

Vehicle routing problem with soft time windows: Analysis of computational complexity and sensitivity

Farouk O. Ligali

Master's Thesis
2025



Tomas Bata University in Zlín
Faculty of Applied Informatics

Tomas Bata University in Zlín
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence

Academic year: 2024/2025

ASSIGNMENT OF DIPLOMA THESIS

(project, art work, art performance)

Name and surname:	Farouk Olatunji Ligali
Personal number:	A23602
Study programme:	N0613A140023 Information Technologies
Specialization:	Software Engineering
Type of Study:	Full-time
Work topic:	Vehicle Routing Problems With soft Time Windows: Analysis of Computational Complexity and Sensitivity

Theses guidelines

1. Familiarize with vehicle routing and logistics topics and related combinatorial mathematical models. Then, focus on the vehicle routing with (soft) time windows.
2. Make a comprehensive review of existing formulations of time windows within the vehicle routing framework, emphasizing its soft variant. Discuss also computational aspects and solution methods.
3. Formulate a suitable mathematical model of the vehicle routing problem with soft time windows (based on the review) in a selected programming/optimization language (e.g., Python).
4. Create your own (artificial or real) example and solve it using an available optimization solver (e.g., CPLEX or Gurobi).
5. Consequently, real data will be implemented in the verified model and solved using a selected computational approach. Provide a thorough analysis of the problem and algorithmic aspects.
6. Suggest a solution for the practical use of a decision-maker when solving real problems and including new customers in existing routes. Analyze the complexity of such issues and solutions.
7. Properly discuss the obtained results, state and describe the limits of the selected approach(es), and formulate further research directions.

Form processing of diploma thesis:
Language of elaboration:

printed/electronic
English

Recommended resources:

1. GHIANI, Gianpaolo; LAPORTE, Gilbert and MUSMANN, Roberto, 2004. *Introduction to logistics systems planning and control*. Chichester: John Wiley. ISBN 0-470-84917-7.
2. TOTH, Paolo and VIGO, Daniele (ed.), 2014. *Vehicle routing: problems, methods, and applications*. Second edition. MOS-SIAM series on optimization. Philadelphia: Society for Industrial and Applied Mathematics. ISBN 978-1-61197-358-7.
3. FERNÁNDEZ GIL, Alejandro; LALLA-RUIZ, Eduardo; GÓMEZ SÁNCHEZ, Mariam and CASTRO, Carlos. The cumulative vehicle routing problem with time windows: models and algorithm. Online. *Annals of Operations Research*. ISSN 0254-5330. Available at: <https://doi.org/10.1007/s10479-022-05102-7>. [cit. 2024-10-26].
4. TAŞ, Duygu; DELLAERT, Nico; VAN WOENSEL, Tom and DE KOK, Ton, 2014. The time-dependent vehicle routing problem with soft time windows and stochastic travel times. Online. *Transportation Research Part C: Emerging Technologies*. Vol. 48, p. 66-83. ISSN 0968090X. Available at: <https://doi.org/10.1016/j.trc.2014.08.007>. [cit. 2024-10-26].

Supervisors of diploma thesis:

Ing. Dušan Hrabec, Ph.D.
Department of Mathematics

Date of assignment of diploma thesis: **October 27, 2024**

Submission deadline of diploma thesis: **June 2, 2025**

doc. Ing. Jiří Vojtěšek, Ph.D. m.p.
Dean



prof. Mgr. Roman Jašek, Ph.D., DBA m.p.
Head of Department

In Zlín November 17, 2024

MASTER'S THESIS AUTHOR'S DECLARATION

I take cognizance of the fact that

- by submitting my Master's thesis, I give consent to the publication of my thesis in accordance with the Act No. 111/1998 Coll., as amended, regardless of the result of the thesis defence;
- the Master's thesis will be stored in electronic form in the university information system and will be available for consultation;
- one copy of the Master's thesis in hard copy form will be kept and stored by Tomas Bata University in Zlín;
- my Master's thesis is fully subject to the Act No. 121/2000 Coll. on Copyright, on Rights Related to Copyright and on Amendment to Certain Acts (Copyright Act), as amended, particularly to Section 35 Paragraph 3;
- according to Section 60 Paragraph 1 of the Copyright Act, Tomas Bata University in Zlín has the right to conclude a licence agreement on the utilization of a school work as is specified in Section 12 Paragraph 4 of the Copyright Act;
- according to Section 60 Paragraphs 2 and 3, I may utilize my school work – Master's thesis – or grant a licence to utilize it only with the prior written consent of Tomas Bata University in Zlín, which in such a case is entitled to require me to pay a reasonable contribution in order to cover the costs incurred by Tomas Bata University in Zlín for the preparation of the work (up to their full amount);
- if software provided by Tomas Bata University in Zlín or other entities was used for the preparation of the Master's thesis only for learning or research purposes (i.e. for non-commercial use), the conclusions of the Master's thesis cannot be used for commercial purposes;
- if the output of the Master's thesis is any type of software product, the source codes and/or files of which the project is comprised are also considered part of the thesis; failure to submit this part may result in a failure to defend the thesis.

I declare that

- I have worked on my Master's thesis independently and properly cited the literature used; in case of publication of the conclusions, I shall be listed as co-author;
- the submitted hard copy version of the Master's thesis and the electronic version uploaded to the IS/STAG are identical in content.

In Zlín on

.....
author's signature

ABSTRAKT

Tato práce se zabývá problémem trasování vozidel s měkkými časovými okny (VRPSTW), což je kritická optimalizační výzva v logistice a řízení dopravy, kde je zákaznický servis flexibilní, ale penalizován za odchylky od stanovených servisních intervalů. Vzhledem k tomu, že striktní dodržování časových oken je často nepraktické nebo nákladné, výzkum systematicky zkoumá alternativní struktury penalizace – lineární, postupnou, exponenciální a kvadratickou – za účelem efektivní kvantifikace a řízení včasných a pozdních odchylek od servisu. Je formulován a implementován model smíšeného celočíselného lineárního programování (MILP) pomocí Pythonu a optimalizátoru Gurobi pro vyhodnocení dopadu každého systému penalizace na efektivitu trasování, náklady na penalizace a výpočetní výkon. Výpočetní experimenty jsou prováděny s použitím obou umělých instancí a poskytují srovnávací analýzu, která zdůrazňuje praktické důsledky každé penalizační funkce. Výsledky naznačují rozdíly ve výsledcích optimalizace trasy na základě zvoleného typu penalizace a vedou osoby s rozhodovací pravomocí k informovanějším a nákladově efektivnějším rozhodnutím ve složitých provozních prostředích. Zjištění nejen propojují teoretické modelování a praktické aplikace, ale také přispívají cennými poznatky pro další vývoj adaptivních, škálovatelných optimalizačních metod ve scénářích trasování vozidel.

Klíčová slova: Problém směřování vozidel, měkká časová okna, penalizační funkce, smíšené celočíselné lineární programování, optimalizace, logistika, Gurobi

ABSTRACT

This thesis addresses the Vehicle Routing Problem with Soft Time Windows (VRPSTW), a critical optimization challenge in logistics and transportation management, where customer service is flexible but penalized for deviations from specified service intervals.

Recognizing that rigid adherence to time windows is often impractical or costly, the research systematically investigates alternative penalty structures - linear, stepwise, exponential, and quadratic - to quantify and manage early and late service deviations effectively. A mixed-integer linear programming (MILP) model is formulated and implemented using Python and the Gurobi optimizer to evaluate each penalty scheme's impact on routing efficiency, penalty costs, and computational performance.

Computational experiments are conducted using both artificial instances, providing a comparative analysis that highlights the practical implications of each penalty function. Results indicate differences in route optimization outcomes based on the selected penalty type, guiding decision-makers toward more informed, cost-effective choices in complex operational environments. The findings not only bridge theoretical modeling and practical application but also contribute valuable insights for the continued development of adaptive, scalable optimization methods in vehicle routing scenarios.

Keywords: Vehicle Routing Problem, Soft Time Windows, Penalty Functions, Mixed-Integer Linear Programming, Optimization, Logistics, Gurobi

ACKNOWLEDGMENTS

First and foremost, I express my deepest gratitude to God Almighty, whose grace has sustained me throughout this journey. I am truly thankful for the gift of life, good health, wisdom, and understanding that enabled me to comprehend new and complex knowledge. It is through His guidance that I have been able to navigate a domain previously unfamiliar to me with patience and perseverance.

I wish to extend my sincere appreciation to Professor Lars M.H., from Molde University College, whose mentorship and tutelage has been instrumental in laying the foundational knowledge for this research. His generous dedication of time and effort - particularly in providing personal guidance and one-on-one discussions - helped me grasp the core principles necessary for the successful development of this thesis. His support has been invaluable, especially during moments of uncertainty, and I am deeply grateful for his commitment to my academic growth.

