POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Alternative Fractional Solutions in a Column-Generation-Based Diving Heuristic Applied to the Multi-Depot Electric Vehicle Scheduling Problem

THÉO MAËL LOUESDON
Département de mathématiques et génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques

Juillet 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Alternative Fractional Solutions in a Column-Generation-Based Diving Heuristic Applied to the Multi-Depot Electric Vehicle Scheduling Problem

présenté par **Théo Maël LOUESDON**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my thesis supervisor, Guy Desaulniers, for his guidance, patience, and support throughout this research journey. His expertise and insightful feedback have been instrumental in shaping this work.

I am equally grateful to my co-supervisor, Andrea Lodi, whose thoughtful advice and challenging questions pushed me to refine my ideas and deepen my understanding of the subject matter.

My heartfelt thanks go to my parents, whose support and encouragement have sustained me through the challenges of graduate studies. Their belief in me has been a constant source of strength, and I could not have completed this work without their love and support.

This journey would not have been as smooth without all of you.

RÉSUMÉ

Le MDEVSP constitue un défi d'optimisation critique pour les systèmes de transport public en transition vers des flottes électriques. Ce problème constiste à déterminer le meilleur ordonnancement des véhicules électriques provenant de plusieurs dépôts, en tenant compte de leurs contraintes spécifiques comme l'autonomie limitée des batteries et les besoins de recharge, afin de satisfaire un ensemble de tâches tout en minimisant les coûts opérationnels. Cette thèse se concentre sur l'amélioration des heuristiques de plongée basées sur la génération de colonnes pour résoudre efficacement le MDEVSP. Les approches traditionnelles résolvent le problème maître jusqu'à l'optimalité par génération de colonnes, fixent à 1 la variable ayant la plus grande valeur fractionnaire, puis recommencent l'ensemble du processus avec cette contrainte supplémentaire. Bien que calculatoirement abordable, cette procédure de fixation séquentielle conduit souvent à des solutions sous-optimales en raison de décisions précoces et irréversibles qui se propagent lors des itérations suivantes.

Cette recherche examine si l'exploration de solutions optimales alternatives du problème maître, en utilisant le groupe de colonnes existant, peut améliorer la qualité finale des solutions. Nous proposons et analysons deux approches méthodologiques : une technique de sondage qui évalue individuellement les candidats prometteurs par programmation linéaire, et des programmes en nombres entiers conçus pour identifier les variables dont les caractéristiques pourraient conduire à de meilleures solutions globales. Les méthodes proposées conservent la nature heuristique de la résolution tout en tentant de prendre des décisions de branchement plus réfléchies.

Des expériences computationnelles ont été menées sur plusieurs types d'instances de complexités variées : des instances à dépôt unique avec environ 700 tâches, des instances à deux dépôts avec environ 800 tâches, et des instances à deux dépôts plus importantes avec environ 1300 tâches. Les tests initiaux ont utilisé 10 instances de chaque type, les techniques prometteuses étant ensuite évaluées à plus grande échelle.

Les résultats démontrent des améliorations d'écart d'optimalité moyennes modestes allant dans les meilleurs cas de 0,25% à 1,0%, avec des améliorations maximales généralement supérieures à 2,5%. Cependant, certaines méthodes ont présenté des détériorations dans le pire des cas dépassant 2%, ce qui indique la difficulté de développer des stratégies de sélection de variables efficaces. Contrairement aux hypothèses initiales, une corrélation limitée a été trouvée entre la proximité d'une variable à 1 et son impact sur la qualité de la solution lorsqu'elle est fixée.

Nos expérimentations montrent également que l'exécution parallèle de plusieurs méthodes permet d'améliorer l'écart d'optimalité moyen tout en réduisant, dans plusieurs cas, l'écart maximal jusqu'à 40%. Cette approche combinatoire présente un avantage significatif pour la robustesse globale du processus de résolution.

Notre recherche apporte des améliorations pratiques applicables aux approches par génération de colonnes utilisant des heuristiques de plongée, particulièrement pour les problèmes de planification de véhicules électriques. Les résultats soulignent l'impact critique des décisions de branchement précoces, car elles contraignent l'espace de solution pour les itérations suivantes, mettant en évidence un défi des heuristiques de plongée qui mérite d'être approfondi.

ABSTRACT

The MDEVSP constitutes a critical optimization challenge for public transportation systems transitioning to electric fleets. This problem involves determining the best scheduling of electric vehicles from multiple depots, taking into account their specific constraints such as limited battery autonomy and charging needs, in order to satisfy a set of tasks (bus trips) while minimizing operational costs. This thesis focuses on enhancing column generation-based diving heuristics for solving the MDEVSP efficiently. Traditional approaches solve the master problem to optimality through column generation, fix the variable with the largest fractional value to one, then restart the entire process with this additional constraint. While computationally tractable, this sequential fixing procedure often leads to suboptimal solutions due to early, irreversible decisions that propagate through subsequent iterations.

This research investigates whether exploring alternative optimal solutions of the master problem, using the existing column pool, can improve final solution quality. We propose and analyze two methodological approaches: a probing technique that systematically evaluates promising candidates through linear programming, and specialized mixed-integer programs designed to identify variables with characteristics potentially leading to better overall solutions. The proposed methods maintain the heuristic nature of the resolution while attempting to make more informed branching decisions.

Computational experiments were conducted on multiple instance types with varying complexities: single-depot instances with approximately 700 tasks, two-depot instances with approximately 750 tasks, and larger two-depot instances with approximately 1300 tasks. Initial testing used 10 instances of each type, with promising techniques subsequently evaluated on larger scales.

Results demonstrate modest improvements of average optimality gaps ranging, in the best cases, from 0.25% to 1.0%, with maximum improvements generally exceeding 2.5%. However, certain methods exhibited deteriorations in worst-case scenarios exceeding 2%, indicating the difficulty of developing systematically beneficial variable selection strategies. Contrary to initial hypotheses, a limited correlation was found between a variable's proximity to 1 and its impact on solution quality when fixed.

Our experiments further demonstrate that parallel execution of multiple methods enables improvement in the average optimality gap while reducing, in several cases, the maximum gap by up to 40%. This combinatorial approach presents a significant advantage for the overall robustness of the resolution process.

Our research contributes practical enhancements applicable to column generation approaches employing diving heuristics, particularly for electric vehicle scheduling problems. The findings underline the critical impact of early branching decisions, as these fundamentally constrain the solution space for subsequent iterations, highlighting an intrinsic challenge in diving heuristics that warrants further investigation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

MDVSP	Multi-depot vehicle scheduling problem
MDEVSP	Multi-depot electric vehicle scheduling problem
MIP	Mixed-integer programming
LP	Linear program
RMP	Restricted master problem
CFIX	Column fixing
ITFIX	Inter-task fixing
RBFIX	Retrospective branching
SPPRC	Shortest Path Problems with Resource Constraints
SoC	State of Charge

CHAPTER 1 INTRODUCTION

1.1 Context and motivation

The urban transportation landscape is undergoing a substantial transformation with the electrification of public transit fleets. According to Bloomberg, electric buses now constitute 26% of global bus sales, with over 720,000 units operating worldwide [1]. This shift represents not just a technological evolution but a fundamental rethinking of how public transit systems function in urban environments.

The advantages of electric bus implementation extend beyond environmental considerations. While reduced greenhouse gas emissions and decreased noise pollution are significant benefits [2], the transition also presents unique opportunities to reimagine transit planning frameworks that have remained largely unchanged for decades. Electric vehicles introduce new variables into transit optimization, particularly regarding energy consumption patterns, charging infrastructure requirements, and operational constraints related to battery range.

Transit agencies face challenges in this transition. The conventional approach to bus scheduling typically focuses on minimizing vehicle count and operational hours while ensuring service coverage. With electric buses, this problem must expand to incorporate charging time windows, energy consumption, the non-linear nature of charging functions, and the strategic placement of charging infrastructure. These considerations create a substantially more complex optimization problem.

The financial implications of this transition are considerable. Electric buses require higher initial capital investment but offer reduced operational costs over their lifecycle. Optimizing schedules and routes becomes even more critical when balancing these economic factors. Inefficient scheduling that fails to account for battery constraints could necessitate additional vehicles or compromise service reliability, undermining the economic usefulness for electrification.

Transit planning traditionally unfolds through sequential optimization stages: network design, timetable development, vehicle scheduling, and crew assignment. The introduction of electric vehicles creates stronger interdependencies among these stages. For example, vehicle scheduling decisions now directly impact charging infrastructure requirements, which in turn can affect depot design and location planning.

Municipal transit authorities and transportation companies increasingly recognize the need for sophisticated decision support systems that can handle these new complexities. Some methods that work efficiently for conventional vehicles may perform poorly or produce sub-optimal solutions when adapted to electric fleets.

This research addresses a critical component of the electric transit planning process—the Multiple Depot Electric Vehicle Scheduling Problem (MDEVSP). Working in collaboration with GIRO, a leading provider of transit optimization software, this study aims to develop a different solution methodology that could effectively manage the complexity of the problems while improving overall results compared to the current heuristic solution methodology.

1.2 Research objectives

The primary objective of this research is to investigate and develop techniques for improving column generation based diving heuristics for the MDEVSP by exploring alternative optimal solutions to the master problem, without adding the computational expense of regenerating columns. Specifically, this thesis aims to:

- Design and implement probing methods that evaluate promising candidate variables through linear programming to predict their effects on solution quality.
- Formulate specialized mixed-integer programming models that identify variables with characteristics potentially leading to better overall solutions.
- Evaluate the performance of the proposed methods against traditional diving heuristics through experiments on diverse MDEVSP instances.

The research focuses on methodological improvements that maintain the heuristic nature of the resolution while making more informed branching decisions. By addressing these objectives, this thesis aims to contribute practical enhancements to column generation approaches for electric vehicle scheduling problems, with potential applications to similar optimization problems solved using diving heuristics.

1.3 Thesis structure

The rest of this thesis is organized into five chapters:

- Chapter 2 presents a comprehensive literature review covering the Multiple Depot Vehicle Scheduling Problem (MDVSP), its extension to electric vehicles (MDEVSP), degeneracy issues in column generation, and approaches for exploring alternative optimal solutions. The chapter concludes by highlighting the specific contributions of this thesis to the existing body of knowledge.

- Chapter 3 defines the MDEVSP in detail and describes the column generation-based solution method. It also discusses existing branching heuristics used to obtain integer solutions, with particular emphasis on diving heuristics.

- Chapter 4 introduces the proposed techniques for generating and evaluating alternative fractional solutions. It details two main approaches: probing techniques that evaluate promising candidates, and specialized mixed-integer programming formulations designed to identify variables with desirable characteristics.

- Chapter 5 presents the experimental results, analyzing the performance of the proposed methods on various MDEVSP instances. It provides a comparative analysis of different approaches and identifies patterns in their effectiveness.

- Chapter 6 concludes the thesis with a summary of the work, acknowledging limitations, and suggesting directions for future research.

Throughout these chapters, the thesis emphasizes the practical implications of the research for improving electric vehicle scheduling in public transportation systems, contributing to more sustainable and efficient operations.

CHAPTER 2 LITERATURE REVIEW

This chapter presents a review of the relevant literature concerning vehicle scheduling problems, with a particular focus on electric vehicles and multiple depot configurations. The review progresses from fundamental vehicle scheduling concepts to more specialized applications involving electric vehicles, highlighting the key algorithmic approaches and methodological advances that inform our research.

2.1 Multiple Depot Vehicle Scheduling Problem

The MDVSP is a classic combinatorial optimization problem that involves assigning trip schedules to vehicles from different depots while minimizing operational costs. This problem has been extensively studied due to its importance in public transportation planning. The MDVSP is known to be NP-hard, which means that there is no polynomial algorithm to solve it exactly for all instances [3]. In contrast, it is worth noting that the Single Depot Vehicle Scheduling Problem (SDVSP), which is a special case of the MDVSP where only one depot is available, is polynomially solvable as a minimum-cost flow problem [4].

Several approaches have been proposed to solve the MDVSP. One effective method was developed by Bunte and Kliewer, who used an integer linear programming approach to model the problem [5]. Their work was particularly significant as it introduced relaxation techniques to reduce the problem's complexity and demonstrated how these techniques could be applied to large-scale instances. They also conducted extensive computational experiments that showed the efficiency of their approach for real-world transit networks.

Another popular approach that has been successfully applied is column generation. This method solves the problem iteratively by progressively adding new schedules (columns) to a restricted master problem, which significantly reduces the search space. Oukil et al. [6] made substantial contributions by demonstrating how column generation could be effectively implemented for the MDVSP, especially for large instances where traditional methods fail. Their work was instrumental in establishing column generation as a preferred solution method for complex vehicle scheduling problems.

The research on MDVSP has established several critical insights that inform more specialized vehicle scheduling problems. Column generation has emerged as a particularly effective technique for addressing large-scale instances, offering a balance between solution quality and computational efficiency. The methodological advances in solving the MDVSP provide valuable foundations for tackling the more complex MDEVSP, which incorporates additional constraints related to electric vehicles.

2.2 Multiple Depot Electric Vehicle Scheduling Problem

The MDEVSP is an extension of the MDVSP that takes into account the specific constraints of electric vehicles, such as limited battery range, charging times, and the availability of charging stations. These additional constraints make the problem more complex and require adapted solution approaches.

Unlike the polynomially solvable SDVSP, its electric vehicle counterpart introduces significant computational challenges. Li [7] established the NP-hardness of the Single Depot Electric Vehicle Scheduling Problem by examining transit bus scheduling under energy constraints. His formulation incorporated battery management considerations, including both battery exchange strategies and rapid charging operations with fixed durations and costs, demonstrating that these additional dimensions render polynomial-time solutions infeasible despite maintaining the single depot structure. Further substantiating this complexity classification, Sassi and Oulamara [8] provided a theoretical analysis of the Electric Vehicle Scheduling and Charging Problem, formally proving its NP-hardness through rigorous complexity reduction techniques.

While these foundational studies established the complexity of the single depot variant, research progressed toward the more practical multi-depot case. One of the first studies on the MDEVSP was conducted by Adler and Mirchandani [9], who proposed an integer linear programming model to solve the problem. Their model takes into account the range and charging constraints of electric vehicles, as well as the associated operational costs. They also introduced relaxation techniques to reduce the problem's complexity. This pioneering work laid the foundation for subsequent research by identifying the key challenges unique to electric vehicle scheduling, particularly how to effectively integrate charging requirements into traditional scheduling frameworks.

Another promising approach to solving the MDEVSP is the use of heuristic and metaheuristic methods. For example, Keskin and Çatay [10] proposed a genetic algorithm to solve the MDEVSP. Their method combines local search techniques with crossover and mutation mechanisms to efficiently explore the solution space. This approach has made it possible to solve large instances in reasonable time while obtaining good quality solutions. Their research was particularly valuable as it demonstrated how evolutionary algorithms could be adapted to handle the complexities of electric vehicle constraints, offering a viable alternative to exact methods for large-scale problems.

Furthermore, column generation has also been successfully applied to the MDEVSP. For example, Guo et al. [11] developed a column generation method with linear charging time. Their strategy relies on an original combination of a genetic algorithm and column generation, significantly accelerating the solution process compared to classical branch-and-price techniques. This improvement is mainly due to the replacement of branch-and-bound with a genetic algorithm, thus offering a more efficient alternative. The significance of this work lies in its innovative hybridization approach, which leveraged the strengths of both exact and heuristic methods to overcome the computational challenges posed by the MDEVSP's complex constraint structure.

