEGEN (2022), «Time dependent orienteering problem with time windows and service time dependent profits», *Computers & Operations Research*, **143**, 105 794.

KOBEAGA, G., J. ROJAS-DELGADO, M. MERINO and J. A. LOZANO (2024), «A revisited branch-and-cut algorithm for large-scale orienteering problems», *European Journal of Operational Research*, **313**(1), 44–68.

LAPORTE, G. and S. MARTELLO (1990), «The selective travelling salesman problem», *Discrete Applied Mathematics*, **26**(2), 193–207.

LEIFER, A. C. and M. B. ROSENWEIN (1994), «Strong linear programming relaxations for the orienteering problem», *European Journal of Operational Research*, **73**(3), 517–523.

LIN, S.-W. and F. Y. VINCENT (2017), «Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing», *Computers & Industrial Engineering*, **114**, 195–205.

LU, Y., U. BENLIC and Q. WU (2018), «A memetic algorithm for the orienteering problem with mandatory visits and exclusionary constraints», *European Journal of Operational Research*, **268**(1), 54–69.

MARTINS, L. D. C., R. D. TORDECILLA, J. CASTANEDA, A. A. JUAN and J. FAULIN (2021), «Electric vehicle routing, arc routing, and team orienteering problems in sustainable transportation», *Energies*, **14**(16), 5131.

MATAI, R., S. P. SINGH and M. L. MITTAL (2010), «Traveling salesman problem: an overview of applications, formulations, and solution approaches», *Traveling salesman problem, theory and applications*, **1**(1), 1–25.

NEKOVÁŘ, F., J. FAIGL and M. SASKA (2021), «Multi-tour set traveling salesman problem in planning power transmission line inspection», *IEEE Robotics and Automation Letters*, **6**(4), 6196–6203.

PADBERG, M. and G. RINALDI (1991), «A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems», *SIAM review*, **33**(1), 60–100.

PALOMO-MARTÍNEZ, P., M. A. SALAZAR-AGUILAR and V. M. ALBORNOZ (2017a), «Formulations for the orienteering problem with additional constraints», *Annals of Operations Research*, **258**, 503–545.

PALOMO-MARTÍNEZ, P. J., M. ANGÉLICA SALAZAR-AGUILAR, G. LAPORTE and A. LANGEVIN (2017b), «A hybrid variable neighborhood search for the Orienteering Problem with mandatory visits and exclusionary constraints», *Computers & Operations Research*, **78**, 408–419.

PEYMAN, M., X. A. MARTIN, J. PANADERO and A. A. JUAN (2024), «A Sim-Learnheuristic for the Team Orienteering Problem: Applications to Unmanned Aerial Vehicles», *Algorithms*, **17**(5), 200.

PÉREZ-FRANCO, M., V. BOYER and M. A. SALAZAR-AGUILAR (forthcoming), «The Orienteering Problem with Mandatory Visits and Conflicts: A Branch and Cut Approach», *Journal of the Operational Research Society TJOR*.

RAMESH, R., Y. SEOK YOON and M. H. KARWAN (1992), «An Optimal Algorithm for the Orienteering Tour Problem», *INFORMS J. Comput.*, **4**, 155–165.

SALAZAR-AGUILAR, M. A., A. LANGEVIN and G. LAPORTE (2014), «The multi-district team orienteering problem», *Computers & Operations Research*, **41**, 76–82.

THAYER, T. C. and S. CARPIN (2021), «An adaptive method for the stochastic orienteering problem», *IEEE Robotics and Automation Letters*, **6**(2), 4185–4192.

TSILIGIRIDES, T. (1984), «Heuristic Methods Applied to Orienteering», *Journal of the Operational Research Society*, **35**, 797–809.

WU, Q., M. HE, J.-K. HAO and Y. LU (2024), «An effective hybrid evolutionary algorithm for the clustered orienteering problem», *European Journal of Operational Research*, **313**(2), 418–434.

XU, W., Z. XU, J. PENG, W. LIANG, T. LIU, X. JIA and S. K. DAS (2020), «Approximation algorithms for the team orienteering problem», en *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, IEEE, 1389–1398.

Autobiography

Marlene Pérez Franco
Candidato para obtener el grado de
Maestría en Ciencias de la Ingeniería
con orientación en Sistemas

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:
A Branch and Cut Approach to The Orienteering Problem with Mandatory Visits and Conflicts

I studied Mechatronics Engineering at the "Universidad Nacional Autónoma de México" (UNAM). I started my Master's studies at the Graduate Program on Systems Engineering (PISIS) at "Universidad Autónoma de Nuevo León" (UANL). I had the opportunity to participate in the Latin-Iberoamerican Conference on Operations Research (CLAIO) in Guadalajara and present advances of this thesis work. A scientific article has also been written and is under review.