My heartfelt thanks also go to my supervisor, Dušan Hrabec, Ph.D., whose insight and encouragement have been pivotal throughout this research. His expertise in the application of mathematical concepts and interpretation not only broadened my perspective but also motivated me to delve deeper into this area of study. His continuous support and constructive feedback were key drivers in maintaining my focus and commitment to the project.

I am especially grateful to my family and friends for their unwavering support. To my beloved wife and son, your emotional and physical support, constant encouragement, and belief in my potential have been the backbone of my perseverance. Your sacrifices and understanding during this demanding period have meant the world to me.

A special thank you goes to my in-laws, brothers, and sisters, whose presence and encouragement have strengthened me in more ways than I can express. Your support has not gone unnoticed, and I cherish every contribution you have made toward my journey.

This accomplishment is not mine alone but belongs to everyone who stood by me. Thank you all for walking this path with me.

CONTENTS

INTRODUCTION	13
I THEORETICAL PART	15
1 LITERATURE REVIEW	16
1.1 COMPUTATIONAL COMPLEXITY OF VRP AND TIME WINDOW VARIANTS	17
1.2 EXACT SOLUTION METHODS	19
1.2.1 Branch-and-Bound (B&B)	19
1.2.2 Branch-and-Cut (B&C)	19
1.2.3 Branch-and-Price (B&P)	19
1.2.4 Constraint Programming (CP)	19
1.2.5 Dynamic Programming (DP)	20
1.2.6 Mathematical Programming Solvers	20
1.3 HEURISTIC AND METAHEURISTIC METHODS	21
1.3.1 Heuristics	21
1.3.2 Metaheuristics	21
1.4 HYBRID APPROACHES	22
1.5 COMPUTATIONAL ASPECTS AND PERFORMANCE EVALUATION	23
1.6 A COMPARATIVE LANDSCAPE OF ALGORITHMIC APPROACHES FOR VRP	24
1.6.1 Heuristic and Metaheuristic	25
1.7 SYNTHESIS AND RESEARCH AIM	36
II PRACTICAL PART	37
2 MATHEMATICAL MODEL	38
2.1 MODEL PENALTY FUNCTION	40
2.1.1 Penalty Function Design	40
2.1.2 Rationale for the Penalty Structure	42
2.2 CONSIDERATIONS FOR SELECTING PENALTY METHODS IN VRPTW	42
2.2.1 Penalty method	43
2.3 MODEL IMPLEMENTATION IN PYTHON (GUROBI)	48
2.4 MODEL ENHANCEMENTS FOR PENALTY FUNCTIONS	50
2.4.1 Linear Penalty	50
2.4.2 Stepwise Penalty	51

2.4.3	Exponential Penalty.....	51
2.4.4	Quadratic Penalty	52
3	RESULTS AND ANALYSIS	53
3.1	INSTANCE GENERATION AND EXPERIMENTAL CONFIGURATION	53
3.2	RESULT.....	54
3.2.1	Linear Penalty.....	54
3.2.2	Stepwise Penalty.....	56
3.2.3	Exponential Penalty	57
3.2.4	Quadratic Penalty	59
3.3	COMPARATIVE EVALUATION	61
3.3.1	Decision-making standpoint.....	62
	CONCLUSION.....	63
	BIBLIOGRAPHY	65
	LIST OF FIGURES	70
	LIST OF TABLES	71
	LIST OF ABBREVIATIONS	72
	APPENDICES	74

INTRODUCTION

The Vehicle Routing Problem with Soft Time Windows (VRPSTW) is an extension of the classical Vehicle Routing Problem (VRP), where each customer has a preferred time window within which service should ideally begin. In contrast to the hard time window variant (VRPHTW) where vehicles are strictly prohibited from arriving before the earliest start time or after the latest permissible time, the VRPSTW allows such time window constraints to be violated at the expense of an associated penalty cost or may also choose to wait after arriving if this is convenient.

In many practical applications, enforcing time windows as rigid constraints can lead to infeasibility or excessive costs. Soft time windows introduce flexibility by penalizing early or late arrivals, thus balancing the trade-off between route efficiency (e.g., travel distance, number of vehicles used) and the timeliness of deliveries or services. Typical examples include distribution logistics, parcel delivery, and public transportation scheduling, where punctuality is important but not absolute, and minor earliness or lateness may be acceptable at some cost. The goal is to determine a set of vehicle routes originating and ending at a central depot, such that all customers are served exactly once, vehicle capacity constraints are respected, and total costs (including routing costs and penalties for time window violations) are minimized.

E-commerce and same-day delivery have squeezed service windows so tightly that routing models with hard time constraints often become infeasible or unaffordable. Allowing soft windows and pricing deviations restores feasibility, yet the literature rarely compares how the shape of that price - linear, stepwise, exponential, quadratic - changes the routes a solver picks or the bill a carrier pays. Practitioners therefore lack guidance on how to tune penalties to contractual truth and avoid hidden cost spirals. This thesis tackles that gap by embedding multiple penalty shapes in a single model and quantifying their impact on synthetic instances.

This work makes four specific contributions:

- **Unified MILP formulation.** We extend the standard time-indexed VRPSTW model with auxiliary variables that let the same core constraints support *four* penalty types - linear, stepwise, exponential, and quadratic - without changing model topology .

- **Open-source implementation.** A modular Python/Gurobi codebase generates, solves and exports VRPSTW instances, enabling rapid “plug-and-play” testing of new penalty schedules.
- **Controlled computational study.** Using a curated 10-customer benchmark and a synthetic retail case study, we compare how each penalty shape influences route structure, cost split (travel vs. penalty), timing adherence and solve time.
- **Managerial guidance.** We derive practical rules for selecting a penalty function: e.g., quadratic penalties enforce on-time service at modest extra distance, while a linear slope can incentivise early delivery if priced too low. Sensitivity charts illustrate how raising or lowering penalty rates shifts optimal behaviour.

Together these contributions bridge theoretical modelling and day-to-day routing decisions, offering both a toolkit and evidence-based insights for planners.

Structure of the thesis

The remainder of the document is organised as follows:

- **Literature Review.** Surveys exact, heuristic and meta-heuristic approaches to VRP, VRPTW and VRPSTW, highlighting how penalties are currently modelled and where gaps remain.
- **Mathematical Model.** Presents the unified MILP, details each penalty formulation, and discusses implementation choices.
- **Results and Analysis.** Reports the computational experiments, compares penalty scenarios and performs sensitivity analyses.
- **Conclusions.** Summarises findings, managerial implications and avenues for future research.
- **Appendices.** Provide the full dataset, pseudocode and supplementary tables/figures.

I THEORETICAL PART

1 LITERATURE REVIEW

The Vehicle Routing Problem's (VRP) primary goal is to minimize cost through fundamental combinatorial approach by determining the optimal routes for a given set of vehicles for a certain set of customers (Dantzig & Ramser, 1959). When VRP is to be implemented in real-life scenarios to tackle distribution problems, there is the initiation of time windows (which is the varying delivering time for each customer) and other constraints like a certain vehicle can travel a particular terrain better, capacity of the vehicles, and much more. With this constraint taken into consideration, several variants of the classical VRP have emerged in literature, each tailored to model specific real-world constraints and operational complexities.

Capacitated Vehicle Routing Problem (CVRP): This is the most fundamental and widely studied VRP variant. It introduces vehicle capacity constraints, ensuring that the total demand of customers assigned to a single vehicle does not exceed its predefined carrying capacity. CVRP models situations such as goods distribution and waste collection where vehicle capacities cannot be violated (Golden, Raghavan, & Wasil, 2008).

Vehicle Routing Problem with Time Windows (VRPTW): If there is a possibility to be able to serve customers outside their time windows then this is regarded as a soft time window but if it is only possible to serve customers within their stipulated time window, this is regarded as hard time window (Miranda & Conceição, 2016). VRPTW extends the basic VRP by adding customer-specific time constraints. Each customer must be serviced within a specified time interval (time window). There are two subcategories: hard and soft time windows. Hard time windows strictly prohibit deliveries outside the stipulated time interval, while soft time windows permit deliveries outside these intervals at a penalty cost. VRPTW is particularly relevant for home deliveries, food distribution, and time-critical services (Solomon, 1987).

Vehicle Routing Problem with Soft Time Windows (VRPSTW): In this variant, violations of the specified time windows are allowed, but penalized. Penalties are typically proportional to the magnitude of the violation (early or late arrivals), thereby allowing more flexibility in route optimization at a manageable cost. VRPSTW captures real-world trade-offs between service level and operational costs, making it highly applicable in practice (Figliozzi, 2010).

Vehicle Routing Problem with Stochastic Demands (VRPSD): Unlike deterministic models, the VRPSD assumes that customer demands are not known precisely in advance but follow known probability distributions. Solutions to VRPSD seek to minimize expected costs, considering uncertainties in customer demand quantities. This variant is essential for modeling realistic logistics scenarios involving uncertain customer demands (Gendreau, Laporte, & Séguin, 1996).