Recent advancements in the field have introduced increasingly sophisticated modeling elements to better capture real-world operational complexities. The survey by Perumal et al. [12] provides a comprehensive review of electric bus scheduling problems, categorizing the literature according to problem types, solution methodologies, and the charging infrastructures considered. The identified problem types contain not only electric vehicle scheduling but also strategic and tactical planning aspects, such as investment in charging infrastructure, optimal placement of charging stations, investment planning for electric bus fleets, and charging scheduling, which involves optimizing charging costs based on time-of-use electricity prices and power loads at stations. Additionally, the review includes integrated electric bus planning, where electric vehicle technology is jointly considered with other planning problems such as line planning, timetabling, or crew scheduling. Their work traces the evolution of the field, from basic formulations to complex, integrated models that capture the full range of operational challenges in electric bus scheduling.

Furthermore, Nafstad et al. [13] developed a branch-price-and-cut algorithm that addresses two critical aspects of modern electric vehicle operations: heterogeneous recharging technologies and nonlinear recharging functions. Their approach recognizes that charging rates typically decline as battery levels increase and that different charging technologies may be available across the network.

Gerbaux et al. [14] tackled large-scale MDEVSP instances with two critical real-world constraints: piecewise linear charging functions and capacitated charging stations. Their machine-learning-based column generation heuristic employs graph neural networks to intelligently reduce network size, demonstrating remarkable efficiency improvements (over 70% reduction in computation time) while maintaining solution quality on instances with up to 2500 trips derived from real Montreal bus lines.

The most recent research has increasingly focused on modeling characteristics that closely mirror real-world electric vehicle operations. These include piecewise linear charging functions that more accurately represent battery charging behavior compared to simplified constant-rate models; explicit consideration of charging station capacity constraints that limit simultaneous vehicle charging; integration of multiple charging technologies with varying power outputs, costs, and availability across locations; and time-of-use electricity tariffs that introduce the need of charging scheduling. Gerbaux et al.'s [14] work specifically addresses the first two characteristics, demonstrating that these constraints can be effectively handled even for large-scale real-world problems through intelligent computational techniques. Similarly, Nafstad et al.'s [13] research tackles heterogeneous charging technologies and nonlinear functions, further confirming the field's progression toward more realistic modeling approaches. These and other emerging modeling features are comprehensively captured and classified in the survey by Perumal et al. [12], illustrating the depth of current research directions in electric vehicle scheduling.

The literature on MDEVSP reveals a progression toward increasingly sophisticated solution approaches that effectively balance computational tractability with solution quality. While exact methods provide theoretical optimality guarantees, their practical application to large-scale instances remains challenging. Hybrid approaches that combine the strengths of exact and heuristic methods show particular promise, especially when tailored to the specific characteristics of electric vehicle operations.

2.3 Degeneracy in the MDEVSP

Degeneracy is a common phenomenon in linear and combinatorial optimization problems, including the MDEVSP. It occurs when there are multiple feasible basic solutions that correspond to the same objective function value [3]. In the context of the MDEVSP, this means that there may be several schedule configurations and vehicle assignments that result in the same total cost, but differ in how trips are covered or in the use of charging stations. Importantly, degeneracy manifests primarily in fractional solutions, where multiple basic feasible solutions can yield identical objective values while satisfying all constraints.

Degeneracy can pose problems when solving the MDEVSP, particularly with iterative methods such as column generation. When multiple feasible basic solutions have the same objective function value, the algorithm may go through many iterations without improving the solution, which significantly slows down convergence. Barnhart et al.'s [15] comprehensive work on column generation was instrumental in identifying this issue, providing insights into how degeneracy affects solution quality and computational efficiency in large-scale optimization problems.

It can also lead to oscillations between different basic solutions, making it difficult to select which columns to add to the restricted master problem. Lübbecke and Desrosiers [16] highlight how degeneracy can create unstable dual solutions in column generation applications, which in turn affects the pricing subproblem's effectiveness in identifying promising new columns.

In the MDEVSP, degeneracy can be caused by several factors. First, schedule redundancy can lead to several different schedules for the same set of trips, each with the same total cost. For example, two schedules can cover the same trips but with slightly different charging sequences, which does not change the overall cost. Second, flexibility in vehicle assignment can create equivalent alternative solutions, as electric vehicles can often be assigned to different depots or charging stations without affecting the total cost. Finally, charging constraints can introduce multiple solutions where charging times are slightly offset, but with no impact on the total cost. Oukil et al.'s [6] analysis of these specific causes provided valuable insights for researchers developing solution approaches that could mitigate degeneracy's negative effects.

A technique commonly used to reduce the effects of degeneracy in column generation problems, including the MDEVSP, is perturbation. This method involves slightly modifying the problem constraints to prevent multiple basic solutions from having exactly the same objective function value. The idea, first formalized by Charnes [17] in his work on optimality and degeneracy in linear programming, is to introduce small variations in the constraints to break the symmetry between degenerate solutions.

In the context of the MDEVSP, perturbation can be applied by slightly modifying the task coverage constraints. For example, instead of requiring each trip to be covered exactly once, slight over-coverage or under-coverage of trips can be allowed. The degree to which coverage is relaxed is constraint-dependent, as if this were not the case, we would face the same degeneracy issues. These constraint-specific modifications introduce perturbation variables that reduce degeneracy by eliminating equivalent basic solutions and help accelerate the convergence of the column generation algorithm. This practical implementation approach, detailed by Oukil et al. [6], has proven particularly effective for transit scheduling problems and has been adopted by numerous subsequent researchers.

The use of perturbation in the MDEVSP reduces the undesirable effects of degeneracy in several ways. First, it reduces the number of iterations required by avoiding oscillations between equivalent solutions, allowing the algorithm to converge more quickly to an optimal solution [18]. Second, it improves algorithm stability by stabilizing the dual variables, which facilitates the selection of columns to add to the restricted master problem. Finally, it allows for more efficient exploration of the search space by eliminating degenerate solutions, which helps the algorithm explore the space of feasible solutions more effectively. These performance improvements, quantified in Oukil et al's [6] extensive computational experiments, demonstrated that perturbation techniques could reduce solution times by up to 40% for complex instances of vehicle scheduling problems.

The literature on degeneracy in column generation applications to the MDEVSP highlights the importance of addressing this computational challenge to develop efficient solution approaches. Perturbation techniques have proven particularly effective, offering significant reductions in solution time without compromising solution quality. These methods provide valuable tools for managing degeneracy in practical MDEVSP applications and inform our research approach.

2.4 Column generation with diving heuristic for the MDEVSP

Converting the fractional solutions obtained from column generation approaches to integer solutions represents a critical step in solving the MDEVSP effectively. This section examines diving heuristics that bridge the gap between theoretical optimality and practical implementation. These methods form the core of our research approach, which focuses specifically on enhancing the selection criteria for the diving process to improve solution quality while maintaining computational efficiency.

Indeed, an effective approach to solving the MDEVSP involves using column generation, followed by a simple yet efficient diving heuristic. In this methodology, the original integer problem is first relaxed to allow for fractional vehicle assignments, significantly reducing computational complexity. The column generation process iteratively improves this relaxed solution by generating columns with negative reduced costs until convergence is achieved. What makes this approach particularly noteworthy is the subsequent application of a straightforward diving heuristic to convert the fractional solution into a feasible integer solution. Specifically, the heuristic identifies the variable with the largest fractional value and sets it to 1, effectively committing to that vehicle schedule. This process is repeated, solving the resulting residual problem until all trips are covered [19]. While this method does not guarantee optimality, it strikes an effective balance between solution quality and computational efficiency, making it particularly suitable for large-scale MDEVSP instances. Integrating this approach with the perturbation techniques discussed earlier can enhance solution stability and accelerate convergence, providing a comprehensive framework for addressing the MDEVSP in practical applications.

2.5 Exploring alternative optimal solutions

Once an optimal solution to the master problem is obtained through column generation, the presence of multiple optimal solutions with identical objective values presents both challenges and opportunities. This section explores various methodologies for navigating the space of alternative optimal solutions to find those with desirable properties, particularly those that might lead to better integer solutions when applying diving heuristics. The ability to efficiently explore this solution space is especially relevant for the MDEVSP, where slight variations in fractional solutions can lead to significantly different integer solutions after diving.

2.5.1 Classic simplex pivoting

The traditional approach to exploring alternative optimal solutions relies on simplex pivoting techniques, with foundational work established by Dantzig in his seminal text on linear programming [20]. Dantzig demonstrated that after obtaining an optimal solution, non-basic variables with zero reduced costs can be introduced into the basis without changing the objective value, thereby producing alternative optimal solutions. This fundamental property of Linear Programs (LPs) provides the theoretical basis for systematically exploring the space of alternative optima through pivoting operations.

The pivoting process can be systematically implemented to enumerate alternative optimal solutions by exploring different combinations of basic variables. However, the classic pivoting approach has limitations when applied to large-scale problems like the MDEVSP. The number of alternative optimal solutions can be exponential, making complete enumeration computationally prohibitive. Additionally, navigating the space efficiently requires careful selection of entering and leaving variables to avoid cycling and to focus the search on promising regions of the solution space. Moreover, while these pivots could occur in the Restricted Master Problem (RMP) without generating new columns, searching the whole solution space, including the dimensions provided by the unknown variables, would be much more complex.

2.5.2 PUMP: Pushing towards integrality

The Feasibility Pump heuristic [21], initially developed for general Mixed-Integer Programming (MIP) problems, offers a more directed approach to exploring alternative optimal solutions with the specific goal of finding those closest to integrality.

The basic principle involves alternating between two solution spaces: the space of optimal linear programming solutions and the space of integer solutions. Starting with a fractional solution, the method first rounds it to create a reference integer solution (which may not be feasible for the original problem). It then solves a modified LP that minimizes the distance to this integer reference point. Achterberg et al. [22] later enhanced this approach to incorporate the original objective function, making it possible to maintain solution quality while pushing toward integrality.

The distance measure used in the pump methodology is the absolute difference between each variable's current value and its closest integer value. The objective is to minimize the sum of these individual distances across all variables. This minimization is subject to the constraints that the original objective value remains optimal and all other problem constraints are satisfied.

The exploration of alternative optimal solutions represents a promising avenue for improving the quality of integer solutions in the MDEVSP context. While classic simplex pivoting methods established by Dantzig [20] provide the theoretical foundation for identifying alternative optima, their practical application to large-scale vehicle scheduling problems is limited by computational complexity. The pump methodology offers a more targeted approach by systematically pushing fractional solutions toward integrality while maintaining optimality in the original objective space.

2.6 Contributions of the thesis

This review has traced the evolution of vehicle scheduling methodologies from the foundational MDVSP to the more complex MDEVSP, highlighting the additional constraints and considerations introduced by electric vehicle operations. The literature reveals a progression of increasingly sophisticated approaches designed to balance computational tractability with solution quality in these challenging combinatorial optimization problems.

Several key insights emerge from this examination of the literature. First, column generation has established itself as a particularly effective technique for addressing large-scale instances of both MDVSP and MDEVSP, offering a valuable framework that can be adapted to accommodate the unique constraints of electric vehicles. Second, the prevalence of degeneracy in these problems necessitates specific mitigation strategies, with perturbation techniques demonstrating considerable promise in improving algorithmic stability and convergence. Third, the conversion from fractional to integer solutions remains a critical step in practical implementations, with diving heuristics offering an attractive balance between computational efficiency and solution quality.

Our research builds upon these foundations by specifically focusing on enhancing the diving process through the strategic exploration of alternative optimal solutions. By integrating perturbation techniques with methods for navigating the space of alternative optima, we aim to develop a more robust approach to the MDEVSP that can effectively address real-world electric vehicle scheduling challenges. The methodologies examined in this review provide valuable theoretical frameworks and practical implementation strategies that inform our research direction.

As electric vehicle adoption continues to accelerate in public transportation systems, the need for efficient scheduling algorithms becomes increasingly critical. The literature suggests that hybrid approaches—combining the theoretical rigor of exact methods with the practical advantages of heuristic techniques—offer particularly promising avenues for future research. Our work contributes to this emerging paradigm by enhancing the bridge between relaxed optimal solutions and practical integer implementations through improved diving methodologies.

CHAPTER 3 PROBLEM DEFINITION AND SOLUTION METHOD

This chapter introduces the MDEVSP and outlines the methodological framework used for its solution. We begin with a description of the problem, highlighting the key constraints and objectives that differentiate it from conventional vehicle scheduling problems. We then present the complete mathematical formulation that serves as the foundation for our solution approach. Following this, we explain the column generation approach with a diving heuristic used to tackle the problem's complexity, with particular attention to the structure of the master problem and the relaxation of integrality constraints. Finally, we discuss existing branching heuristics that have been applied to this problem class, setting the stage for the alternative decisions proposed in subsequent chapters.

3.1 Description

The MDEVSP addresses the daily scheduling of electric buses in public transit systems. Consider a set $\mathcal{T}$ containing n transit tasks that must be serviced. Each task $t \in \mathcal{T}$ is defined by five key parameters: origin terminal α_t, destination terminal ω_t, departure time δ_t, arrival time λ_t, and energy requirement ϵ_t. Throughout this paper, we use the terms *tasks* and *trips* interchangeably.

The transit agency operates a homogeneous fleet of electric buses distributed across multiple depots represented by set $\mathcal{K}$. Each depot $k \in \mathcal{K}$ houses g^k buses at the beginning of the operational day. All vehicles begin fully charged at the maximum State of Charge (SoC) level $\overline{S}$ and must maintain their SoC within the operational range $[\underline{S}, \overline{S}]$ throughout service delivery.

To maintain operational continuity, a network of charging facilities represented by set $\mathcal{H}$ is available. Each charging facility $h \in \mathcal{H}$ is equipped with b^h charging units, limiting the number of concurrent charging buses. The charging process follows a piecewise linear function, where the final SoC S^f depends on the initial SoC S^i and the charging duration Δ, expressed as $S^f = g(S^i, \Delta)$.

For precise tracking of charging infrastructure utilization, the operational day is segmented into $|\mathcal{P}|$ consecutive time intervals of equal duration θ. The set of time intervals is denoted by $\mathcal{P} = \{p_1, p_2, \ldots, p_{|\mathcal{P}|}\}$, where each interval $p \in \mathcal{P}$ has start time t_p^s and end time t_p^e, with $t_{p_j}^e = t_{p_{i+1}}^s$ for all $i \in \{1, 2, \ldots, |\mathcal{P}| - 1\}$. To simplify the process, charging activities must begin at the start of a time interval and span an integer number of intervals. When a vehicle achieves maximum charge before the completion of its final charging interval, it continues to occupy the charging unit until the interval concludes.

The transportation network connects various locations including depots, charging facilities, and task terminals. For any location pair (i, j), we define τ_{ij} as the deadheading travel time and ϵ_{ij} as the corresponding energy consumption.

A valid bus schedule consists of a sequence of serviced tasks, deadheading movements, and charging operations that forms a feasible schedule beginning and ending at the same depot. Schedule feasibility requires following task timetables and maintaining SoC within permissible limits at all times. To enhance driver efficiency, we impose a maximum idle time of μ minutes between consecutive tasks when no charging or depot return occurs. Buses may temporarily return to any depot during operational hours but must remain there for at least ν minutes if they do so.

The cost structure contains:

- A fixed deployment cost c^F per vehicle used
- An idle time cost c^W per minute spent waiting outside depots
- Variable deadheading costs c^D_{ij} between locations i and j
- A penalty c^R for each mid-day depot return
- A penalty c^H for each charging operation

No idle time costs are incurred at depots since vehicles can remain unattended, and task service costs are omitted as they represent constant operational expenses. The penalties c^R and c^H serve to discourage unnecessary depot returns and charging events.