Pickup and Delivery VRP (PDVRP): This variant considers that each vehicle is responsible for transporting goods from specific pickup locations to specified delivery locations. It generalizes the problem by introducing coupling constraints between pickup and delivery points, making routing solutions considerably more complex and realistic for parcel services and on-demand transportation (Parragh, Doerner, & Hartl, 2008).

Electric Vehicle Routing Problem (EVRP): Driven by sustainability considerations, EVRP incorporates additional constraints such as limited vehicle battery capacities, recharge station availability, and charging times. These constraints significantly alter route planning, highlighting the challenges faced by modern logistics operations using electric vehicle fleets (Schneider, Stenger, & Goeke, 2014).

Each variant of VRP introduces unique computational complexities and solution characteristics. Understanding these differences is crucial for selecting the appropriate solution methodologies, algorithms, and heuristic approaches suited to specific application contexts.

1.1 Computational Complexity of VRP and Time Window Variants

The VRP and its time-window extensions are well known to be computationally intractable in the general case. The classical VRP is an NP-hard combinatorial optimization problem (Lenstra, 2006). In fact, VRP contains the Traveling Salesman Problem (TSP) as a special case (e.g. with a single vehicle and no capacity constraints), and the TSP is NP-hard; thus VRP inherits this hardness result (Wang, Wang, Liu, & Yu, 2018). Lenstra and Rinnooy Kan formally proved the NP-hardness of the VRP (Lenstra, 2006), implying that no polynomial-time algorithm is known for solving it to optimality in general. The VRPTW is also NP-hard, since adding time window constraints does not simplify the problem, VRPTW reduces to the VRP when time windows are sufficiently large or ignored, so it remains at least as difficult (Pavone, Bisnik, Frazzoli, & Isler, 2008). Notably, even the feasibility problem of VRPTW with a fixed number of vehicles is NP-complete (deciding

if a set of deliveries can be scheduled into a fixed fleet is as hard as a bin-packing or partitioning problem) (Pavone, Bisnik, Frazzoli, & Isler, 2008). All these VRP variants belong to the class of NP-hard problems, which means their decision versions are NP-complete and no efficient (polynomial-time) algorithms are known for them (barring a breakthrough proving $P = NP$).

The NP-hard classification has deep implications for solving VRPs. Under the widely believed assumption that $P \neq NP$, no algorithm can guarantee an optimal solution for arbitrary VRP instances in time growing only polynomially with the input size. Instead, the worst-case runtime of exact methods grows exponentially with the number of customers or other problem parameters (Pavone, Bisnik, Frazzoli, & Isler, 2008). In practical terms, this means that exact solution approaches (such as brute-force search, branch-and-bound, branch-and-price, or dynamic programming) are only viable for relatively small instances (Lenstra, 2006). For example, classic VRP instances beyond only a few dozen customers become extremely difficult to solve optimally; VRPTW instances with around 100 customers are already *“very hard to solve optimally”* in practice (Nai-Wen & Chang-Shi, 2013). Academic and commercial solvers can handle small-to-moderate sized problems by exploiting structure and using clever pruning, but large-scale instances (e.g. hundreds or thousands of customers) are essentially intractable to solve to proven optimality (Suresh Nanda & Ramasamy, 2012). This intractability has driven researchers to develop a wide range of heuristic and metaheuristic algorithms for VRPs. Since finding the exact optimum is computationally prohibitive for big instances, heuristics, metaheuristics, and approximation methods are the primary tools to obtain high-quality solutions within reasonable time (Suresh Nanda & Ramasamy, 2012). In summary, the NP-hardness of VRP, VRPTW, and VRPSTW implies that these problems lie on the “NP” side of the famous P vs NP divide, and thus suffer from exponential growth in complexity as problem size increases, a hallmark of intractable optimization problems in practice.

There are several existing formulations for VRPTW by researcher and solution methods for various VRPs and their extensions to solve real-life problems. The solution methods can be broadly categorized into exact methods, heuristics, meta-heuristics and hybrid approaches with each having its own computational characteristics and suitability for different problem types and sizes. They are described below.

1.2 Exact solution methods

1.2.1 Branch-and-Bound (B&B)

In the eighties, B&B was regarded as the most effective exact approach for the CVRP (Matteo, Paolo, & Daniele, 1994). These methods incorporate combinatorial relaxations based on the Assignment Problem or the Shortest Spanning Tree and were able to solve instances with some tens of customers with the available hardware in the late eighties. Branching schemes, such as branching on arcs, are crucial components of these algorithms. Heuristic algorithms can also be integrated into B&B to find feasible solutions and potentially improve the incumbent ((Paolo & Daniele, 2014), Ch. 2, Sec. 2.2).

1.2.2 Branch-and-Cut (B&C)

This aim to strengthen the linear relaxation of VRP formulations by dynamically adding valid inequalities (cuts) ((Paolo & Daniele, 2014) Ch. 2, Sec. 2.4). The separation problem, which involves identifying violated cuts, can be computationally challenging. B&C is mentioned in the context of solving the Pickup and Delivery Problem with Stochastic Demands (VRPSD) using the integer L-shaped method.

1.2.3 Branch-and-Price (B&P)

This method is used to find optimal solutions, particularly when combined with column generation formulated using set partitioning ((Paolo & Daniele, 2014) Ch. 5, Sec. 5.3.1). It is mentioned as a successful exact approach to tackle deterministic and stochastic variants of VRPs (Duygu, Nico, Tom van, & Ton). In (Taş, Dellaert, Woensel, & Kok, 2014), the authors applied a column generation procedure embedded within a B&P method for VRP with soft time windows and stochastic travel times. The authors of (Salani & Battarra, 2018) applied branch-and-cut-and-price (BCP) algorithms for their variant of VRPSTW.

1.2.4 Constraint Programming (CP)

This is a paradigm for solving combinatorial problems by declaring decision variables with finite domains and specifying constraints over these variables. A CP solver then uses constraint propagation (e.g., arc consistency) and search to prune infeasible assignments and identify feasible solutions (Van Hentenryck, Michel, Perron, & Régin, 1999). In the VRP context, CP excels at modeling complex logical and scheduling constraints through global constraints such as circuit (ensuring route continuity) and cumulative (modeling

resource consumption over time). Cortés et al. (Cortés, Gendreau, Rousseau, Souyris, & Weintraub, 2014) demonstrated the effectiveness of CP within a B&P framework: they used CP to generate feasible routes for a technician dispatching problem with stochastic time windows, leveraging CP's powerful propagation to reduce the pricing subproblem complexity before passing columns to a master mixed-integer program (MIP). Therefore, modern exact methods for VRPTW and VRPSTW often embed CP within decomposition schemes where CP handles subproblem route generation under rich constraints, while a MIP or set-partitioning master model selects the optimal combination of routes (Cortés, Gendreau, Rousseau, Souyris, & Weintraub, 2014).

1.2.5 Dynamic Programming (DP)

This solves optimization problems by breaking them into overlapping subproblems and storing interim results to avoid redundant computations. For VRP variants, DP is most commonly applied to the elementary shortest path problem with resource constraints (ESPPRC), which is the pricing subproblem in column-generation approaches. It has been used for exact solutions, particularly for problems like the Orienteering Problem (Hayes & Norman, 1984), to site controls along a route, illustrating DP's power in small-scale routing under time constraints. In VRPTW and VRPSTW column generation, DP-based labeling algorithms solve the ESPPRC exactly, using advanced dominance rules to limit the number of labels (Taş, Dellaert, Woensel, & Kok, 2014). However, the worst-case number of labels grows exponentially with the number of nodes and resources, making DP a computational bottleneck in exact solvers for instances larger than ~50–100 customers. Therefore, DP approaches might not scale well to larger instances due to the state-space explosion (Steenbergen, Mes, & Heeswijk, 2023).

1.2.6 Mathematical Programming Solvers

Solvers, like CPLEX and Gurobi, can be used to solve Mixed Integer Programming formulations of VRPs. They solve VRP formulations by combining branch-and-bound, cutting planes, and heuristics (Gurobi Optimization, 2024). VRPSTW MIP models typically include binary routing variables, continuous time variables, and penalty variables for time window violations. However, for many VRP variants, especially larger instances, finding optimal solutions within reasonable time can be intractable (NP-hard) (He, Huang, & Yan, 2015).

1.3 Heuristic and Metaheuristic Methods

1.3.1 Heuristics

These are solution methods that can often find good-quality feasible solutions relatively quickly but without a guarantee of optimality ((Paolo & Daniele, 2014), Ch. 4). Examples include constructive heuristics like the savings algorithm (Clark & Wright, 1964) and insertion heuristics (Taş, Dellaert, Woensel, & Kok, 2014) (Ning & Zheng-ke, 2025). They often prioritize minimizing the number of vehicles and then the cost.

1.3.2 Metaheuristics

These are higher-level problem-solving strategies designed to guide underlying heuristics to explore the solution space more effectively and escape local optima. To tackle larger or more detailed VRPTW instances - especially when embedding complex penalty functions - meta-heuristics offer a powerful trade-off between solution quality and run-time. Some metaheuristics are:

- **Tabu Search (TS)** is a local search metaheuristic that uses a tabu list to prevent cycling and explore new solutions (Duygu, Nico, Tom van, & Ton) (Glover, Tabu Search - Part I, 1989) (Glover, Tabu Search - Part II, 1989). It can be combined with other techniques like heuristic DP or a scheduling method for post-optimization (Taş, Jabali, & Woensel, A Vehicle Routing Problem with Flexible Time Windows, 2014).
- **Adaptive Large Neighborhood Search (ALNS)** is a metaheuristic that iteratively destroys and repairs parts of a solution using a set of removal and insertion operators (Ning & Zheng-ke, 2025). It often incorporates mechanisms for adapting the selection probabilities of these operators based on their performance (Sun, Veelenturf, Hewitt, & Woensel, 2020). ALNS can be combined with exact solvers for subproblems (Merve, Gilbert, & Bülent, 2019).
- **Genetic Algorithms (GA)** are population-based metaheuristics that use principles of natural selection to evolve a set of candidate solutions (He, Huang, & Yan, 2015) (Holland, 1975). They can be less effective at exploring the solution space compared to combinations with local search.

- **Simulated Annealing (SA)** is a probabilistic metaheuristic that accepts worse solutions with a certain probability to escape local optima. It can be integrated into ALNS (Sun, Veelenturf, Hewitt, & Woensel, 2020).
- **Ant Colony System** is a population-based metaheuristic inspired by the foraging behavior of ants, used for problems like the Site-Dependent Vehicle Routing Problem with Soft Time Windows (SDVRPSTW) (Esmat & S. Hamid, 2015).
- **Particle Swarm Optimization** is a population-based metaheuristic where particles move in the solution space based on their own best-known position and the best-known position of the entire swarm. It can be combined with GA for local search enhancement (He, Huang, & Yan, 2015).
- **Electromagnetism (EM) Algorithm** is a metaheuristic that can be hybridized with column generation (Chang, Chen, & Fan, 2009) (Beheshti & Hejazi, 2015).
- **Maximum Entropy Principle based approach** is a deterministic approach related to SA that has been applied to various routing problems (Baranwal, Marla, Beck, & Salapaka, 2022).
- **Greedy Randomized Adaptive Search Procedure (GRASP)** is a multi-start metaheuristic that combines constructive and local search phases and can be integrated with mathematical programming for partial optimization in a metaheuristic approach (Alejandro Fernández, Eduardo, María Carmen Gómez, & Carlos, 2023).
- **Artificial Bee Colony (ABC) algorithm** is a swarm-based metaheuristic that has been used for solving Multi-Objective VRPs (Iqbal, Kaykobad, & Rahman, 2015).

1.4 Hybrid Approaches

Many successful solution methods for complex VRPs involve hybridizing exact and heuristic techniques or combining different metaheuristics. Some hybridization examples include:

- Combining column generation with metaheuristics like QIEA and EM (Beheshti & Hejazi, 2015)
- Integrating heuristic DP with TS (Han, Zhao, Chen, Luo, & Mishra, 2017).

- Using ALNS to search the feasible space while optimizing charging decisions in the EVRP using exact solvers (Merve, Gilbert, & Bülent, 2019).
- Developing matheuristics that combine metaheuristics like GRASP with exact optimization techniques for subproblems (Alejandro Fernández, Eduardo, María Carmen Gómez, & Carlos, 2023).
- Embedding column generation within a B&P scheme (Taş, Dellaert, Woensel, & Kok, 2014).

1.5 Computational Aspects and Performance Evaluation

Computational Complexity: Many VRPs and their variants are NP-hard, implying that finding optimal solutions can be computationally expensive, especially for large instances (Taş, Dellaert, Woensel, & Kok, 2014) (He, Huang, & Yan, 2015) (Merve, Gilbert, & Bülent, 2019).

Benchmark Instances: The performance of algorithms is often evaluated using standard benchmark instances such as Solomon's instances (Solomon, 1987) for VRPTW (Salani & Battarra, 2018) (Narath, Ali Gul, & Eiichi, 2014) (Steenbergen, Mes, & Heeswijk, 2023) and Gehring and Homberger's larger problem instances (Gehring & Homberger, 1999). Modified versions of these benchmarks are also used to test specific problem features or to evaluate robust VRP variants (Alejandro Fernández, Eduardo, María Carmen Gómez, & Carlos, 2023).

Performance Metrics: Common metrics for evaluating solution quality include total cost, total distance, number of vehicles used, total expected delay, earliness, overtime, and penalty costs for violating soft time windows (Ning & Zheng-ke, 2025) (Duygu, Nico, Tom van, & Ton). Computational time is also a crucial performance indicator (Narath, Ali Gul, & Eiichi, 2014) (He, Huang, & Yan, 2015). Computational times are sometimes normalized to account for differences in computing hardware.

Computational Resources (hardware and software): Experiments are typically run on standard computer hardware with specific processors and memory (Narath, Ali Gul, & Eiichi, 2014) (He, Huang, & Yan, 2015). Time limits are often imposed on exact methods, and optimality gaps are reported when optimal solutions are not found within the limit (Steenbergen, Mes, & Heeswijk, 2023).

Scalability: A key challenge is developing algorithms that can efficiently solve large-scale real-world VRP instances (Taş, Dellaert, Woensel, & Kok, 2014). Decomposition techniques are used to handle larger problems by dividing them into smaller subproblems (Jie, Liu, & Sun, 2022).

Hybridization Benefits: Hybrid algorithms often demonstrate improved performance compared to pure exact or heuristic methods, leveraging the strengths of different approaches (Beheshti & Hejazi, 2015) (Merve, Gilbert, & Bülent, 2019) (Han, Zhao, Chen, Luo, & Mishra, 2017).

1.6 A Comparative Landscape of Algorithmic Approaches for VRP

The VRP stands as a cornerstone in the fields of logistics and operations research, addressing the fundamental challenge of designing optimal delivery or service routes for a fleet of vehicles to serve a set of customers (Wan Chen & Chen Yang, 2017). Given the practical complexities and diverse constraints often encountered in real-world scenarios, such as time windows, stochastic travel times, and multiple vehicle types, a multitude of algorithmic approaches have been developed and refined over the years (Duygu, Nico, Tom van, & Ton). Understanding the strengths and limitations of these diverse methodologies is crucial for researchers and practitioners alike, guiding the selection of the most appropriate tools for specific problem contexts and inspiring further advancements in the field (Paolo & Daniele, 2014).

To provide a structured overview of prominent solution techniques for the VRP and its variants, the following table synthesizes information gleaned from recent research. It offers a comparative analysis of several key algorithms, highlighting their respective strength and limitation as reported in the literature, and provides specific instances of their application and evaluation on benchmark problems. This compilation aims to facilitate a deeper understanding of the current algorithmic landscape and inform future investigations into optimizing vehicle routing operations.

1.6.1 Heuristic and Metaheuristic

Table 1 Comparative Analysis of Heuristic and Metaheuristic Algorithms

Algorithm	Pros	Cons
Tabu Search (TS)	Obtained most of the best solutions for 100-customer instances (Solomon's benchmarks) when service times were not included (Duygu, Nico, Tom van, & Ton). Yielded better final solutions for 200-customer instances in both cases (with and without service times). Can be well integrated with some heuristics (Han, Zhao, Chen, Luo, & Mishra, 2017).	Generally required longer computation times compared to ALNS, especially for 200-customer instances (Duygu, Nico, Tom van, & Ton). Computational analysis details (speed, scalability) are not extensively provided in some excerpts. (Duygu, Nico, Tom van, & Ton)
Adaptive Large Neighborhood Search (ALNS)	Effective in obtaining very good solutions for well-known problem instances (Duygu, Nico, Tom van, & Ton). Provided shorter computation times than TS for 200-customer instances (Duygu, Nico, Tom van, & Ton). Can obtain high-quality solution under a shorter computation time compared with the state-of-the-art MIP solver (Gurobi) for instances up to N=25 (Ning & Zheng-ke, 2025). Adaptable to standard VRP, VRPTW, and SDVRPSTW with comparable performance to standard ALNS (Ning & Zheng-ke, 2025).	TS often achieved slightly better final solutions for 200-customer instances (Duygu, Nico, Tom van, & Ton). Performance can be affected by profit distribution among requests, potentially leading to getting stuck in local optima for some single-vehicle instances (Sun, Veelenturf, Hewitt, & Woensel, 2020).