The MDEVSP's complexity stems from the interaction between temporal constraints, spatial relationships, and energy management requirements. The challenge is even more present in real-world scenarios involving hundreds of tasks, multiple depots, and limited charging infrastructure. The objective is to construct a set of feasible bus schedules that services each task exactly once while minimizing total operational costs and respecting all depot and charging facility capacity constraints. Our solution approach, detailed in the next section, employs column generation techniques to efficiently handle the large-scale combinatorial nature of this problem.

3.2 Mathematical formulation

For our mathematical model, we extend the notation introduced in Section 3.1. Let Ω^k represent the set of valid schedules starting and ending at depot $k \in \mathcal{K}$. For each schedule $s \in \Omega^k$, we define c_s as the total cost of schedule s, which incorporates all operational costs described earlier. We also define a_s^t as a binary parameter equal to 1 if task $t \in \mathcal{T}$ is covered in schedule s and 0 otherwise, and $o_s^{p,h}$ as a binary parameter equal to 1 if schedule s includes a charging operation at station $h \in \mathcal{H}$ during time interval $p \in \mathcal{P}$ and 0 otherwise. The decision variables in our model are represented by x_s, a binary variable equal to 1 if schedule s is selected in the solution and 0 otherwise.

The MDEVSP can be mathematically expressed as follows:

$$\min \sum_{k \in \mathcal{K}} \sum_{s \in \Omega^k} c_s x_s \tag{3.1}$$

$$\text{subject to} \quad \sum_{k \in \mathcal{K}} \sum_{s \in \Omega^k} a_s^t x_s = 1, \qquad \forall t \in \mathcal{T} \tag{3.2}$$

$$\sum_{s \in \Omega^k} x_s \leq g^k, \qquad \forall k \in \mathcal{K} \tag{3.3}$$

$$\sum_{k \in \mathcal{K}} \sum_{s \in \Omega^k} o_s^{p,h} x_s \leq b^h, \qquad \forall h \in \mathcal{H},\ \forall p \in \mathcal{P} \tag{3.4}$$

$$x_s \in \{0,1\}, \qquad \forall k \in \mathcal{K},\ \forall s \in \Omega^k \tag{3.5}$$

The objective function (3.1) minimizes the total cost of all selected schedules across all depots. Constraints (3.2) ensure that each task is covered exactly once by a bus, which is essential for meeting all service requirements. Constraints (3.3) respect the fleet size limitations at each depot, ensuring that we do not utilize more vehicles than are available at each depot k. Constraints (3.4) enforce the capacity limitations of charging stations during each time interval, preventing more simultaneous charging operations than available charging ports at each station. Finally, constraints (3.5) enforce the binary nature of the decision variables.

3.3 Solution methodology

To address the computational challenges of the MDEVSP, we employ a column generation approach embedded within a diving heuristic framework. Column generation [15] provides an efficient mechanism for handling the vast number of potential vehicle schedules without explicitly enumerating them, while the diving heuristic facilitates the conversion of fractional solutions to integer solutions required for practical implementation.

The solution process begins with formulating the set-partitioning master problem (3.1)–(3.5) that ensures each trip is covered exactly once. Due to the problem's complexity, we relax the integrality constraints to obtain a linear programming formulation that can be solved more efficiently.

Since the complete enumeration of all potential schedules in Ω^k would be too large, we use column generation to dynamically generate promising schedules. The process alternates between solving the restricted master problem (which contains only a subset of potential schedules) and solving pricing subproblems.

The pricing component of our approach involves solving multiple subproblems, one for each depot in the system. Each subproblem is responsible for constructing feasible bus schedules that can be operated from its associated depot. Following the methodology of Irnich et Desaulniers. [23], these pricing problems are formulated as Shortest Path Problems with Resource Constraints (SPPRC) on acyclic networks.

In each SPPRC subproblem, resources are carefully managed to ensure operational feasibility, particularly regarding the electric buses. Specifically, these resources track the state-of-charge of each bus to guarantee it remains within permissible lower and upper bounds throughout its schedule. The formulation also enforces that a recharge operation occurs each time a bus visits a recharging station, and that during each such visit, there is exactly one recharge event spanning a consecutive number of time intervals. This approach handles the complex recharging logistics inherent in electric bus operations.

For solving these SPPRC subproblems efficiently, a specialized labeling algorithm was implemented as detailed in Gerbaux et al. [14]. This algorithm follows the principles outlined in Irnich and Desaulniers [23], beginning with an initial partial schedule containing only the source node o_d and iteratively extending schedules until reaching the sink node k_d. To manage computational complexity, the algorithm employs strategic pruning techniques, eliminating infeasible schedules that violate resource constraints and schedules that are proven to be dominated by other more promising alternatives. This approach significantly reduces the search space while still guaranteeing the identification of optimal solutions for each subproblem, ultimately generating columns with negative reduced costs for the master problem.

To address the inherent degeneracy often encountered in vehicle scheduling problems, we incorporate perturbation techniques that modify constraints (3.2) by adding a pair of slack and surplus variables to each covering constraint. While the right-hand side remains at 1, each slack and surplus variable is assigned a small, randomly generated upper bound, allowing for controlled constraint violations. This approach breaks the symmetry that typically contributes to degeneracy by introducing unique perturbations to each constraint. Although minor violations are permitted, the small values of perturbation variables preserve practical solution quality while providing the algorithm with clear preferences that stabilize dual variables and improve column generation convergence. The complete formulation is detailed in model (27)–(32) by Desfontaines and Desaulniers [24].

Once column generation converges to an optimal solution for the master problem, we employ a diving heuristic to obtain an integer solution, as illustrated in Figure 3.1. This figure presents a flowchart of the entire column generation process with the diving heuristic integration.

As shown in the flowchart, the process begins with initial columns feeding into the RMP. The column generation cycle involves solving the RMP with the current set of columns, followed by solving the pricing subproblems to identify new columns with negative reduced costs. This cycle continues until no more promising columns are generated.

The algorithm evaluates the quality of the master problem solution by comparing it against the best integer solution found so far (if one exists). If the master problem solution has a worse objective value than the best integer solution, the algorithm terminates and returns the best integer solution as the final result. However, in most cases, the master problem solution contains fractional variables and provides a better objective value than any previously discovered integer solution, prompting the algorithm to continue the search process.

This is where the diving heuristic comes into play. Instead of implementing a full branch-and-bound tree exploration, which would be computationally prohibitive for large instances, the diving heuristic makes a decision by fixing to one the values of some variables to reduce the search space. The different variable fixing techniques used are detailed in Section 3.4, and once the fixing decisions are made, the algorithm returns to the column generation process with these additional constraints.

The heuristic effectively commits to a partial solution and then regenerates columns that are compatible with this commitment. This creates a progressively reduced solution space with each iteration, gradually building a complete integer solution.

It is important to note that the diving heuristic does not guarantee the discovery of a feasible integer solution. The successive variable fixing decisions may lead to an infeasible problem, particularly when the problem instance has tight constraints or limited resources. For example, if the fleet size is insufficient to cover all required tasks, the diving process may reach a state where no feasible schedules can be selected to satisfy the remaining tasks.

What makes this approach particularly effective for the MDEVSP is that it balances com-

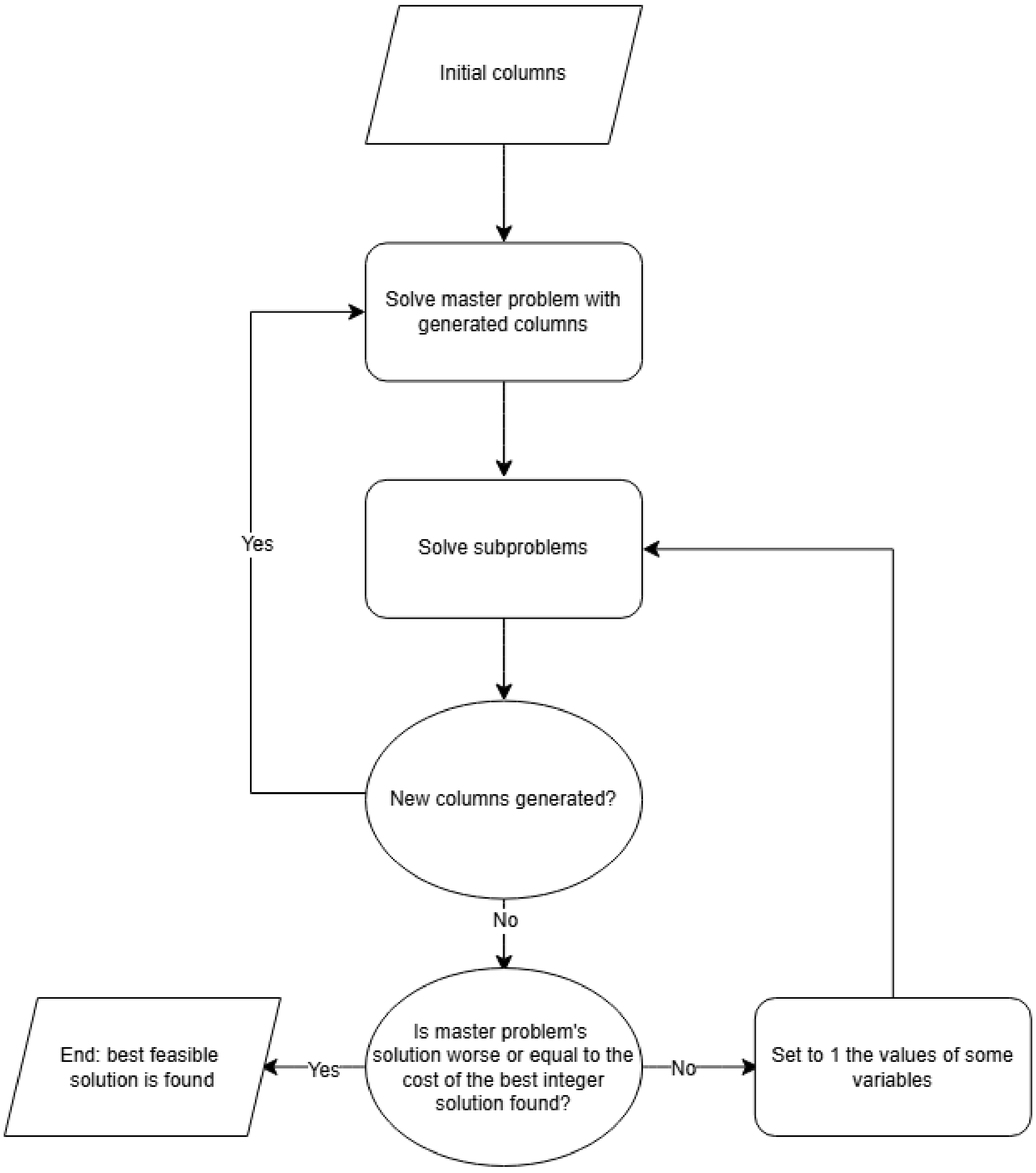

Initial columns
Solve master problem with generated columns
Solve subproblems
New columns generated?
Yes
No
Is master problem's solution worse or equal to the cost of the best integer solution found?
Yes
No
End: best feasible solution is found
Set to 1 the values of some variables

Figure 3.1 Flowchart of the column generation algorithm with diving heuristic

putational gains with solution quality, allowing us to handle large-scale instances that would be impractical to solve with exact methods. Figure 3.1 clearly illustrates how the fixing decisions are integrated with the column generation framework, showing the cyclical nature of the solution process and the critical decision points that guide the algorithm toward integer solutions.

The key advantage of this methodology lies in its ability to decompose the complex MDEVSP into more manageable components while handling the initially large solution space. The column generation approach tackles the schedule feasibility constraints (including battery limitations and charging requirements) within the pricing subproblems, while the master problem coordinates the overall trip assignment, vehicle utilization, and charging resource allocation.

In the following section, we discuss specific branching heuristics that have been developed to guide the conversion from fractional to integer solutions in this framework.

3.4 Existing branching heuristics

Converting the fractional solutions obtained from the master problem into feasible integer solutions presents a significant challenge in the MDEVSP. While branch-and-price algorithms could theoretically find optimal integer solutions, their computational requirements become prohibitive for large-scale instances. Instead, diving heuristics offer a more practical approach by making sequential branching decisions that progressively construct an integer solution. This section describes three key heuristic variable fixing techniques that have already been developed to guide this process. These fixing techniques can either be used on their own or can be combined with each other.

3.4.1 Column fixing

The most straightforward approach is column fixing, which identifies fractional variables with large fractional values and fixes them to one in the master problem. The underlying idea is that variables with values close to one are likely part of a good integer solution, making them promising candidates.

In its basic implementation, column fixing selects the variable with the largest fractional value in each iteration. A more sophisticated variant allows fixing multiple columns simultaneously if they exceed a predefined threshold (typically around 0.7) and if the sum of their differences from 1.0 does not exceed a certain parameter. This parameter helps control the aggressiveness of the fixing strategy.

While column fixing offers computational efficiency by quickly reducing the problem size, it may lead to suboptimal or infeasible solutions when early fixing decisions propagate suboptimal decisions through subsequent iterations. Since these decisions are irreversible, a poor early choice can significantly degrade the final solution quality. Additionally, the greedy nature of selecting the largest fractional values does not account for potential interactions with columns not yet generated or the impact on future iterations of column generation that follow each fixing decision.

3.4.2 Inter-task fixing

A more nuanced approach is inter-task fixing, which operates at a finer granularity by focusing on sequences of consecutive tasks (inter-tasks) rather than entire columns. This technique examines the fractional columns and aggregates values for each inter-task across all columns. When the cumulative value for an inter-task exceeds a threshold, it is fixed in the solution, effectively imposing this task sequence in all future iterations.

Similar to column fixing, multiple inter-tasks can be fixed simultaneously if they exceed the threshold and their cumulative difference from 1.0 remains below a certain parameter. The advantage of inter-task fixing is its ability to identify common elements across multiple fractional columns, potentially capturing structural patterns that might be missed when considering columns in isolation.

This approach provides greater flexibility than column fixing, as it preserves more of the solution space while still making meaningful progress toward integrality. The implementation of these inter-task fixing constraints is achieved by adding appropriate resource constraints in the subproblems as detailed by Irnich and Desaulniers [23]. However, it introduces additional complexity in tracking and enforcing inter-task assignments across iterations. Furthermore, fixing certain inter-tasks may also lead to suboptimal solutions, as once a poor decision is made, all future columns generated will be negatively impacted by this decision.

3.4.3 Retrospective branching

Addressing the irreversibility limitation of previous techniques, retrospective branching introduces a mechanism to reconsider potentially risky fixing decisions. It begins similarly to column fixing by selecting variables with large fractional values, but it maintains a set of "risky" columns.

When there are no columns with a fractional value greater than or equal to a threshold t, the column with the largest fractional value is selected and declared as a risky column. These risky columns are not immediately fixed to 1 as in traditional column fixing, but instead are collectively managed through constraint enforcement.

This approach was explored by Quesnel et al. [25] for the crew pairing problem but can also be transferred to the MDEVSP solution as both contain a column generation process with the use of techniques such as column fixing.

Initially, the algorithm adds a constraint to the master problem specifying that the sum of the risky column variables must equal the number of risky columns, effectively fixing all these columns to 1. However, when the optimal value of a linear relaxation exceeds the sum of the root node bound and a predetermined threshold, the algorithm creates a sibling node where the right-hand side of this constraint is reduced to the number of risky columns minus one. This modification allows the algorithm to revise one of its previous decisions.