	Can handle large-sized instances within relatively short CPU time (Sun, Veelenturf, Hewitt, & Woensel, 2020).	
Genetic Algorithms (GA)	A hybrid GA showed viability and superiority, saving computing and iteration times compared to unimproved GA (Yang, et al., 2022).	Detailed computational analysis regarding performance, speed, and scalability is not prominent in the provided excerpts for general GA applications to VRPSTW. Can require significant computational time for larger instances (e.g., 2-3.5 hours for Agent-10 and Agent-30 instances) (Zhou, Kris, Creighton, & Wu, 2022)
Artificial Bee Colony (ABC)	Hybrid ABC algorithm (Bee_VRPSTW) provided competitive solutions with the best-known results (Yang, et al., 2022). Achieved best results within a moderate amount of time, deemed reasonable for an offline problem (Beheshti & Hejazi, 2015). Was reasonably faster than some other compared algorithms (Beheshti & Hejazi, 2015) (Iqbal, Kaykobad, & Rahman, 2015). Successful in finding optimal or near-optimal solutions within a reasonable	Computational time could be longer when aiming for higher quality solutions (Iqbal, Kaykobad, & Rahman, 2015).

	computation time for small-size test problems (Baradaran, Shafaei, & Hosseinian, 2019).	
Iterated Local Search (ILS)	Adapted MS-ILS often found solutions with 100% non-violated time windows and sometimes decreased the best-known distance (Taş, Jabali, & Woensel, A Vehicle Routing Problem with Flexible Time Windows, 2014) (Beheshti & Hejazi, 2015) (Michallet, Prins, Amodeo, Yalaoui, & Vitry, 2014).	Reported CPU times indicated that MS-ILS was more time-consuming than its competitors (Beheshti & Hejazi, 2015) (Michallet, Prins, Amodeo, Yalaoui, & Vitry, 2014). This was attributed to the generality of the method and unsimplified code
Hybrid Approaches	Aim to leverage the strengths of different approaches, often showing promising results for both solution quality and efficiency across different dataset sizes. Dependent Vehicle Routing Problem with Soft Time Window in terms of the number of vehicles, total transportation distance, and total waiting time compared to other heuristics (Sun, Veelenturf, Hewitt, & Woensel, 2020) (Beheshti & Hejazi, 2015). GRASP matheuristic with exact optimization provided optimal solutions for smaller instances and feasible/improved routes for medium and large instances (Sun, Veelenturf, Hewitt, & Woensel, 2020).	Calculation time for some hybrid approaches can increase faster with the number of customer points, suggesting room for improvement in speed for very large datasets (Sun, Veelenturf, Hewitt, & Woensel, 2020). Performance depends heavily on the specific combination of algorithms and their integration strategy.

	Could reduce fuel consumption and emissions by allowing soft time windows.	
--	--	--

Table 2 Summary of Soft Time Window Formulations in VRPSTW Literature

Paper (Authors, year)	Soft time window definition	Objective function	Mathematical formulation
Calvete et al. (2007) (Calvete, Galé, Oliveros, & Sánchez-Valverde, 2007)	Linear (Goal programming): Customer STW preferences with linear penalty for early or late service (per time unit deviation from desired window) Two-phase exact approach: generate all feasible routes then select optimal set via a set-partitioning ILP (solved to optimality)	Multi-objective: minimize travel cost while also minimizing time-window violations (and other goals like maximizing vehicle capacity utilization). STW penalties are incorporated as one of the objectives (handled via goal programming weights).	$\min \ddot{W} \ddot{g}^+ + \sum_{i=2}^n \widehat{W}_i \widehat{g}_i^- + \sum_{i=2}^n \widetilde{W}_i \widetilde{g}_i^+ + \sum_{k=1}^V \dot{W}_k \dot{g}_k^- + \sum_{k=1}^V \check{W}_k \check{g}_k^-$ Where: $\ddot{W}, \ddot{g}^+$: related to operational cost (goal 1) $\widehat{W}_i, \widehat{g}_i^-$: early arrival deviation (goal 2) $\widetilde{W}_i, \widetilde{g}_i^+$: late departure deviation (goal 2) $\dot{W}_k, \dot{g}_k^-$: vehicle capacity underuse (goal 3) $\check{W}_k, \check{g}_k^-$: labour time underuse (goal 4)
Russell & Urban (2008) (Russell & Urban, 2008)	Soft time windows with early and late penalties, modelled as fixed, linear, or quadratic cost penalties. A stochastic model using Erlang-distributed travel times. Solved via tabu-search metaheuristic with	Minimize number of vehicles, total travel distance, and expected penalty due to time window violations.	Minimize $1000V + \alpha D + \beta P$ Where: V: Number of vehicles used D: Total distance travelled

	post-optimization. Time windows influence expected penalties via probabilistic functions.		P: Total expected penalty (early + tardy) due to time window violations α, β : Weighting coefficients to reflect application-specific costs, e.g., ($\alpha = 1.0, \beta = 0.2$) or ($\alpha = 0.5, \beta = 0.5$) in their experiments
Qureshi et al. (2009) (Qureshi, Taniguchi, & Yamada, 2009)	Semi Soft time window: vehicle may arrive after the latest time at a customer with a lateness penalty (linear cost per unit time late). Early arrivals are handled by waiting (no early penalty in this model). B&P (column generation for set-partitioning master problem) to find optimal routes, embedded in a branch-and-bound framework.	Minimize total cost (primarily travel distance/time and vehicle usage) + lateness penalties. In the dynamic VRPSTW, the objective at each re-optimization is to minimize current routing cost plus any incurred overtime (lateness) cost.	$\min \sum_{k \in K} \sum_{(i,j) \in A} c'_{ijk} x_{ijk}$ x_{ijk} is a binary variable indicating whether vehicle k travels directly from customer i to customer j. c'_{ijk} is the modified time-dependent travel cost, incorporating both travel cost and late arrival penalty, defined below as: $c'_{ijk} = \begin{cases} c_{ij}, & \text{if } S_{jk} \leq b_j \\ c_{ij} + c_i(S_{jk} - b_j), & \text{if } S_{jk} > b_j \end{cases}$ Where: c_{ij} is the base cost of traveling from customer i to j S_{jk} is the service start time at customer j by vehicle k b_j is the latest acceptable start time at customer j (original time window)

			c_l is the unit penalty cost for late arrival.
Bhusiri et al. (2014) (Bhusiri, Qureshi, & Taniguchi, 2014)	Time-dependent soft time windows: service can start after the desired latest time, incurring a time-proportional penalty (penalty rate may depend on how late the service is) . Early arrivals assumed waiting (focus is on late penalties). Column Generation + B&P (set partitioning master with a specialized shortest-path pricing). Novel labeling and dominance techniques are used to handle time-dependent penalty costs in the pricing subproblem.	Multi-objective: Minimize total route cost, which includes: travel cost, fixed vehicle cost, early arrival penalty (waiting cost), and late arrival penalty. The objective is to also minimize waiting time, supporting better resource utilization.	$\min \sum_{k \in K} \sum_{(i,j) \in A} c'_{ij} x_{ijk}$ Where: x_{ijk}: Binary variable (1 if vehicle k uses arc (i,j) else 0) c'_{ij} is the modified time-dependent travel cost $c'_y = \begin{cases} \infty & \text{if } p_{jk} < a'_j \\ c_{ij} + c_e(a_j - p_{jk}) & \text{if } a'_j \leq p_{jk} < a_j \\ c_{ij} & \text{if } a_j \leq p_{jk} \leq b_j \\ c_{ij} + c_l(p_{jk} - b_j) & \text{if } b_j < p_{jk} \leq b'_j \\ \infty & \text{if } p_{jk} > b'_j \end{cases}$ p_{jk}: Arrival time of vehicle k at customer j c_e, c_l: Unit early/late penalty costs $[a'_j, b'_j]$: Extended time window limits
Taş et al. (2014) (Taş, Dellaert, Woensel, & Kok, 2014)	Soft TW with early and late penalties: Vehicles may serve a customer before the earliest or after the latest time at a penalty cost for each unit of earliness or lateness. Column Generation + B&P: the master is a classical set-partitioning	Minimizes the total weighted cost combining: Service costs: Penalties for early/late arrivals. $c_d \sum D_{jv}(x)$: Total delay penalties.	$\min \rho \frac{1}{C_1} \left(c_d \sum_{j \in N} D_{jv}(x) + c_e \sum_{j \in N} \sum_{v \in V} E_{jv}(x) \right) + (1 - \rho) \frac{1}{C_2}$

	model. The pricing subproblem is an ESPPRC that accounts for time-window violations. An optimal integer solution is obtained by a B&P algorithm (column generation with branch-and-bound).	$c_e \sum E_{jv}(x)$: Total earliness penalties. Transportation costs: $c_t \sum d_{ij} x_{ijv}$: Total travel distance cost. $c_f \sum x_{0jv}$: Fixed cost per vehicle used. $c_o \sum O_v(x)$: Total driver overtime cost. Weighting: Balanced by ρ (service cost weight) and $(1 - \rho)$ (transportation cost weight).	$\left(c_t \sum_{j \in N} \sum_{v \in V} d_{ij} x_{ijv} + c_f \sum_{j \in N \setminus \{0\}} \sum_{v \in V} x_{0jv} + c_o \sum_{v \in V} O_v(x) \right)$ Variables: - $D_{jv}(x)$: Expected delay at node j for vehicle v. - $E_{jv}(x)$: Expected earliness at node j for vehicle v. - $O_v(x)$: Expected overtime for vehicle v. - x_{ijv}: Binary variable. 1 if arc (i, j) is traversed by vehicle v, else 0. Parameters: - c_d, c_e: Unit penalties for delay/earliness. - c_t, c_f, c_o: Costs per distance, vehicle, and overtime unit. - C_1, C_2: Scaling factors for normalization.
--	---	--	---

Wang et al. (2018) (Wang, Wang, Liu, & Yu, 2018)	Soft TW as customer dissatisfaction: If a delivery occurs outside the preferred time window, the deviation (earliness or lateness) contributes to a dissatisfaction metric (assumed linear with violation time). Both early and late arrivals are penalized in this model to measure total violation. Mixed-Integer Linear Programming(MILP) model solved exactly for small instances. They apply an augmented ε-constraint method (iteratively solving single-objective MILPs with CPLEX 12.6) to obtain the optimal Pareto front of the bi-objective problem . For larger instances, Heuristic methods were used but the exact CPLEX approach finds optimal solutions for smaller sizes.	Bi-objective MILP: (1) Minimize total energy consumption (f_1) (2) Minimize customer dissatisfaction (f_2)	$\min f_1 = \sum_{k \in K} \sum_{i \in N_0} R_{ik}$ $\min f_2 = \sum_{k \in K} \sum_{i \in N_1} L_{ik}$ Where: N_0: is number of departure and arrival depots N_1: number of customers K: number of vehicles R_i^k: the cumulative energy consumption of vehicle k at vertex i, $i \in N_0 \cup N_1$ and $k \in K$ L_{ik}: tardiness of vehicle k at vertex i, $L_i^k = \max\{0, a_i^k - l_i\}$, $i \in N_1$ $k \in K$
---	--	---	--

Eydi and Ghasemi-Nezhad (2021) (Eydi & Ghasemi-Nezhad, 2021)	Dual-demand penalty: Customers have two demand levels: First demand (d_i^1): Fully satisfied if served within the time window $([l_i, u_i])$. Second demand (d_i^2): Reduced demand (lower than (d_i^1) if served outside the window. No explicit cost penalty; penalty manifests as reduced demand fulfillment.	Bi-Objective Optimization: Generates Pareto-optimal solutions via modified NSGA-II, ensuring no objective is sacrificed without explicit trade-offs. First objective is to minimize total traveling costs Second is to maximize customer demand satisfaction	$\text{Min} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^v c_{ij} x_{ijk}$ $\text{Max} \sum_{i=1}^n [d_i^1(q_i + w_i) + d_i^2(1 - q_i - w_i)]$ Where: c_{ij}: Travel cost between nodes i and j x_{ijk}: Binary variable indicating if vehicle k travels from i to j d_i^1: Primary demand (fully met if served within $([l_i, u_i])$). d_i^2: Secondary demand (lower than d_i^1, met if served outside the window). q_i, w_i: Binary variables indicating service within or before the time window.
---	--	---	---

Ismail et al. (2021) (Ismail, et al., 2021)	Time windows may be violated with a penalty cost added to the total cost. The penalty is proportional to the amount of time violated. MILP formulated and solved using Branch-and-Bound in LINGO. Vehicles are heterogeneous. Key constraints include vehicle capacity, route continuity, and service time.	Minimize total transportation cost including route cost and penalty cost due to time window violations (early and late arrival).	$\begin{aligned} \text{Minimize } & \sum_{k=1}^K \sum_{i=0}^N \sum_{j=0}^N C_{ijk} X_{ijk} \\ & + \sum_{k=1}^K \sum_{j=1}^N PE_C_j \cdot PE_T_{jk} \\ & + \sum_{k=1}^K \sum_{j=1}^N PL_C_j \cdot PL_T_{jk} \end{aligned}$ Where: C_{ijk} : Travel cost between node i and j by vehicle k PE_C_j, PL_C_j : Penalty cost for early/late arrival at customer j PE_T_{jk}, PL_T_{jk} : Binary indicator for early/late arrival violations X_{ijk} : Binary decision variable if vehicle k goes from node i to j
--	---	---	---

1.7 Synthesis and Research Aim

The reviewed literature underscores the critical role of penalty structures and algorithmic efficiency in addressing VRPSTW. While exact methods (Section 1.2) guarantee optimality for small instances, their scalability limitations (Section 1.5) highlight the need for hybrid or heuristic approaches, particularly for real-world applications involving dynamic customer insertion or stochastic travel times. Metaheuristics like ALNS and TS (Table 1) demonstrate superior computational performance for medium-to-large instances but often lack integration with rigorous penalty frameworks that balance cost and service quality. Existing penalty formulations (Table 2) further reveal a trade-off: linear penalties (e.g., Calvete et al. (Calvete, Galé, Oliveros, & Sánchez-Valverde, 2007)) prioritize computational tractability, whereas quadratic or stepwise penalties better reflect real-world logistics contracts but complicate optimization. Building on these insights, this research develops a flexible penalty-based MILP framework that incorporates not only linear penalties but also piecewise, quadratic and exponential penalty structures to better capture the trade-off between route efficiency and time window flexibility. The model is implemented in Python/Gurobi and validated through a series of artificial and real-world case studies (Sections 4–5), where sensitivity analyses quantify the impact of varying penalty parameters on routing costs (Figure 1). This work thus offers decision-makers a robust, scalable toolkit for balancing operational costs with service reliability in complex logistics settings.

II PRACTICAL PART

2 MATHEMATICAL MODEL

A common mathematical formulation for the VRPSTW with penalty costs for time-window violations is given below.

Objective Function

$$\text{minimize } \sum_{i \in N} \sum_{j \in N \setminus \{i\}} \sum_{k \in V} c_{ij} x_{ijk} + \sum_{i \in N_C} (c_i^E p_i^E + c_i^L p_i^L) \quad (1)$$

The objective function, equation (1) minimizes the total cost, which is the sum of the transportation costs and the penalty costs for early or late arrivals.

Decision Variables

x_{ijk} : Binary variable; 1 if vehicle k travels from i to j , 0 otherwise.

t_i : Time of servicing at node i .

p_i^E : Penalty on early arrival at node i .

p_i^L : Penalty on late arrival at node i .

Parameters

$[A_i, B_i]$: Time window for node i .

Q : Capacity of each vehicle.

d_i : Demand at node i .

T_{ij} : Travel time from node i to node j .

c_{ij} : Travel cost from node i to node j .

c_i^L : Penalty cost for late arrival at node i .

c_i^E : Penalty cost for early arrival at node i .

$N_C = \{1, 2, \dots, n\}$: Customer nodes.

$N = N_C \cup \{0\}$: All nodes, including the depot (0).

$V = \{1, 2, \dots, m\}$: Vehicles.

Constraints

$$\sum_{j \in N \setminus \{i\}} \sum_{k \in V} x_{ijk} = 1 \quad \forall i \in N_C \quad (2)$$

$$\sum_{j \in N} x_{0jk} = 1 \quad \forall k \in V \quad (3)$$

$$\sum_{i \in N} x_{i,0,k} = 1 \quad \forall k \in V \quad (4)$$

$$\sum_{i \in N \setminus \{h\}} x_{ihk} - \sum_{j \in N \setminus \{h\}} x_{hjk} = 0 \quad \forall h \in N_C, \forall k \in V \quad (5)$$

$$t_i \geq T_{0i} - T_{0i} \left(1 - \sum_{k \in V} x_{0ik} \right) \quad \forall i \in N_C \quad (6)$$

$$t_j \geq t_i + T_{ij} - (B_0 + T_{ij}) \left(1 - \sum_{k \in V} x_{ijk} \right) \quad \forall i, j \in N_C, j \neq i \quad (7)$$

$$t_i + T_{i0} \leq B_0 \quad (8)$$

$$\sum_{i \in N_C} d_i \sum_{j \in N \setminus \{i\}} x_{ijk} \leq Q \quad \forall k \in V \quad (9)$$

$$P_i^E \geq A_i - t_i, \quad P_i^E \geq 0 \quad \forall i \in N \quad (10)$$

$$P_i^L \geq t_i - B_i, \quad P_i^L \geq 0 \quad \forall i \in N \quad (11)$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j, k \quad (12)$$

$$t_i \geq 0 \quad \forall i \quad (13)$$

The flow conservation constraints (2) guarantee that each customer node is visited exactly once by a single vehicle. Constraints (3) state that every vehicle must leave the depot exactly once, while constraints (4) state that the vehicle must return to the depot exactly once. Constraints (5) ensures that for every vehicle entering a node, the number of times it enters and leaves the node must be equal to zero. The time constraints (6) ensure that service at each customer begins after the vehicle's arrival. Considering travel times between nodes, constraints (7), prevents unrealistic loops and ensures vehicles follow the correct travel sequence, and constraints (8) ensures that vehicles return to the depot within the specified time window before its closing time. The load constraints (9) ensures total

demand of customers served by any vehicle must not exceed the vehicle's capacity Q . Additionally, the model incorporates penalty constraints (10) and (11) which calculates the penalties for early and late arrivals respectively, ensuring that the service time at each customer node t_i must lie within the specified time window $[A_i, B_i]$. Lastly, the variable domains (12) define the binary nature of the routing variables, indicating whether a vehicle travels between two nodes, and constraints (13) ensures non-negative service times.