Once the sibling node is created, the exploration can continue in both branches, and similar additional branching decisions can be performed as needed, possibly requiring new constraints when the set of risky columns differs between branches. The effectiveness of retrospective branching depends on parameters that determine how many columns to revise and what criteria define a risky column. These parameters allow fine-tuning the balance between solution quality and computational effort.

What makes retrospective branching particularly noteworthy is that the additional constraint it introduces modifies the dual values in the master problem, potentially generating entirely different columns during subsequent column generation phases. This characteristic leads to exploring significantly different regions of the solution space compared to traditional column fixing.

However, even if retrospective branching allows for revising a decision made previously during the solution process, it does not remove the impact that this decision had on other columns generated between the time it was considered a risky column and the time it was revised.

3.4.4 Comparative considerations

Each of these techniques represents a different trade-off between computational efficiency and solution quality. Column fixing offers the most straightforward implementation but may lead to suboptimal solutions due to its decision making. Inter-task fixing provides greater flexibility by working at a finer granularity. Retrospective branching addresses the irreversibility issue but requires more time to find a suitable integer solution.

The presence of these diverse approaches highlights a fundamental challenge in solving the MDEVSP: while column generation efficiently handles the linear relaxation of the problem, the conversion to integer solutions introduces significant complexity. The irreversible nature of branching decisions in diving techniques means that early choices fundamentally constrain the solution space for all subsequent iterations.

This propagation effect demonstrates the importance of making informed branching decisions that consider not just the current fractional values but also their potential impact on future iterations. The exploration of alternative fractional solutions, as proposed in the following chapter, aims to address this challenge by identifying branching candidates with more favorable long-term consequences.

CHAPTER 4 ALTERNATIVE FRACTIONAL SOLUTIONS

This chapter introduces and explores methodologies for finding alternative fractional solutions to the master problem. The core premise of our research is that even with identical objective values, different fractional solutions might lead to substantially different integer solutions when subject to the same column fixing algorithm described in Chapter 3. By identifying and evaluating these alternative solutions, we aim to improve the quality of the final integer solution without significantly increasing computational complexity.

A key challenge in exploring alternative optimal solutions arises from the perturbation techniques incorporated to address degeneracy. While these perturbations successfully stabilize the dual variables and improve convergence during column generation, they also have the effect of significantly reducing the presence of alternative optimal solutions. In many cases, the perturbation leads to a unique optimal solution in the master problem, eliminating the possibility of finding alternative solutions with identical objective values.

To overcome this limitation, we introduce a controlled deterioration factor that allows for a slight increase in the objective value. This approach intentionally widens the search space to include near-optimal solutions that might have more favorable chosen properties for the fixing process. The deterioration factor creates a tolerance band around the optimal objective value, within which alternative solutions are considered acceptable.

The theoretical impact of this deterioration factor is multifaceted:

- Expanded solution space: As the deterioration factor increases, the feasible region of acceptable solutions grows exponentially. This provides access to a richer set of alternative solutions with potentially diverse fractional patterns.
- Structural diversity: Solutions within the tolerance band often exhibit significantly different structural characteristics despite their proximity in objective value.
- Balancing quality and diversity: The deterioration factor introduces a fundamental trade-off between solution quality (proximity to the optimum) and solution diversity. A deterioration factor too small may not have sufficiently different alternatives, while a factor too large may incorporate solutions that are structurally inferior despite their mathematical feasibility.
- Computational considerations: Larger deterioration factors increase the computational effort required to explore the expanded solution space effectively.

The techniques proposed in this chapter operate on the premise that introducing this controlled deterioration can ultimately lead to better final integer solutions. Before applying these techniques, we preprocess the problem by removing the perturbation variables that were introduced during the column generation phase. While these perturbation variables were essential for stabilizing the column generation process and avoiding degeneracy issues, they are no longer needed when exploring alternative solutions since no new columns are being generated at this stage. Removing these perturbation variables reduces the number of variables that need to be considered, thereby improving computational efficiency without compromising solution quality.

In the following sections, we introduce different approaches for exploring this expanded solution space: probing techniques that evaluate promising candidates through linear programming, and specialized MIP formulations designed to identify alternatives with specific structural properties. All approaches aim to discover alternative fractional solutions that, when subject to the column fixing algorithm described in Chapter 3, might lead to different outcomes. The goal of this research is to investigate whether searching for solutions with these different properties actually improves the final integer solutions. Regardless of the outcome, these techniques will certainly explore different regions of the solution space, potentially offering new insights into the structure of efficient solutions for the MDEVSP.

4.1 Candidate selection by probing

Probing is a technique that evaluates individual candidate columns to predict their potential effect on solution quality when fixed. This approach leverages the computational efficiency of linear programming to explore numerous alternative solutions without incurring the substantial computational overhead associated with MIP formulations.

The fundamental concept behind probing is straightforward yet powerful: using the existing pool of non-perturbation columns from the master problem, we identify promising candidate columns that might lead to good integer solutions when fixed to one. For each candidate column, we solve a dedicated LP that maximizes its value while maintaining feasibility within a restricted gap from the optimal objective value. The candidates are then ranked based on their resulting fractional values, with higher values indicating greater potential for improving the integer solution when fixed to one.

A key advantage of the probing approach lies in its computational efficiency. While solving multiple LPs requires additional computation time compared to a single LP, this overhead remains modest relative to the computational cost of solving mixed-integer programs. This efficiency allows us to evaluate numerous candidates within a reasonable time frame, exploring the solution space for promising alternatives. These LPs can also easily be parallelized which would reduce the computation time.

The implementation of probing involves two main steps:

1. Candidate selection: Rather than evaluating all possible columns, which would be computationally prohibitive, we focus on promising candidates based on specific criteria. Two primary selection strategies are employed:
    - Threshold-based selection, which identifies all columns with fractional values above a certain threshold (e.g., 0.15)
    - Rank-based selection, which considers the top columns with the largest fractional values (e.g., the 100 largest columns)
2. Sequential evaluation: For each selected candidate, we solve a dedicated LP that maximizes the value of that column while maintaining solution quality (full formulation in Section 4.1.1). This initial evaluation may sometimes result in multiple columns with nearly identical objective values. In such cases, we can conduct a secondary evaluation using alternative objective functions (detailed in Section 4.1.2) to differentiate between these closely ranked candidates.

This two-stage approach allows us to efficiently narrow down the search space to a manageable number of promising alternatives while maintaining the ability to distinguish between candidates with similar primary characteristics. By strategically limiting the number of LPs solved, probing provides a computationally viable technique for exploring alternative solutions without sacrificing the potential to identify structurally interesting column choices

The following subsections detail the specific linear programming formulations used in the primary and secondary evaluations of candidate columns.

4.1.1 Maximizing fractional values

The primary probing technique evaluates each candidate column by determining how large its value could become in an alternative solution while maintaining near-optimality. For this evaluation, in addition to the notation introduced in Section 3.2, we define the following terms, which will be used in the formulation below:

- s_k: The schedule corresponding to the candidate column being evaluated

- $\tilde{x}_s$, $\forall s \in \Omega^k$: Decision variables that are continuous between 0 and 1 due to the linear relaxation of the master problem
- z^*: Optimal objective value of the original master problem
- d: Deterioration factor, typically set to a small value (e.g., 0.00001 or 0.00005)

For each candidate column k selected through the threshold or rank-based approach, we solve the following LP that maximizes the value of this specific column:

$$\begin{aligned}
\max \quad & \tilde{x}_{s_k} && && (4.1)\\
\text{subject to} \quad & (3.2)\text{–}(3.4) && && \\
& \sum_{k \in \mathcal{K}} \sum_{s \in \Omega^k} c_s \tilde{x}_s \leq (1+d)\, z^* && && (4.2)\\
& \tilde{x}_s \in [0,1], && \forall s \in \Omega^k,\ \forall k \in \mathcal{K} && (4.3)
\end{aligned}$$

In this formulation, constraints (3.2)–(3.4) ensure that the solution of this LP is also a feasible solution of the master problem. Constraint (4.2) limits the potential deterioration in objective value, allowing the solution to deviate slightly from optimality by a factor of d. Finally, constraints (4.3) maintain the bounds on all continuous decision variables in the problem.

The intuition behind this formulation is straightforward: we seek to identify alternative solutions where a specific candidate column can have a substantially larger value than in the original solution, potentially making it a more appropriate choice for fixing to one in the diving heuristic. By solving this LP for each candidate column, we can rank them according to their maximum potential value, prioritizing those that can achieve values closer to one. The maximum value achieved for each candidate column provides valuable information about its potential importance in alternative solution structures.

The computational efficiency of this approach stems from the fact that we reuse the existing column pool without generating new columns. Furthermore, since we focus only on promising candidates rather than all columns, the number of LPs to solve remains manageable even for large-scale instances.

After evaluating all candidate columns with this primary objective, we select the column that achieved the largest maximum value as the most promising candidate for fixing. However, in cases where multiple columns achieve very similar maximum values (for example, within

0.01 of each other), additional criteria can be applied to differentiate between these nearly equivalent candidates. These secondary criteria are discussed in the following section.

4.1.2 Secondary objectives

When the primary probing technique identifies multiple columns with similar maximum potential values, additional criteria can help differentiate between these candidates. We implement three distinct secondary objectives to provide this differentiation, each exploring different aspects of solution quality and structural characteristics.

Maximizing the average value of other candidate columns

The first secondary objective aims to identify solutions where not just one but multiple promising columns have large values. This objective is based on the hypothesis that solutions with several large-value columns might offer greater flexibility for subsequent fixing decisions in the diving heuristic.

For this evaluation, we use the previously defined notation and add the following terms:

- $\tilde{x}_{s_k}^{\max}$: The maximum value achieved for column s_k in the primary probing evaluation
- $\mathcal{C}$: The set of candidate columns evaluated in the initial probing phase

The formulation for this secondary objective is:

$$\begin{aligned}
\max \quad & \frac{1}{|\mathcal{C}|-1} \sum_{\substack{i \in \mathcal{C} \\ i \neq s_k}} \tilde{x}_i && && (4.4) \\
\text{subject to} \quad & (3.2)\text{–}(3.4),\ (4.2) && && \\
& \tilde{x}_{s_k} = \tilde{x}_{s_k}^{\max} && && (4.5) \\
& \tilde{x}_s \in [0,1], && \forall s \in \Omega^k,\ \forall k \in \mathcal{K} && (4.6)
\end{aligned}$$

In this formulation, constraints (3.2)–(3.4) ensure that the solution is feasible for the master problem. Constraint (4.2) limits the deterioration in objective value compared to the optimal solution. Constraint (4.5) fixes the value of the candidate column being evaluated to its maximum value determined in the primary probing phase. Finally, constraints (4.6) maintain the bounds on all continuous decision variables.

This approach intentionally seeks solutions where multiple columns have relatively large values. The underlying assumption is that having a cluster of promising columns rather than a single dominant one might provide more robust branching options throughout the diving process.

Maximizing the relative improvement from the original solution

The second secondary objective focuses on columns that show the most significant proportional growth from their values in the original master problem solution. This approach prioritizes columns that demonstrate substantial potential for improvement when given the opportunity to grow.

For this objective, we rank candidates based on the ratio between their maximum potential value and their original value:

$$\text{Improvement Ratio} = \frac{\tilde{x}_{s_k}^{\text{max}}}{\tilde{x}_{s_k}^{\text{original}}}$$

Where:

- $\tilde{x}_{s_k}^{max}$ is the maximum value achieved for column s_k in the primary probing evaluation.
- $\tilde{x}_{s_k}^{\text{original}}$ is the value of the column s_k in the initial solution of the master problem.

No additional LP is required for this evaluation, as it simply compares the results from the primary probing with the original solution values. This metric identifies columns that might have been underrepresented in the original solution but show considerable potential for importance in alternative solution structures.

The underlying hypothesis is that columns with large improvement ratios might represent structural components that were suppressed in the original solution due to degeneracy or the specific path taken during column generation, but which could lead to better integer solutions when prioritized.

Minimizing objective value with fixed column

The third secondary objective provides the most direct evaluation of a column's impact on solution quality. For each candidate column that achieved a similar maximum value in the primary probing, we evaluate the effect of fully committing to this column in the solution.

This approach uses the linear relaxation of the original master problem (3.1)–(3.5) augmented by the additional constraint:

$$\tilde{x}_{s_k} = 1. \tag{4.7}$$

By forcing each candidate column completely into the solution and evaluating the resulting objective value, this technique directly measures the immediate impact of committing to each column. The column that produces the smallest increase in the objective value when fixed to one is considered the most promising candidate.

This approach simulates fixing the value of the column to one for each candidate column, providing a preview of their immediate impact on solution quality. The underlying assumption is that columns causing minimal objective deterioration when fixed to one are likely to be part of high-quality integer solutions.

Unlike the other secondary objectives, this approach does not maintain the deterioration constraint, as fixing to one might necessarily cause a greater deterioration than the tolerance allowed in the primary probing phase. This reflects the reality of the diving heuristic, where forcing variables to integer values increases significantly the objective value compared to the optimum.

Each of these secondary objectives explores different aspects of solution quality and structural characteristics, providing alternative perspectives for differentiating between otherwise similar candidate columns. By combining these approaches, particularly when the primary probing results are inconclusive, we gain a more comprehensive understanding of each column's potential contribution to the final integer solution.

4.2 Candidate selection by mixed-integer programming

While the probing techniques described in the previous section offer computational efficiency by solving a series of LPs, they evaluate candidate columns individually rather than considering their collective interactions. MIP formulations provide an alternative approach that can capture these interactions by optimizing over the entire set of columns simultaneously, potentially identifying alternative fractional solutions with structural properties that might be missed by the more localized probing techniques.

However, MIP formulations introduce significant computational challenges. The solution space for the MDEVSP is vast, with hundreds or thousands of columns generated during the column generation process. Solving mixed-integer programs over this entire column space would be computationally prohibitive. To manage these challenges, we implemented several filtering techniques and imposed reasonable time limits on the solution process.

First, we restricted the MIP formulations to consider only columns with zero reduced cost in the optimal solution of the master problem, as these columns define the space of alternative optimal solutions. We further expanded this set by including a limited number of columns with small positive reduced costs, using a threshold tied to the deterioration factor. This approach allowed us to explore near-optimal solutions while maintaining a manageable problem size.

Despite these filtering techniques, the resulting mixed-integer programs remained computationally intensive for many instances. Time limits were necessary to ensure the practical applicability of these techniques within the overall solution framework.

The following subsections detail three specific MIP formulations we developed to explore different aspects of alternative fractional solutions. Each formulation targets a particular structural property that might lead to improved integer solutions when subject to the column fixing procedure.

4.2.1 Maximizing the best fractional value

The first MIP formulation focuses on identifying the alternative fractional solution that maximizes the value of the largest fractional variable. This approach aligns conceptually with the primary probing technique but considers all columns simultaneously rather than evaluating them individually. The main difference is that all columns present in the MIP are considered as candidates for maximization, whereas in the probing technique, only the variables with the largest fractional values from the master problem solution are selected as candidates.

For this formulation, we introduce the following additional notation:

- w: An auxiliary variable representing the maximum value among all fractional columns
- y_s: Binary variables equal to 1 if the variable $\tilde{x}_s$ has the largest fractional value, 0 otherwise.