2.1 Model Penalty Function

In this study, a soft time window penalty function is employed to regulate deviations from the preferred service time windows of customers. The penalty function is designed to balance the need for scheduling flexibility with the objective of minimizing disruptions caused by early or late arrivals. This section provides a detailed explanation of the penalty structure, its mathematical formulation, and its practical implications.

2.1.1 Penalty Function Design

The penalty function in this model follows a piecewise **linear structure**, applying penalties to both early and late arrivals while allowing for a limited degree of schedule flexibility.

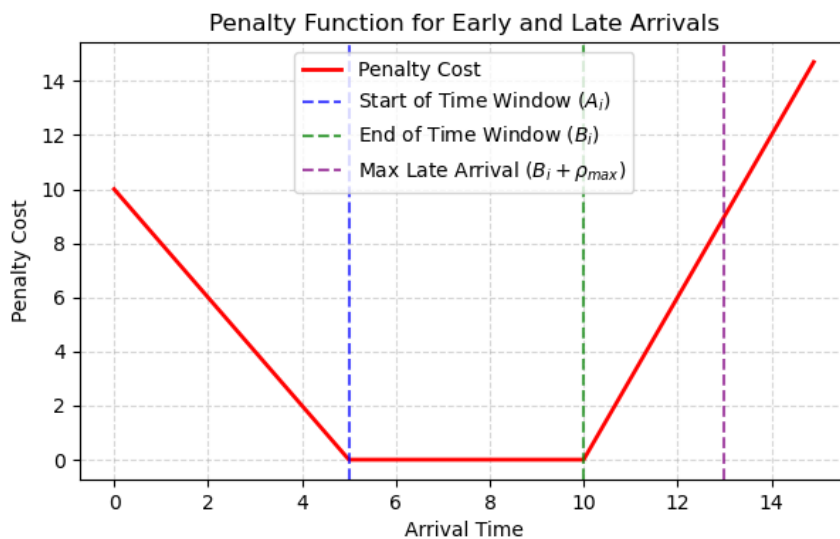


Figure 1: Penalty function for Early and Late Arrivals

(source: own elaboration)

The graph above illustrates the penalty cost associated with early and late arrivals:

- The x-axis represents the arrival time of a vehicle at a customer.

- The y-axis represents the penalty cost incurred due to deviations from the ideal service window.
- The blue dashed line at $A_i = 5$ represents the start of the time window.
- The green dashed line at $B_i = 10$ represents the end of the time window.
- The purple dashed line at $B_i + \rho_{max} = 13$ represents an implicit “latest possible time” before service is denied. We refer to this as an *artificial latest service time*—it’s not a penalty or prohibition on the customer, but a side-effect of forcing every route to finish at the depot before B_0 .

Specifically, the penalty function is defined as follows:

1. No Penalty Zone

A vehicle that arrives within the customer’s designated time window (A_i, B_i) incurs no penalty. This range represents the ideal service time during which customer satisfaction is maximized.

2. Early Arrival Penalty

If a vehicle arrives before the lower bound A_i a penalty is applied at a rate proportional to the difference between the actual arrival time and A_i . The penalty function for early arrivals is given by:

$$p_i^E = \max(0, A_i - t_i) \quad (14)$$

$$Total_t^E = p_i^E * c_i^E \quad (15)$$

where c_i^E is how much it cost per unit being early at customer i , p_i^E is how much late the visit is made and $Total_t^E$ represents the total early arrival penalty.

3. Late Arrival Penalty

If a vehicle arrives after the upper bound (B_i), a linear penalty is applied, increasing proportionally with the lateness duration. The penalty function for late arrivals is given by:

$$p_i^L = \max(0, t_i - B_i) \quad (16)$$

$$p_t^L = p_i^L * c_i^L \quad (17)$$

where c_i^L is how much it cost per unit being late at customer i , p_i^L is how much late the visit is made, and p_i^L represents the total late arrival penalty.

The total penalty cost across all customers is incorporated into the objective function as:

$$\sum_{i \in N_C} (c_i^E p_i^E + c_i^L p_i^L) \quad (18)$$

where N_C represents the set of all customers.

2.1.2 Rationale for the Penalty Structure

The piecewise linear penalty function is chosen due to its computational efficiency and practical applicability. Unlike quadratic or exponential penalties, which impose harsher penalties for deviations, a linear penalty provides a gradual increase in cost, encouraging on-time arrivals without making small deviations prohibitively expensive. The benefits of this approach include:

- **Flexibility in Scheduling:** Small deviations from the preferred service time are tolerated, allowing for practical adjustments in real-world vehicle routing.
- **Cost-Effectiveness:** The penalty function discourages extreme deviations without making minor delays excessively costly.
- **Computational Simplicity:** A linear penalty structure is easier to integrate into mixed-integer linear programming (MILP) models compared to nonlinear penalty functions.

To better understand the behaviour of the penalty function, Figure 1 presents a graphical representation of the applied penalties. The figure illustrates how penalty costs increase linearly for early and late arrivals while maintaining a no-penalty zone within the time window. Additionally, the penalty function enforces a maximum limit on late arrivals, beyond which service is infeasible.

2.2 Considerations for Selecting Penalty Methods in VRPTW

When dealing with time-sensitive deliveries, such as perishable goods or critical medical supplies, it is essential to implement a penalty method that emphasizes punctuality. In these cases, harsher penalties - such as exponential penalties - can deter late arrivals by imposing steep costs on significant violations. Conversely, in contexts where slight

deviations are acceptable, such as flexible delivery services or non-critical shipments, softer penalties like quadratic penalties may be more appropriate. These methods allow for some leeway while still discouraging excessive tardiness or earliness.

Moreover, industry-specific factors can influence the choice of penalty method. For example, retail and distribution logistics often employ stepwise penalties to reflect real-world tolerance thresholds for delays. In contrast, logistics planning may benefit from linear delay costs that align directly with operational expenses such as fuel consumption and labour costs. Therefore, understanding the unique characteristics of each application is vital in selecting a penalty method that balances operational efficiency with customer satisfaction.

Ultimately, the decision to adopt a particular penalty approach should be guided by a thorough analysis of the specific requirements and constraints of the routing problem at hand. By carefully considering these factors, practitioners can enhance their routing solutions and better meet service level agreements while minimizing operational costs.

2.2.1 Penalty method

1. Linear Penalty

The linear penalty is the simplest way to price schedule deviations: every minute early or late costs the same amount. It is the base penalty in the objective function (1).

$$P_i^E = \max(0, A_i - t_i) \quad (19)$$

$$P_i^L = \max(0, t_i - B_i) \quad (20)$$

$$\text{Total}_i = (c_i^L \times P_i^L) + (c_i^E \times P_i^E) \quad (21)$$

where t_i is the actual arrival time at customer i , and c_i^E and c_i^L are cost per unit of being early and late at customer i , respectively. The total being the sum of product of the cost and penalty for early and late arrivals.

Advantages: Charges a constant rate for every minute of earliness or lateness, making it easy to interpret and explain to planners and drivers. Adds only linear terms and two continuous slack variables per customer (no binaries), so the MILP remains convex and solves very quickly.

Disadvantages: Penalises small and large deviations proportionally, which may under-penalise very late arrivals and over-penalise trivial timing slips. It lacks the ability to sharply deter extreme delays - every extra minute costs the same regardless of context.