This mixed-integer program can be formulated as follows:

$$\max \quad w \tag{4.8}$$

$$\begin{aligned}
\text{subject to} \quad & (3.2)\text{–}(3.4),\ (4.2) & & \\
& w \leq \tilde{x}_s + (1 - y_s), & \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \quad & (4.9) \\
& \sum_{k \in \mathcal{K}} \sum_{s \in \Omega^k} y_s = 1 & & (4.10) \\
& \tilde{x}_s \in [0, 1], & \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \quad & (4.11) \\
& y_s \in \{0, 1\}, & \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \quad & (4.12) \\
& w \in [0, 1] & & (4.13)
\end{aligned}$$

In this formulation, constraints (3.2)–(3.4) and (4.2) play the same role as before. Constraints (4.9) establish the relationship between the auxiliary variable w and the decision variables $\tilde{x}_s$, using binary variables y_s to select exactly one column to maximize. When $y_s = 1$ (indicating the column is selected for maximization), the constraint simplifies to $w \leq \tilde{x}_s$, forcing w to be no greater than the value of the selected column. For all non-selected columns (where $y_s = 0$), the constraint becomes $w \leq \tilde{x}_s + 1$, which is always satisfied given that $\tilde{x}_s$ and w are bounded by 1.

Constraint (4.10) ensures that exactly one column is selected for maximization. Finally, constraints (4.11)–(4.13) define the domains of the variables.

By maximizing the largest fractional value in the solution, we aim to identify variables that are stronger candidates for fixing in the diving heuristic.

This formulation addresses the objective of finding alternative fractional solutions where at least one column has a large value, making it a stronger candidate for fixing in the diving heuristic. Unlike the probing approach, which evaluates columns individually, this MIP formulation considers the inter-dependencies between all columns simultaneously, potentially identifying solutions with structural properties that might be overlooked by more localized techniques.

The computational challenge associated with this formulation stems from the introduction of binary variables y_s for each column in the filtered set. To manage this complexity, implementation includes filtering, as mentioned at the beginning of Section 4.2, to consider only columns with zero or near-zero reduced costs, as well as imposing a reasonable time limit on the solution process.

4.2.2 Minimizing the sum of the difference to the bounds

The second MIP formulation aims to identify alternative fractional solutions where column values are collectively closest to integer values (either 0 or 1). This approach draws inspiration from the feasibility pump heuristic and the PUMP methodology [21] described in Section 2.5.2, which alternates between finding linear programming feasible solutions and integer feasible solutions to guide the search toward integer feasibility.

For this formulation, we introduce the following additional notation:

- t_s: An auxiliary variable representing the absolute difference between $\tilde{x}_s$ and its closest integer value
- r_s: Binary variables representing the closest integer value (0 or 1) to the continuous variable $\tilde{x}_s$

The mathematical formulation for this approach can be expressed as follows:

$$
\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}} \sum_{s\in\Omega^k} t_s && && (4.14)\\
\text{subject to} \quad & (3.2)\text{–}(3.4),\ (4.2) && &&\\
& t_s \geq \tilde{x}_s - r_s, && \forall s \in \Omega^k,\ \forall k \in \mathcal{K} && (4.15)\\
& t_s \geq r_s - \tilde{x}_s, && \forall s \in \Omega^k,\ \forall k \in \mathcal{K} && (4.16)\\
& \tilde{x}_s \in [0,1], && \forall s \in \Omega^k,\ \forall k \in \mathcal{K} && (4.17)\\
& r_s \in \{0,1\}, && \forall s \in \Omega^k,\ \forall k \in \mathcal{K} && (4.18)\\
& t_s \in [0,1], && \forall s \in \Omega^k,\ \forall k \in \mathcal{K} && (4.19)
\end{aligned}
$$

In this formulation, constraints (3.2)–(3.4) and (4.2) play the same role as before. Constraints (4.15) and (4.16) work together to define the relationship between the continuous variables $\tilde{x}_s$, binary variables r_s, and auxiliary variables t_s. These constraints ensure that $t_s = |\tilde{x}_s - r_s|$, effectively capturing the absolute difference between each decision variable and its nearest integer value.

The objective function (4.14) minimizes the sum of these absolute differences across all columns, effectively pushing the fractional solution toward integrality while maintaining feasibility and near-optimality.

The hypothesis underlying this formulation is that solutions with column values closer to integer values might provide better starting points for the column fixing process, potentially leading to higher quality integer solutions with fewer diving iterations and number of columns generated after each variable fixing. By identifying near-optimal fractional solutions that already have a structure closer to integrality, we aim to reduce the gap between the solution of the linear relaxation and the final integer solution.

A significant challenge with this approach arises when all columns have fractional values below 0.5 in the optimal solution of the master problem. In such cases, the binary variables r_s would all be set to 0, and the formulation would essentially push all fractional values toward 0 rather than identifying promising columns to fix to 1. This situation, common in highly degenerate problems with many columns covering similar sets of tasks, could contradict the initial idea of the approach by failing to identify meaningful structural characteristics.

This MIP formulation, despite its potential limitations, provides a different perspective on alternative fractional solutions compared to the other techniques. By focusing on the collective integrality of the solution rather than maximizing individual column values, it explores a different structural characteristic that might yield valuable insights into the solution space of the MDEVSP.

4.2.3 Maximizing the values of columns above a threshold

The third MIP formulation takes a different approach by seeking to maximize the cumulative value of all columns that exceed a predefined threshold. This strategy aims to increase the number of large value columns available as branching candidates at each node of the diving heuristic, potentially providing more robust options for column fixing decisions.

For this formulation, we introduce the following additional notation:

- τ: A parameter representing the minimum value that a column must achieve to be counted in the objective function
- v_s: An auxiliary variable representing the contribution of column $\tilde{x}_s$ to the objective function
- u_s: Binary variables indicating whether column $\tilde{x}_s$ exceeds the threshold τ

The mathematical formulation for this approach is:

$$\max \quad \sum_{k\in\mathcal{K}} \sum_{s\in\Omega^k} v_s \tag{4.20}$$

$$\text{subject to} \quad (3.2)\text{–}(3.4),\ (4.2)$$

$$\tilde{x}_s \geq \tau\, u_s, \qquad \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \tag{4.21}$$

$$\tilde{x}_s \geq v_s, \qquad \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \tag{4.22}$$

$$u_s \geq v_s, \qquad \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \tag{4.23}$$

$$\tilde{x}_s \in [0,1], \qquad \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \tag{4.24}$$

$$u_s \in \{0,1\}, \qquad \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \tag{4.25}$$

$$v_s \in [0,1], \qquad \forall s \in \Omega^k,\ \forall k \in \mathcal{K} \tag{4.26}$$

In this formulation, constraints (3.2)–(3.4) and (4.2) play the same role as before. Constraints (4.21) establish a threshold mechanism using parameter τ. When $u_s = 1$, this constraint enforces $\tilde{x}_s \geq \tau$, indicating the column exceeds the threshold. When $u_s = 0$, the constraint reduces to $\tilde{x}_s \geq 0$, which is trivially satisfied.

Constraints (4.22) and (4.23) define the relationship between the continuous variables $\tilde{x}_s$, binary variables u_s, and auxiliary variables v_s. Together, they ensure that v_s can only be strictly positive when $u_s = 1$ (indicating the column exceeds the threshold), and that v_s cannot exceed the actual value of $\tilde{x}_s$.

The objective function (4.20) maximizes the sum of these auxiliary variables, effectively pushing up the values of all columns that exceed the threshold. Unlike the first MIP formulation that focuses on maximizing a single column's value, or the second formulation that pushes all columns toward integrality, this approach identifies a set of promising columns and collectively maximizes their values.

This approach offers a balanced perspective between maximizing individual column values and considering the collective structure of the solution. By focusing on columns above a threshold, it identifies a set of promising candidates for column fixing while maximizing their values to differentiate between them. This differentiation is particularly valuable when we fix fewer columns than the total number of columns above the threshold, as it helps identify those most likely to contribute to high quality integer solutions.

CHAPTER 5 EXPERIMENTAL RESULTS

This chapter presents our experimental results for the MDEVSP. We begin by introducing the benchmark instances and evaluation methodology, establishing context for our findings. We then present baseline performance using standard variable fixing techniques and analyze the distribution of fractional values in master problem solutions, highlighting challenges in column selection. We proceed by evaluating our proposed approaches with various deterioration factors: probing-based techniques for finding alternative fractional solutions and MIP formulations. Each method is assessed across different instance sizes, examining their impact on solution quality and computational efficiency. We conclude with a comparative analysis of these approaches and validate our findings on an expanded dataset, demonstrating how exploring alternative fractional solutions can improve integer solution quality for the MDEVSP.

5.1 Instances and benchmarks

To evaluate the effectiveness of our alternative fractional solution methods, we conducted computational experiments on three distinct types of instances of the MDEVSP. These instances vary in complexity, network structure, and problem size, providing a basis for assessing the performance of our approaches. These instances were originally generated by Gerbaux [26] for her master's thesis, using the random MDEVSP instance generator of Brasseur [27] that exploits real operational data from the Montreal transit network. This ensures that our test instances, while synthetic, accurately reflect the complexities and constraints of practical urban transit operations.

5.1.1 Instance characteristics

The three types of instances used in our experiments are:

- **Small single-depot instances (SD-600):** These instances contain a single depot and between 609 and 862 tasks to be scheduled. While simpler in depot configuration, they represent realistic medium-sized urban transit operations.
- **Medium dual-depot instances (DD-800):** These instances feature two depots and between 736 and 816 tasks. The dual-depot structure introduces additional complexity by requiring coordination between multiple vehicle origins and destinations.

- **Large dual-depot instances (DD-1300):** These instances also have two depots but with significantly larger task sets ranging from 1274 to 1424 tasks. They represent large urban transit networks with high operational complexity.

Ten instances were generated for each type, providing a preliminary benchmark to evaluate the efficacy of our solution techniques across multiple problem configurations.

5.1.2 Performance baseline

To establish a performance baseline, we first solved all instances using four standard variable fixing techniques (see Section 3.4) that yield four diving heuristics:

- Column fixing alone (CFIX)
- Combined column fixing and inter-task fixing (CFIX+ITFIX)
- Retrospective branching (RBFIX)
- Combined retrospective branching and inter-task fixing (RBFIX+ITFIX)

Table 5.1 presents a comparison of optimality gaps achieved by four distinct diving heuristics across three instance categories of varying complexity. The results reveal significant performance variations both within and across instance types.

Table 5.1 Optimality gaps (%) for standard diving heuristics

	SD-600			DD-800			DD-1300		
Method	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
CFIX	1.43	0.14	2.74	2.91	1.06	4.34	2.94	1.88	3.90
CFIX+ITFIX	1.05	0.25	3.07	2.04	1.02	5.75	3.07	1.72	4.26
RBFIX	1.33	0.17	2.70	1.91	0.86	3.10	2.62	1.66	4.01
RBFIX+ITFIX	1.01	0.21	3.47	1.85	0.84	3.54	2.13	1.34	3.02

For clarity, we define the *optimality gap* as the relative difference between the integer solution cost and the optimal value of the master problem:

$$\text{Optimality gap} = \frac{\text{Integer solution cost} - \text{Linear relaxation's optimal value}}{\text{Linear relaxation's optimal value}}$$

Since the exact integer optimum is not available, this metric reflects both the quality of the current integer solution and the inherent gap between the linear and integer formulations. It therefore captures both the integrality gap (the difference between the optimal values of the integer and linear relaxation problems) and the optimality gap (the difference between the current integer solution and the unknown integer optimum).

This definition provides a consistent and practical measure of solution quality across different instance sizes and configurations.

For the small single-depot instances (SD-600), all methods achieve relatively tight optimality gaps averaging between 1.01% and 1.43%. Notably, the combined approaches of CFIX+ITFIX and RBFIX+ITFIX outperform their standalone counterparts, achieving average gaps of 1.05% and 1.01% respectively. This suggests that the structural insights captured by inter-task fixing contribute meaningfully to solution quality in these instances.

As we move to medium dual-depot instances (DD-800), the optimality gaps increase across all methods, with averages ranging from 1.85% to 2.91%. This indicates that the introduction of multiple depots significantly increases problem complexity. CFIX alone performs notably worse with an average gap of 2.91%, while the combined approaches incorporating retrospective branching deliver the best results (1.91% for RBFIX and 1.85% for RBFIX+ITFIX). The performance differential between the simplest approach (CFIX) and the most sophisticated one (RBFIX+ITFIX) grows substantially in these instances, showing the potential value of advanced heuristic techniques for more complex problems.

For large dual-depot instances (DD-1300), the optimality gaps slightly increase compared to those of the medium instances, even with the significant increase in problem size. RBFIX+ITFIX continues to be the most effective approach with an average gap of 2.13%, while CFIX+ITFIX surprisingly performs worse than CFIX alone (3.07% vs. 2.94%). This counterintuitive result suggests that inter-task fixing may sometimes lead to poor early decisions that propagate through the solution process in larger instances, without the correction mechanism that retrospective branching provides.

The variability within each instance category, as indicated by the minimum and maximum gaps, reveals that performance is highly instance-dependent. This inconsistency demonstrates the challenge of developing universally effective heuristics for the MDEVSP and motivates our exploration of alternative fractional solutions.

Critical insights into the computational efficiency trade-offs between the standard column fixing approach and the more complex retrospective branching method is provided by Table 5.2. The table quantifies these differences through solution time ratios across the three instance categories.

For small instances (SD-600), RBFIX requires on average 3.39 times more computational

Table 5.2 Computational performance comparison between CFIX and RBFIX

	SD-600			DD-800			DD-1300		
Solution time ratio	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
RBFIX/CFIX	3.39	2.07	5.63	4.53	1.88	8.44	10.94	8.30	14.36
RBFIX+ITFIX/CFIX+ITFIX	1.18	0.70	1.70	1.84	1.56	2.22	1.79	1.57	2.24

effort than CFIX, with the ratio ranging from 2.07 to 5.63 across different instances. This substantial increase in computational time is somewhat mitigated when inter-task fixing is incorporatated, with RBFIX+ITFIX requiring only 1.18 times more effort than CFIX+ITFIX on average. This suggests that the added computational time of retrospective branching is partially absorbed by the use of inter-task fixing, as it guides the column generation algorithm to a specific region of the solution space.

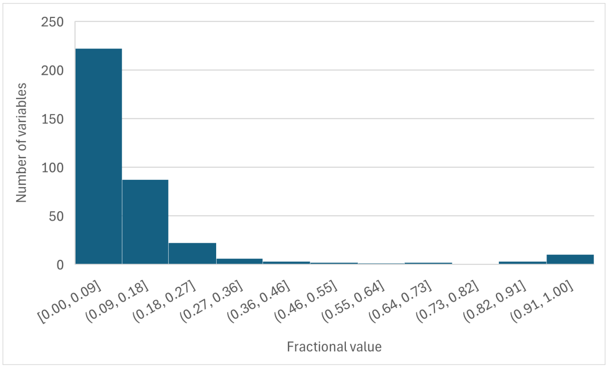
The computational disadvantage of retrospective branching becomes more pronounced in medium instances (DD-800), where RBFIX demands 4.53 times more computational resources than CFIX on average. For the combined approaches, the ratio remains more manageable at 1.84, indicating that the computational impact of retrospective branching remains consistently lower when paired with inter-task fixing.

For large instances (DD-1300), the computational disparity reaches its peak, with RBFIX requiring nearly 11 times more computational effort than CFIX on average. Interestingly, the ratio for the combined approaches remains similar to that of medium instances at 1.79, suggesting that the computational scaling behavior of retrospective branching with inter-task fixing is more favorable than that of retrospective branching alone.

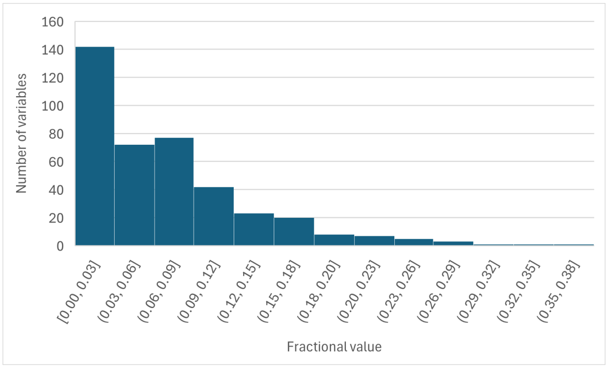
These findings reveal a critical trade-off between solution quality and computational efficiency. While retrospective branching consistently produces better solutions (as shown in Table 5.1), this improvement comes at a substantial computational cost that grows disproportionately with problem size when used alone. The more modest computational overhead observed when combining retrospective branching with inter-task fixing suggests this integrated approach may offer the most balanced performance profile, especially for larger problem instances.

5.1.3 Distribution of fractional values

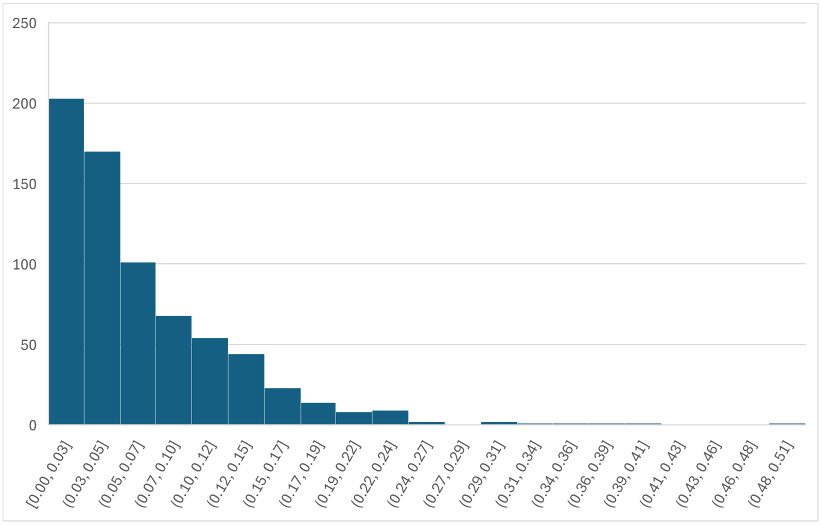
The effectiveness of diving heuristics is closely tied to the distribution of fractional values in the solution of the master problem. Figure 5.1 illustrates this distribution of these values for one representative instance from each type, showcasing the typical patterns observed across all instances at the root node.



(a) Small instance (SD-600)



(b) Medium instance (DD-800)

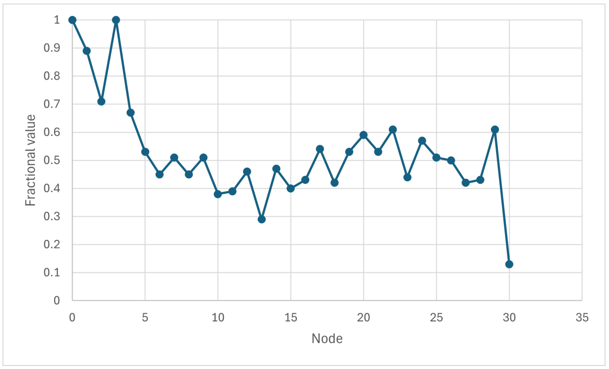


(c) Large instance (DD-1300)

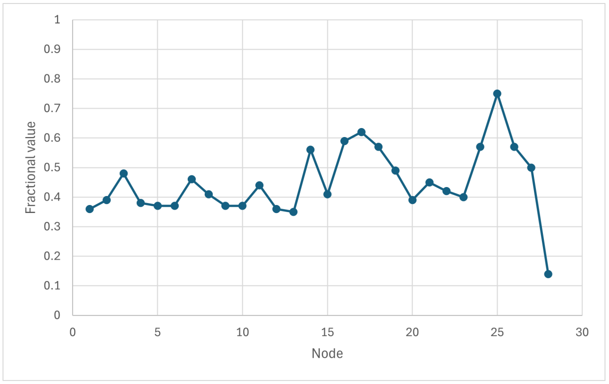
Figure 5.1 Evolution of largest fractional values through diving iterations for different instance types

A key observation from these distributions is the striking similarity across all instance types, despite their varying complexities and sizes. As shown in Figure 5.1, the vast majority of variables have near-zero fractional values at the root node, with only a small number of columns having slightly larger values. The consistent pattern across all instance types suggests that this is an inherent property of the MDEVSP's solution structure rather than an artifact of specific problem configurations, further motivating the need for more sophisticated methods to identify promising columns for fixing and differentiating columns that have larger values.

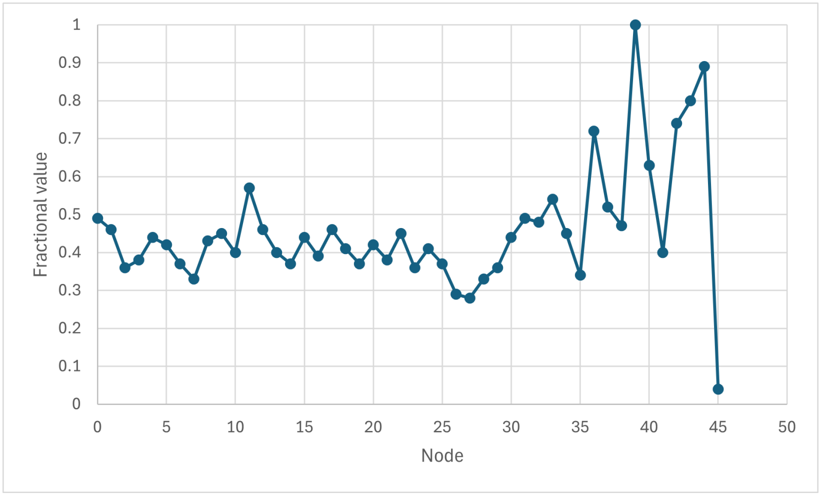
Figure 5.2 shows how these fractional values evolve throughout the solution process for one representative instance from each category, tracking the largest fractional value at each node when the decision to fix variables to one is made.



(a) Small instance (SD-600)



(b) Medium instance (DD-800)



(c) Large instance (DD-1300)

Figure 5.2 Evolution of largest fractional values through diving iterations for different instance types

The evolution patterns displayed in Figure 5.2 reveal a concerning trend: the largest fractional values at each decision point throughout the solution process typically hover around 0.5. This is problematic since these variables are precisely the ones selected to be fixed at one in the diving heuristic. Such modest fractional values indicate that the solution provides relatively weak guidance for integer fixing decisions, as no variable stands out as an obvious candidate. This observation shows the potential value of exploring alternative fractional solutions, as different decisions might lead to significantly different trajectories through the solution space and potentially stronger fixing candidates with values closer to one.

These baseline findings highlight the need for more sophisticated approaches to identify promising column fixing candidates, motivating our exploration of alternative fractional solutions as described in Chapter 4. The significant variation in performance across instances and methods suggests that no single approach consistently outperforms others, indicating potential benefits from combining different strategies or adaptively selecting methods based on instance characteristics.

An important question is whether larger fractional values at the root node reliably lead to better integer solutions when using column fixing heuristics. Figure 5.3 presents the relationship between the largest fractional value at the root node and the resulting optimality gap when using the CFIX heuristic across various instances.

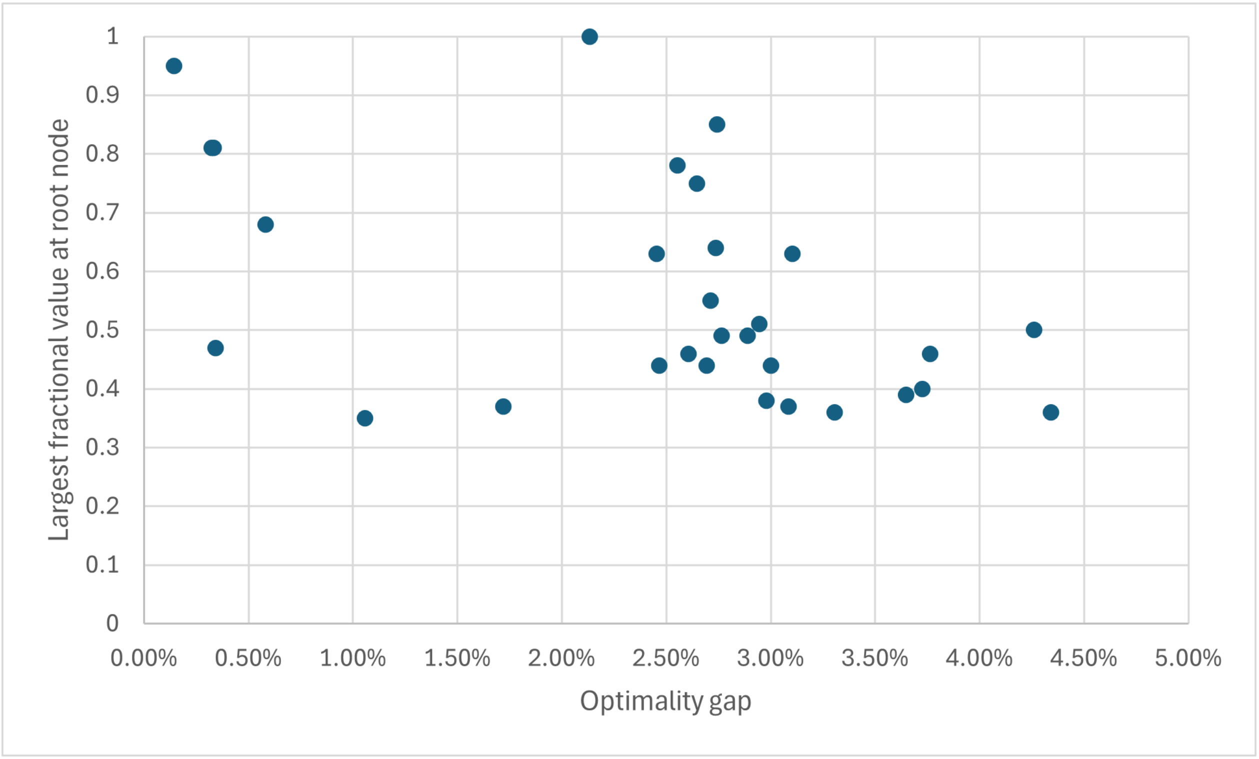


Figure 5.3 Relationship between largest fractional value at root node and optimality gap

As Figure 5.3 clearly demonstrates, there is no strong correlation between the largest fractional value at the root node and the quality of the final integer solution. Some instances with relatively low fractional values (around 0.35–0.5) achieve excellent optimality gaps close to 0%, while other instances with larger fractional values (above 0.8) result in significantly larger optimality gaps. This surprising finding contradicts the intuitive assumption that variables with values closer to 1 would definitely provide better fixing decisions. However, the final integer solution is the result of a sequence of fixings so we do not know if the figure fully reflects the impact of the choice made at the root node.

5.1.4 Comparison methodology

Throughout the remainder of this chapter, we evaluate our proposed methods using two complementary approaches:

Relative improvement analysis: For each proposed method, we compute the average, minimum, and maximum improvement compared to the appropriate baseline (CFIX alone, or CFIX+ITFIX if the method incorporates inter-task fixing). This improvement is measured as the reduction in the optimality gap:

$$\text{Improvement} = \text{Gap}_{\text{baseline}} - \text{Gap}_{\text{proposed}}$$

where $\text{Gap}_{\text{proposed}}$ represents the optimality gap achieved using the technique to find alternative solutions, and $\text{Gap}_{\text{baseline}}$ is the optimality gap obtained from the existing diving heuristic approach.

Best solution analysis: Since the computational requirements of our methods are significant but allow for parallel execution, we also evaluate scenarios where multiple approaches are run concurrently. For each instance type, we report the average, minimum, and maximum optimality gap when always selecting the best solution between the proposed method and the benchmark approach. This analysis quantifies the potential benefit of running multiple heuristics in parallel.

This dual evaluation framework provides complementary perspectives: the first measures direct performance improvements from individual methods, while the second assesses the potential gains from employing multiple approaches in parallel.

5.2 Probing results

To assess the effectiveness of probing-based methods for finding alternative fractional solutions, we conducted experiments using the various probing techniques outlined in Section 4.1. For each method, we applied specific deterioration factors and compared the results against our standard CFIX baseline.

We evaluated the following probing strategies:

- MFV (Maximize Fractional Value): The primary probing approach that maximizes the value of each candidate column.
- MAC (Maximize Average of Candidates): Secondary objective that maximizes the average value of other candidate columns.
- MRI (Maximize Relative Improvement): Secondary objective that prioritizes columns with the largest relative improvement from their original values.
- MOV (Minimize Objective with Variable Fixed): Secondary objective that directly simulates fixing by minimizing the objective value with the candidate column fixed to one.

Table 5.3 presents the improvements in the relative optimality gap achieved by each probing method compared to the standard CFIX baseline across the three instance types.

Table 5.3 Improvements in the relative optimality gap (%) over CFIX for probing-based strategies

Method	**Deterioration**	SD-600			DD-800			DD-1300		
	Factor	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
MVF	0.0001	0.29	-1.89	2.46	0.85	-1.18	2.16	1.14	-0.25	2.00
	0.00005	-0.71	-2.23	2.46	0.38	-1.83	2.17	0.18	-0.86	1.36
MAC	0.0001	-0.11	-2.46	2.33	0.60	-1.69	2.11	0.22	-1.52	1.75
	0.00005	-0.73	-2.56	0.50	0.15	-1.80	1.91	0.22	-1.16	1.67
MRI	0.0001	-0.38	-2.28	2.00	-0.07	-2.28	2.27	1.01	-1.21	2.36
	0.00005	-0.84	-2.71	0.37	0.17	-2.32	2.15	0.32	-2.30	1.77
MOV	0.0001	-0.26	-3.49	1.92	0.11	-3.78	2.21	0.39	-1.14	1.56
	0.00005	-0.87	-3.68	0.34	-0.06	-2.35	1.79	0.53	-1.58	3.12

The MFV approach with the larger deterioration factor (0.0001) demonstrates the most consistent performance across all instance types. For small instances (SD-600), it achieves a small average improvement of 0.29%, while for medium instances (DD-800), this increases to 0.85%. Most notably, for large instances (DD-1300), MFV achieves the largest average improvement of 1.14% among all tested approaches. This suggests that the method becomes more effective as problem complexity increases.

When the deterioration factor is reduced to 0.00005 for MFV, performance degrades significantly for small instances (-0.71% average), while medium and large instances still show positive but diminished improvements (0.38% and 0.18%, respectively). This indicates that the more aggressive deterioration factor is preferable for the MFV approach across all instance types.

All three secondary objectives for equivalent columns show mixed results. For example, MAC has a slight average degradation for small instances (-0.11%), but positive improvements for medium and large instances (0.60% and 0.22%). With the reduced deterioration factor of 0.00005, performance worsens for small and medium instances (-0.73% and 0.15% respectively), while remaining stable for large instances (0.22%).

For the MRI approach, the larger deterioration factor yields negative average improvement for small instances (-0.38%) and medium instances (-0.07%), but performs well on large instances with an average improvement of 1.01%. When using the smaller deterioration factor of 0.00005, performance deteriorates further for small instances (-0.84%), slightly improves for medium instances (0.17%), but shows a considerable reduction in effectiveness for large instances (0.32% compared to 1.01%).

The MOV approach with deterioration factor 0.0001 shows negative average improvement for small instances (-0.26%), marginal improvement for medium instances (0.11%), and moderate improvement for large instances (0.39%). However, with the smaller deterioration factor, performance worsens for small instances (-0.87%) and medium instances (-0.06%), though interestingly shows a slight improvement for large instances (0.53%).

Across most methods, the high variability between minimum and maximum improvements is notable. For example, MAC with a deterioration factor of 0.0001 ranges from a -2.46% degradation to a 2.33% improvement on SD-600. However the MVF method shows the less variability with the best minimum values of the different methods across all three instance types.

Regarding the second comparison methodology, Table 5.4 presents the optimality gaps achieved when selecting the best solution between each probing method and the CFIX baseline. This best solution analysis reveals more promising results, particularly for small and medium instances. For small instances, MAC achieves the lowest average optimality gap (0.73%), representing a substantial improvement over the 1.43% gap of the CFIX baseline reported in Table 5.1. This suggests that maximizing the average value of other candidate columns is particularly effective for smaller problem instances when combined with standard column fixing.

For medium instances, the MFV method with the larger deterioration factor yields the best performance, achieving an average gap of 2.01%, which is significantly better than the 2.91% baseline. This reinforces the earlier observation that this approach works especially well for

Table 5.4 Optimality gaps (%) for CFIX alone and when selecting best solution between probing method and CFIX

Method	**Deterioration Factor**	SD-600			DD-800			DD-1300		
		Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
CFIX		1.43	0.14	2.74	2.91	1.06	4.34	2.94	1.88	3.90
MVF	0.0001	0.96	0.14	2.74	2.01	0.91	3.31	2.24	1.30	3.76
	0.00005	1.13	0.14	2.55	2.26	0.79	4.34	2.69	1.72	3.90
MAC	0.0001	0.73	0.14	2.38	2.04	0.99	3.31	2.57	1.33	3.59
	0.00005	1.28	0.14	2.74	2.50	0.82	4.34	2.64	1.66	3.73
MRI	0.0001	1.18	0.14	2.65	2.48	0.83	4.34	1.94	1.40	3.65
	0.00005	1.34	0.14	2.74	2.27	0.82	4.34	2.48	1.40	3.90
MOV	0.0001	1.20	0.14	2.74	2.32	0.90	4.34	2.55	1.18	3.76
	0.00005	1.36	0.14	2.74	2.57	0.81	3.31	2.53	1.14	3.76

medium sized problems.

In large instances, MRI stands out, achieving an average optimality gap of 1.94% compared to the CFIX baseline of 2.94%. This 1% absolute improvement shows what is achievable with large instances but MRI does not perform particularly well with the two smaller instance types.

When examining the impact of deterioration factors, the larger factor (0.0001) outperforms the more restrictive factor (0.00005) across most methods. For MFV, the average gap increases from 0.96% to 1.13% for small instances, from 2.01% to 2.26% for medium instances, and from 2.24% to 2.69% for large instances when switching to the smaller deterioration factor. This pattern is similarly observed for MAC and MRI approaches, suggesting that allowing greater flexibility around the optimum facilitates the identification of better columns to fix.

Particularly noteworthy is the performance degradation with the smaller deterioration factor for MRI on large instances, where the average gap increases from 1.94% to 2.48%. This significant difference of 0.54 percentage points highlights the sensitivity of this method to the deterioration factor parameter on complex problem instances.

For the MOV approach, the gap difference between deterioration factors is less pronounced for large instances (2.55% vs 2.53%), which aligns with the earlier observation that MOV with the lower deterioration factor performed slightly better on larger problems.

Despite these positive results, it is important to note that all probing methods exhibit significant variability in performance, with wide ranges between minimum and maximum optimality gaps. This inconsistency highlights the challenge of developing universally robust strategies for exploring alternative fractional solutions in the MDEVSP.

Overall, the probing results confirm that exploring alternative fractional solutions can lead to improved integer solutions, particularly when used alongside standard column fixing approaches. The computational efficiency of these techniques, relying on solving LPs rather than full mixed-integer models, makes them especially attractive in practical settings when balancing solution quality and runtime is important. The results also emphasize the importance of carefully selecting an appropriate deterioration factor, with the larger value of 0.0001 generally producing superior results across most methods and instance types.

5.3 MIPs results

To evaluate the effectiveness of our MIP approaches for finding alternative fractional solutions, we conducted experiments using all three MIP formulations introduced in Section 4.2 across our test instances. For each formulation, we tested two different deterioration factors (0.00001 and 0.00005) to assess the impact of allowing different degrees of solution quality deterioration on the final integer solutions.

We refer to the formulations using the following acronyms:

- **MF**: Max Fraction — maximizes the largest fractional variable
- **MSD**: Min Sum Difference — minimizes the sum of differences between fractional values and their nearest bound
- **MAT**: Max Above Threshold — maximizes the total value of fractional variables above a fixed threshold (set to 0.7)

5.3.1 Comparison of MIP formulations with different deterioration factors

Table 5.5 presents the improvements in the relative optimality gap achieved by each MIP formulation compared to the standard CFIX baseline across different instance types.

When looking at maximizing the fractional value (MF), we observe mixed results across instance types. For small instances (SD-600), this method consistently under-performs the baseline with negative average improvements, particularly with the larger deterioration factor (−0.98%). However, for medium instances (DD-800), MF shows positive improvements, with the smaller deterioration factor yielding better results (0.87% improvement). Large instances

Table 5.5 Improvements in the relative optimality gap (%) over CFIX baseline for MIP formulations

Method	Deterioration Factor	SD-600			DD-800			DD-1300		
		Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
MF	0.00005	-0.98	-3.74	0.37	0.39	-1.16	2.36	-0.11	-2.65	1.57
	0.00001	-0.13	-2.20	2.15	0.87	-1.65	2.39	0.60	-1.32	1.58
MSD	0.00005	-0.30	-2.09	0.42	0.24	-2.03	2.47	0.18	-1.13	1.70
	0.00001	-0.30	-2.15	2.21	-0.50	-1.95	0.33	0.25	-1.78	1.54
MAT	0.00005	-0.74	-3.79	2.49	0.46	-0.64	2.27	0.56	-1.17	2.43
	0.00001	0.34	-1.49	2.22	0.37	-0.45	2.28	0.49	-1.02	1.75

(DD-1300) show a similar pattern, with the smaller deterioration factor producing better average improvements (0.60%) compared to a slight degradation (−0.11%) with the larger deterioration factor.

When minimizing the sum of differences to the bounds (MSD), this approach shows generally modest and inconsistent performance improvements. For small instances, it produces average degradations regardless of deterioration factor (−0.30%). Medium instances exhibit mixed results, with a slight improvement (0.24%) using the larger deterioration factor but deterioration (−0.50%) with the smaller factor. For large instances, MSD achieves small positive improvements (0.18% and 0.25%), with the smaller deterioration factor performing marginally better.

Lastly, maximizing the values of variables above the threshold emerges as the most promising MIP formulation, particularly with the smaller deterioration factor. It is the only method to achieve positive average improvements across all instance types with the 0.00001 deterioration factor (0.34%, 0.37%, and 0.49% for small, medium, and large instances respectively). With the larger deterioration factor, performance is mixed, with a notable degradation for small instances (−0.74%) but positive improvements for medium and large instances (0.46% and 0.56%).

Across all methods, the high variability between minimum and maximum improvements is striking. For example, the MAT approach with a 0.00005 deterioration factor ranges from a 3.79% degradation to a 2.49% improvement on small instances. This substantial variability indicates that the effectiveness of these methods is highly instance-dependent, with no approach consistently outperforming others across all problem instances.

When looking at the other comparison methodology, Table 5.6 presents the optimality gaps achieved when selecting the best solution between each MIP formulation and the CFIX baseline. This simulates a strategy where multiple heuristics are run and the best final result is chosen.

Table 5.6 Optimality gaps (%) for CFIX alone and when selecting best solution between each MIP method and CFIX

Method	Deterioration	SD-600			DD-800			DD-1300		
	Factor	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
CFIX		1.43	0.14	2.74	2.91	1.06	4.34	2.94	1.88	3.90
MF	0.00005	1.36	0.14	2.74	2.31	0.75	4.34	2.78	1.72	3.76
	0.00001	0.94	0.14	2.74	1.78	0.71	2.98	2.24	1.16	3.65
MSD	0.00005	1.36	0.14	2.65	2.24	0.64	4.34	2.51	1.29	3.73
	0.00001	1.12	0.14	2.74	2.88	1.06	4.34	2.51	1.22	3.73
MAT	0.00005	0.98	0.14	2.74	2.30	1.06	3.31	2.28	1.33	3.58
	0.00001	0.91	0.14	2.50	2.45	1.06	3.30	2.44	1.14	3.76

This analysis reveals more promising results, demonstrating again the potential benefits of running multiple approaches in parallel. For small instances (SD-600), the MAT formulation with either deterioration factor achieves the lowest average optimality gaps (0.98% and 0.91%), substantially better than the 1.43% gap of the CFIX baseline reported in Table 5.1. This indicates that the MAT approach, when used in conjunction with CFIX, can identify higher-quality integer solutions for some of these instances.

For medium instances (DD-800), the MF formulation with a deterioration factor of 0.00001 achieves the best performance with an average optimality gap of 1.78%, which represents a significant improvement over the 2.91% gap of the CFIX baseline. This strong performance aligns with the positive improvement observed for this configuration in Table 5.5.

The large instances (DD-1300) show similar improvements, with the MAT formulation (deterioration factor 0.00005) and the MF formulation (deterioration factor 0.00001) achieving average optimality gaps of 2.28% and 2.24%, respectively, compared to the 2.94% gap of the CFIX baseline.

The deterioration factor continues to play an important role in this analysis. For MF, the smaller deterioration factor consistently produces better results across all instance types. For MSD and MAT, the relationship is more complex and instance-dependent. This suggests that tuning the deterioration factor based on instance characteristics could further improve performance.

Despite the encouraging results from the best solution analysis, a more critical examination reveals important limitations of the MIP approaches. The high variability observed in

Table 5.5 (with improvements ranging from substantial degradations to meaningful gains) indicates that these methods lack consistency and reliability. The close-to-zero average improvements across many configurations suggest that while these methods occasionally find better solutions, they also frequently perform worse than the baseline.

It is also important to address their computational requirements. Throughout our experiments, we enforced a five minute time limit for each mixed-integer program. This limit was frequently reached during the early nodes of the solution process, though solving time decreased significantly in later nodes as the solution space became more constrained. All three MIP approaches exhibited similar computational profiles regardless of the deterioration factor used. Unlike probing techniques, which can be executed in parallel due to their independence, the MIP-based techniques create a dependence between all variables. This dependency makes the MIP approach inherently more computationally demanding. This computational overhead should be carefully weighed against the modest and sometimes inconsistent improvement in solution quality when considering practical implementations.

This inconsistency is problematic from a practical implementation perspective. In real-world applications, the reliability and predictability of solution quality are often as important as achieving the best possible solution in some cases. The wide performance variations observed with the MIP approaches make them difficult to recommend as standalone replacements for established methods like CFIX.

5.3.2 Integration with inter-task fixing

Building upon our exploration of MIP approaches in combination with standard column fixing, we now examine their integration with inter-task fixing. This strategy takes into account the structural insights offered by inter-task relationships while still exploring alternative fractional solutions through MIP-based formulations. Since inter-task fixing is prioritized over column fixing in this approach, fewer columns are fixed during the process, resulting in less frequent invocation of the MIP formulations. As before, we evaluate all three MIP formulations under two deterioration factors (0.00001 and 0.00005), across all three instance categories.

Table 5.7 presents the improvements in the relative optimality gap over the CFIX+ITFIX baseline. Compared to column fixing alone, the addition of ITFIX shifts the overall performances.

Among the three methods, MF and MSD tend to perform more reliably when applied to the large instances. In these cases, both deterioration factors result in consistent gains. However,

Table 5.7 Improvements in the relative optimality gap (%) over CFIX+ITFIX baseline for MIP formulations with ITFIX

Method	Deterioration Factor	SD-600			DD-800			DD-1300		
		Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
MF	0.00005	-0.34	-1.95	1.90	0.29	-1.64	2.52	0.66	-0.66	2.01
	0.00001	0.26	-2.41	2.26	-0.19	-2.25	4.15	0.40	-0.78	1.79
MSD	0.00005	-0.93	-2.42	2.10	-0.22	-1.97	2.27	0.70	-0.31	1.70
	0.00001	-0.97	-5.15	1.91	0.20	-1.95	2.86	0.46	-0.96	1.92
MAT	0.00005	0.19	-1.94	2.57	-0.34	-2.08	2.65	0.38	-2.22	1.32
	0.00001	-0.40	-1.63	2.59	-0.26	-1.73	1.99	0.54	-0.95	1.90

on smaller and medium-sized instances, all MIP performances are mixed, which often leads to minor degradation or inconsistent behavior.

MSD, which encourages integrality by minimizing the sum of differences to 0 or 1, is the most sensitive to the presence of ITFIX. On smaller instances, it consistently underperforms. In contrast, for large instances, MSD tends to generate improvements, especially when using the larger deterioration factor. This hints at a positive interaction between integrality-pushing and the complex feasibility landscape of larger problems.

MAT delivers the most erratic performance across the board. While some individual instances experience significant improvement, others suffer noticeable degradation. On small instances, MAT shows an improvement only when using the largest deterioration factor, while on medium instances, both factors result in slight declines. On large instances, however, the method again proves its value as both factors yield improvements, with the smaller one being slightly more robust.

Overall, Table 5.7 highlights a key insight: while integrating MIP approaches with ITFIX can yield gains, the combined method is less predictable and more sensitive to tuning than with CFIX alone. This makes it especially important to evaluate not just average performance but best-case results across diverse settings.

Table 5.8 addresses this by showing the optimality gaps obtained when choosing the best solution between each MIP method and the CFIX+ITFIX baseline. Here, the improvements are more pronounced and consistent, particularly for small and large instances.

The best solution analysis has a more optimistic point of view. For small instances, all methods, especially MAT with a larger deterioration factor, produce significantly lower gaps than the CFIX+ITFIX baseline (0.32% vs. 1.05%), despite the variability noted in the direct im-

Table 5.8 Optimality gaps (%) for CFIX+ITFIX alone and when selecting best solution between each MIP technique with ITFIX and CFIX+ITFIX

Method	Deterioration	SD-600			DD-800			DD-1300		
	Factor	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
CFIX+ITFIX		1.05	0.25	3.07	2.04	1.02	5.75	3.07	1.72	4.26
MF	0.00005	0.57	0.23	2.31	1.51	1.00	3.55	2.26	1.31	2.90
	0.00001	0.35	0.18	0.81	1.58	1.02	3.59	2.43	1.31	3.84
MSD	0.00005	0.61	0.25	3.07	1.73	1.02	3.48	2.35	1.46	3.00
	0.00001	0.62	0.25	3.07	1.55	1.02	3.83	2.30	1.72	2.93
MAT	0.00005	0.32	0.18	0.56	1.57	0.82	4.01	2.30	1.40	2.90
	0.00001	0.58	0.25	2.58	1.67	0.86	3.76	2.32	1.21	3.90

provement analysis. Medium instances also benefit from these methods, with MF performing best (1.51% average gap), indicating strong synergy between maximizing key variables and inter-task constraints in moderately sized networks.

Large instances continue to show the greatest advantage. All MIP methods produce average gaps near or below 2.4%, clearly outperforming the 3.07% baseline. Once again, MAT with the large deterioration factor provides the best result (2.30%), supporting the idea that allowing more flexibility in cost deterioration can unlock better structural decisions in large, complex schedules.

Interestingly, the deterioration factor's impact is less straightforward under ITFIX. For MF, the smallest factor performs better for the small instances but not the large ones. For MSD and MAT, performance is less consistent and depends heavily on the instance context. This suggests that adaptive tuning of the deterioration factor, based on instance characteristics, could be valuable.

In summary, integrating MIP-based alternative fractional solution strategies with inter-task fixing offers measurable benefits, particularly in large-scale problems where richer structural patterns can be exploited. However, the sensitivity to instance characteristics and the presence of degradation in certain cases also caution against relying on these methods on their own.

5.4 Overall analysis, enlarging the dataset

After evaluating both probing techniques and MIP formulations for finding alternative fractional solutions, our results indicate that probing-based approaches offer the most promising balance of performance improvement and computational efficiency.

The probing methods demonstrate more consistent performance across all instance types, particularly for medium and large instances. The MFV approach with a deterioration factor of 0.0001 achieved average improvements of 0.29%, 0.85%, and 1.14% for small, medium, and large instances respectively. This consistent upward trend as instance size increases suggests that probing becomes increasingly effective for more complex problems.

In contrast, MIP-based approaches showed greater variability in performance. While some MIP formulations occasionally produced superior individual results, they also frequently led to significant deteriorations, making them less reliable in practice. The MAT formulation with the smallest deterioration factor showed the most consistent results among MIP approaches, but still underperformed compared to probing methods.

The computational efficiency of probing techniques further distinguishes them from MIP formulations. By relying on linear programming rather than solving mixed-integer programs, probing methods can explore alternative solutions with significantly lower computational overhead. This efficiency is particularly valuable in practical applications where solution time is a critical consideration.

To verify that our findings were not limited to the initial test set of 30 instances, we conducted an extensive validation study using 1,000 small instances, 500 medium instances, and 200 large instances. The best performing probing approach (MFV with a deterioration factor of 0.0001) was applied across this expanded dataset.

Table 5.9 shows the improvements in the relative optimality gap achieved by the MFV approach over the standard CFIX baseline. For small instances, we observed an average improvement of 0.09%, with improvements ranging from -4.16% to 4.10%. Medium instances showed similar close to zero average improvements of 0.08%, with a similar range of improvements, where large instances demonstrated slightly stronger performance with an average improvement of 0.18%, ranging from -2.39% to 2.91%.

Table 5.9 Improvements in the relative optimality gap (%) over CFIX for MFV (0.0001) on additional evaluation

Method	Deterioration	SD-600			DD-800			DD-1300		
	Factor	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
MFV	0.0001	0.09	-4.16	4.10	0.08	-3.79	4.12	0.18	-2.39	2.91

These results reveal that while the average improvements are modest, the method can produce substantial improvements in some cases, up to 4.12% for medium instances. The significant range between minimum and maximum values also highlights that the effectiveness of the probing approach varies considerably across different problem instances, with some instances benefiting notably while others show deterioration.

To assess the statistical significance of these improvements across the expanded dataset, we conducted paired t-tests comparing the performance of CFIX alone versus the MFV approach for each instance type. The resulting p-values were 0.664 for SD-600, 0.237 for DD-800, and 0.017 for DD-1300 instances. These p-values represent the probability of observing the measured difference (or a more extreme difference) between the two approaches if there were actually no difference in their true performance (null hypothesis). The results indicate that the improvement for DD-1300 instances is statistically significant at the conventional $\alpha = 0.05$ level, while the improvements for SD-600 and DD-800 instances do not reach statistical significance at this threshold. This statistical analysis reinforces our earlier observation that probing methods become increasingly effective as instance size grows, with the most reliable performance gains achieved on larger, more complex instances.

Table 5.10 presents the optimality gaps of the CFIX baseline and when selecting the best solution between the MFV approach and CFIX for each instance. The benefits of the combined approach become evident when compared to the baseline.

Table 5.10 Optimality gaps (%): CFIX baseline and best of MFV and CFIX

Method	SD-600			DD-800			DD-1300		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
CFIX only	1.44	0.07	6.64	2.78	0.72	6.49	2.77	0.88	4.46
Best of MFV & CFIX	0.87	0.07	4.07	2.21	0.70	5.04	2.31	0.88	3.94

For small instances, the average optimality gap decreased substantially from 1.44% to 0.87%, representing a relative improvement of nearly 40%. More remarkably, the maximum gap was reduced from 6.64% to 4.07%, demonstrating that probing can address some of the most challenging instances.

For medium instances, the average gap improved from 2.78% to 2.21%, while the maximum gap decreased from 6.49% to 5.04%. Similarly, for large instances, the average gap improved from 2.77% to 2.31%, and the maximum gap from 4.46% to 3.94%. These improvements are particularly notable given that they were achieved without substantial additional computational effort.

These results confirm our hypothesis that probing based approaches for finding alternative fractional solutions can complement traditional column fixing strategies. While the probing method does not consistently outperform CFIX across all instances, as evidenced by the near-zero average improvements in Table 5.9, it explores different regions of the solution space, enabling it to identify better quality solutions in cases where CFIX struggles.

The narrower range between minimum and maximum gaps in the best-solution analysis (Table 5.10) also suggests that the combined approach offers greater reliability than either method alone. This increased robustness is particularly valuable in practical applications where consistent performance across a diverse set of problem instances is essential.

CHAPTER 6 CONCLUSION

6.1 Summary of works

This thesis has explored methodologies for finding alternative fractional solutions to the master problem in the MDEVSP, with the goal of improving the quality of integer solutions obtained through column fixing heuristics.

The research began by identifying a key challenge in exploring alternative optimal solutions: the perturbation used to address degeneracy in column generation significantly reduces the presence of alternative solutions with identical objective values. To overcome this limitation, we introduced a controlled deterioration factor that allows for a slight increase in the objective value, effectively expanding the search space to include near-optimal solutions with potentially more favorable properties for the fixing process.

We developed and evaluated two main approaches for finding alternative fractional solutions. The first approach, probing, evaluates individual candidate columns through linear programming to highlight specific characteristics. We implemented probing strategies all having as a first objective to maximizing the chosen column's fractional value. The first method takes a decision after the result of this first objective, the three other methodologies try differentiating equivalent columns with a second objective. These objectives are to maximize the average value of candidate columns, maximize the relative improvement of the column or to minimize the master problem's objective function when the chosen column is set to one. Among these, probing without a second objective and with a deterioration factor of 0.0001 demonstrated the most consistent performance, with average improvements of 0.29%, 0.85%, and 1.14% for small, medium, and large instances respectively compared to the standard column fixing baseline.

The second approach utilized MIP formulations to capture column interactions by optimizing over the entire set of columns simultaneously. We developed three formulations: maximizing the largest fractional variable, minimizing the sum of differences between fractional values and their nearest bound and maximizing the total value of fractional variables above a fixed threshold. While these MIP formulations occasionally produced superior individual results, they showed greater variability in performance compared to probing techniques.

To validate our findings, we conducted an extensive evaluation using 1,000 small instances, 500 medium instances, and 200 large instances. When selecting the best solution between probing without a second objective and the standard column fixing method, we observed substantial improvements in average optimality gaps: from 1.44% to 0.87% for small instances, 2.78% to 2.21% for medium instances, and 2.77% to 2.31% for large instances.

Furthermore, we investigated the integration of our methods with inter-task fixing, finding that this combination could yield additional benefits, particularly for larger instances. This integration highlights the potential complementarity between alternative solution approaches and more traditional fixing heuristic strategies.

Overall, this research demonstrated that exploring alternative fractional solutions can effectively complement traditional column fixing strategies by identifying better quality solutions in cases where standard approaches struggle. While the performance improvements are modest on average, the significant improvements observed in some instances and the reduction in maximum optimality gaps suggest that these methods can enhance the robustness and reliability of solution approaches for the MDEVSP.

6.2 Limitations

The research presented in this thesis faces several important limitations that should be acknowledged when interpreting the results. A fundamental limitation is the highly instance-dependent nature of alternative solution approaches. As evidenced by the wide variability in performance across different problem instances, no single method consistently outperforms others across all scenarios. This inconsistency, which is inherent to the sequential nature of the heuristic methods we aim to refine, makes it difficult to prescribe a universally applicable approach for finding alternative fractional solutions.

Another limitation is the lack of theoretical guarantees regarding which structural characteristics of fractional solutions are most conducive to good quality integer solutions. The research operates on hypotheses about potentially beneficial properties, such as maximizing individual column values or minimizing the distance to integrality, but cannot definitively prove which properties are inherently superior for generating quality integer solutions through column fixing.

The deterioration factor emerged as a crucial parameter with significant impact on performance, yet determining an appropriate value remains largely empirical. Without a theoretical foundation for selecting optimal deterioration values, we must rely on parameter tuning, which adds computational complexity and reduces the practical applicability of these methods.

These limitations collectively suggest that while exploring alternative fractional solutions can improve solution quality in many cases, developing robust, reliable, and computationally efficient methods remains challenging. The results show promise but highlight the need for more adaptive approaches that can better accommodate instance-specific characteristics.

6.3 Future research

Several avenues emerge for future work in this line of research. One promising direction is the integration of machine learning techniques. Future work could involve training predictive models to identify promising branching candidates based on features derived from the RMP solution or characteristics of the problem instance. Such models could reduce dependence on linear programming or MIP solutions during the search process, enabling faster decision-making in large-scale systems. Moreover, this approach may reveal which features of columns are most indicative of their usefulness at various stages of the solution, guiding the optimization model design.

Another important extension involves improving degeneracy handling. While this thesis has briefly mentioned mitigation via perturbation, further research could investigate more sophisticated strategies aimed at managing degeneracy in a way that facilitates access to a richer set of columns during the solution process, without introducing an excessive number of unnecessary variables. Dynamic or instance-specific perturbation methods may offer a path toward better trade-offs between solution quality and computational efficiency.

Finally, although the primary focus of this work is the MDEVSP, the techniques and methodologies proposed here are applicable to a broader class of set partitioning problems involving column generation under degeneracy, such as airline crew pairing. Conducting benchmarking studies across these domains would provide insights into the robustness, generality, and practical relevance of the proposed framework.

REFERENCES

[1] BloombergNEF, "Electric vehicle outlook 2024," 2024, accessed: 2025-04-03. [Online]. Available: https://about.bnef.com/electric-vehicle-outlook/

[2] Ivy Charging Network, "Modernizing public transport: Public transit and EVs," *Ivy Charging Network Blog*, February 2025, accessed: 2025-04-02. [Online]. Available: https://ivycharge.com/blog/modernizing-public-transport-public-transit-and-evs/

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York City, New York, USA: W.H. Freeman, 1979.

[4] R. Freling, A. Wagelmans, and J. Paixão, "Models and algorithms for single-depot vehicle scheduling," *Transportation Science*, vol. 35, pp. 165–180, 05 2001.

[5] S. Bunte and N. Kliewer, "An overview on vehicle scheduling models," *Public Transp*, vol. 1, no. 4, pp. 299–317, 2009. [Online]. Available: https://doi.org/10.1007/s12469-010-0018-5

[6] A. Oukil, H. B. Amor, J. Desrosiers, and H. El Gueddari, "Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems," *Computers & Operations Research*, vol. 34, no. 3, pp. 817–834, 2007, Logistics of Health Care Management. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054805001590

[7] J.-Q. Li, "Transit bus scheduling with limited energy," *Transportation Science*, vol. 48, pp. 521–539, 11 2014.

[8] O. Sassi and A. Oulamara, "Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches," *International Journal of Production Research*, vol. 55, no. 2, pp. 519–535, January 2017. [Online]. Available: https://ideas.repec.org/a/taf/tprsxx/v55y2017i2p519-535.html

[9] J. Adler and P. Mirchandani, "Online routing and battery reservations for electric vehicles with swappable batteries," *Transportation Research Part B: Methodological*, vol. 70, pp. 285–302, Dec. 2014.

[10] M. Keskin and B. Çatay, "A matheuristic method for the electric vehicle routing problem with time windows and fast chargers," *Computers & Operations Research*, vol.

100, pp. 172–188, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054818301849

[11] C. Wang, C. Guo, and X. Zuo, “Solving multi-depot electric vehicle scheduling problem by column generation and genetic algorithm,” *Applied Soft Computing*, vol. 112, p. 107774, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621006955

[12] S. S. Perumal, R. M. Lusby, and J. Larsen, “Electric bus planning & scheduling: A review of related problems and methodologies,” *European Journal of Operational Research*, vol. 301, no. 2, pp. 395–413, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221721009140

[13] G. M. Nafstad, G. Desaulniers, and M. Stålhane, “Branch-price-and-cut for the electric vehicle routing problem with heterogeneous recharging technologies and nonlinear recharging functions,” *Transportation Science*, 2025. [Online]. Available: https://doi.org/10.1287/trsc.2024.0725

[14] J. Gerbaux, G. Desaulniers, and Q. Cappart, “A machine-learning-based column generation heuristic for electric bus scheduling,” *Computers & Operations Research*, vol. 173, p. 106848, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054824003204

[15] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, “Branch-and-price: Column generation for solving huge integer programs,” *Operations Research*, vol. 46, no. 3, pp. 316–329, 1998. [Online]. Available: https://doi.org/10.1287/opre.46.3.316

[16] J. Desrosiers and M. E. Lübbecke, “A primer in column generation,” in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds. Boston, MA: Springer US, 2005, pp. 1–32. [Online]. Available: https://doi.org/10.1007/0-387-25486-2_1

[17] A. Charnes, “Optimality and degeneracy in linear programming,” *Econometrica*, vol. 20, no. 2, pp. 160–170, 1952. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/0012-9682(195204)20:2&lt;160:OADILP&gt;2.0.CO;2-1

[18] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck, “Automation and combination of linear-programming based stabilization techniques in column generation,” *INFORMS*

Journal on Computing, vol. 30, no. 2, pp. 339–360, 2018. [Online]. Available: https://doi.org/10.1287/ijoc.2017.0784

[19] R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri, and E. Uchoa, “Primal Heuristics for Branch and Price: The Assets of Diving Methods,” *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 251–267, April 2019. [Online]. Available: https://ideas.repec.org/a/inm/orijoc/v31y2019i2p251-267.html

[20] G. B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, 1963.

[21] M. Fischetti, F. Glover, and A. Lodi, “The feasibility pump,” *Mathematical Programming*, vol. 104, no. 1, pp. 91–104, Sep 2005. [Online]. Available: https://doi.org/10.1007/s10107-004-0570-3

[22] T. Achterberg and T. Berthold, “Improving the feasibility pump,” *Discrete Optimization*, vol. 4, pp. 77–86, 03 2007.

[23] S. Irnich and G. Desaulniers, “Shortest path problems with resource constraints,” in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds. Boston, MA: Springer US, 2005, pp. 33–65. [Online]. Available: https://doi.org/10.1007/0-387-25486-2_2

[24] L. Desfontaines and G. Desaulniers, “Multiple depot vehicle scheduling with controlled trip shifting,” *Transportation Research Part B: Methodological*, vol. 113, pp. 34–53, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261517310652

[25] F. Quesnel, G. Desaulniers, and F. Soumis, “A new heuristic branching scheme for the crew pairing problem with base constraints,” *Computers & Operations Research*, vol. 80, pp. 159–172, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054816302957

[26] J. Gerbaux, “Résolution heuristique par génération de colonnes et apprentissage automatique du problème d’horaires d’autobus électriques,” Master’s thesis, Polytechnique Montréal, May 2023. [Online]. Available: https://publications.polymtl.ca/53409/

[27] J. Brasseur, “Accélération d’une méthode d’agrégation dynamique de contraintes par apprentissage automatique pour le problème de construction d’horaires de conducteurs d’autobus,” Master’s thesis, Polytechnique Montréal, August 2022. [Online]. Available: https://publications.polymtl.ca/10534/