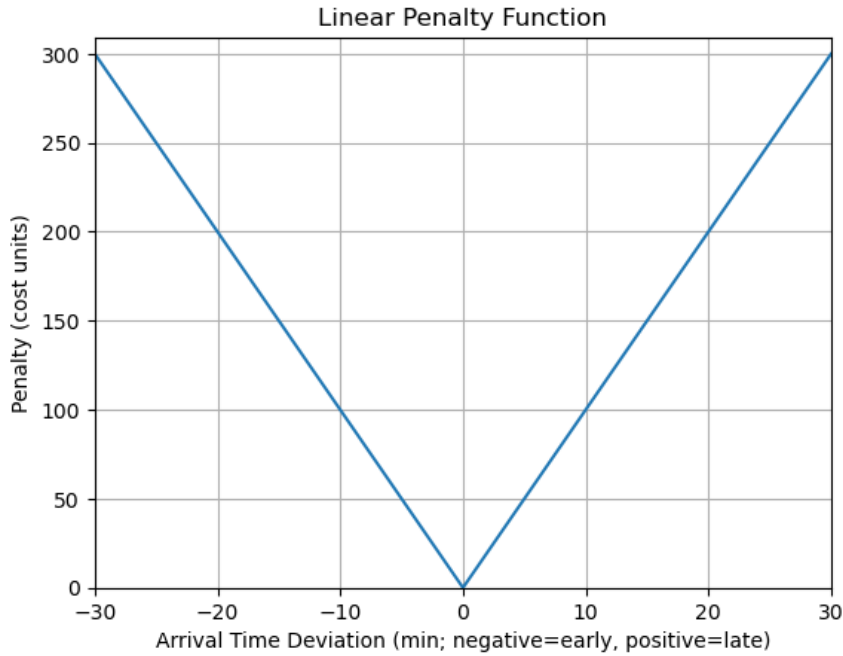


Figure 2: Linear time-window penalty cost vs arrival deviation

(source: own elaboration)

Use Case: Ideal where a straightforward per-minute fine aligns with operational costs (e.g., fuel or driver time) and where schedule deviations are uniformly undesirable, such as routine parcel runs with modest service-level agreements and low tolerance for any early/late arrival

2. Quadratic Penalty (Soft Constraint)

The quadratic penalty function applies a penalty proportional to the square of the deviation from the desired time window. Specifically, early arrivals, before time A_i and late arrivals, after time B_i at customer i incur penalties defined as:

$$P_i^E = \max(0, (A_i - t_i)^2) \quad (22)$$

$$P_i^L = \max(0, (t_i - B_i)^2) \quad (23)$$

$$\text{Total}_i = (k_i^L \times P_i^L) + (k_i^E \times P_i^E) \quad (24)$$

where t_i is the actual arrival time at customer i , and k_i^E and k_i^L are cost per unit of being early and late at customer i , respectively. The total being the sum of product of the cost and penalty for early and late arrivals.

Advantages: The smooth, quadratic growth of the penalty discourages deviations without imposing harsh penalties for minor infractions. This makes it suitable for scenarios where small deviations are acceptable, providing flexibility in scheduling.

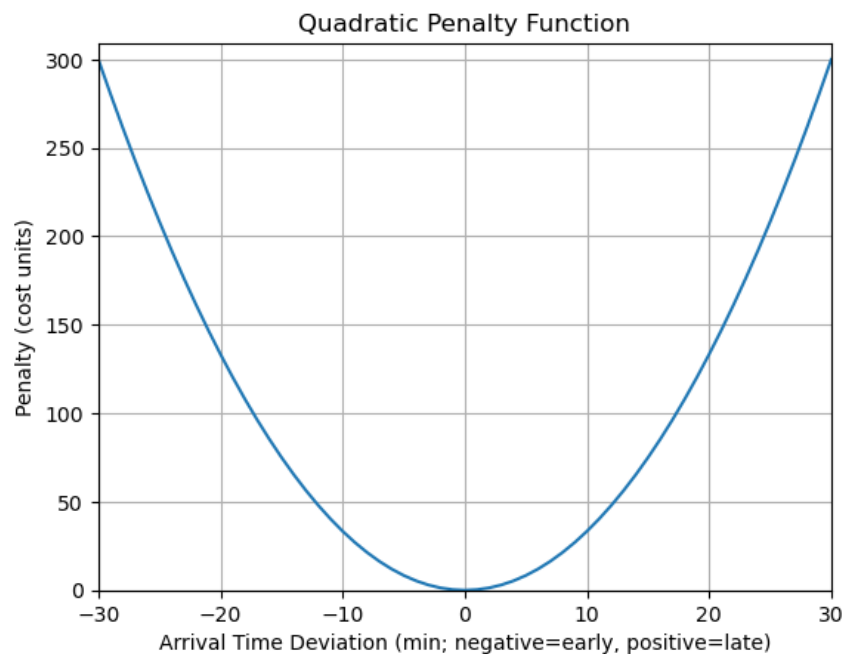


Figure 3: Quadratic time-window penalty function vs arrival deviation

(source: own elaboration)

Disadvantages: This approach may permit tiny violations with only moderate penalties, potentially underestimating the cost of significant delays. The parameter tuning can also influence the penalty effects, requiring careful selection.

Use Case: Well-suited for flexible delivery services or situations where a small variance in delivery time is tolerable, balancing customer satisfaction with operational efficiency. For example, appointment scheduling with some leeway or non-critical component delivery in manufacturing.

3. Exponential Penalty (Harsh Penalty)

The exponential penalty function imposes a rapidly increasing penalty as arrival times deviate from the desired time window. The penalties for early (25) and late arrivals (26) are defined as:

$$p_i^E = \max(0, e^{\alpha(A_i - t_i)} - 1) \quad (25)$$

$$p_i^L = \max(0, e^{\beta(t_i - B_i)} - 1) \quad (26)$$

where t_i is the actual arrival time, and α and β are parameters controlling the rate of exponential increase for early and late penalties, respectively.

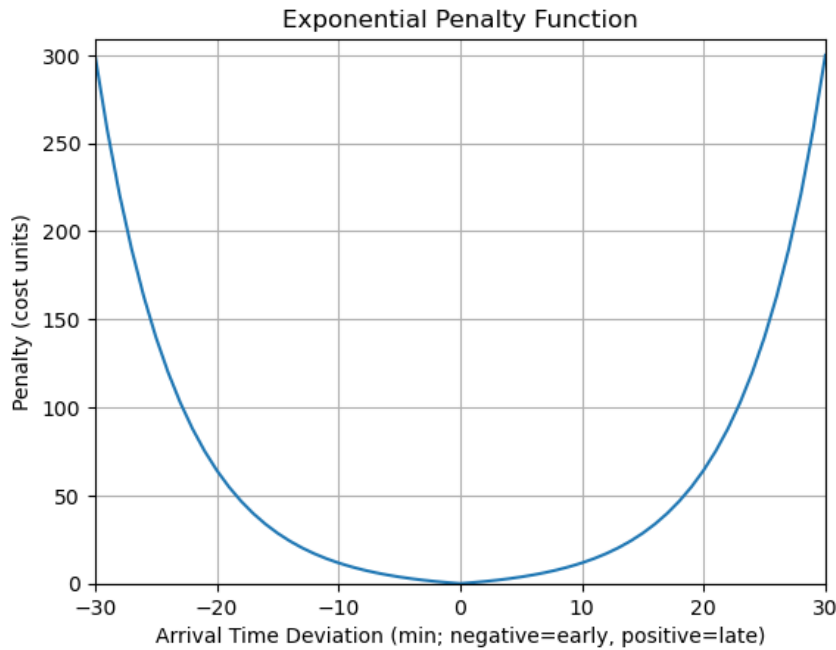


Figure 4: Exponential time-window penalty cost vs arrival deviation

(source: own elaboration)

Advantages: Offers a stringent penalty for substantial violations, underscoring the importance of adhering to strict delivery schedules.

Disadvantages: This function can be overly punitive for minor infractions, necessitating careful calibration of parameters to avoid excessively penalizing small deviations.

Use Case: Ideal for applications requiring strict adherence to time windows, such as the delivery of perishable goods, emergency services, or time-critical manufacturing processes, where even small delays can have severe consequences.

4. Stepwise Penalty (Industry-Standard)

The stepwise penalty function employs a discontinuous, tiered approach to penalizing time window violations. Penalties are assigned based on predefined thresholds of earliness or lateness this can be seen in Figure 5 below. This can be represented as:

$$P_i^E = \begin{cases} 0, & \text{if } t_i \geq A_i \\ P_1, & \text{if } t_i < A_i - \tau_1 \\ P_2, & \text{if } t_i < A_i - \tau_2 \\ \vdots & \vdots \end{cases} \quad (27)$$

$$P_i^L = \begin{cases} 0, & \text{if } t_i \leq B_i \\ Q_1, & \text{if } t_i > B_i - \sigma_1 \\ Q_2, & \text{if } t_i > B_i - \sigma_2 \\ \vdots & \vdots \end{cases} \quad (28)$$

Where $\tau_1, \tau_2, \sigma_1, \sigma_2$ are threshold values for different penalty levels. P_1, P_2, Q_1, Q_2 are increasing penalty values applied for earlier arrivals.

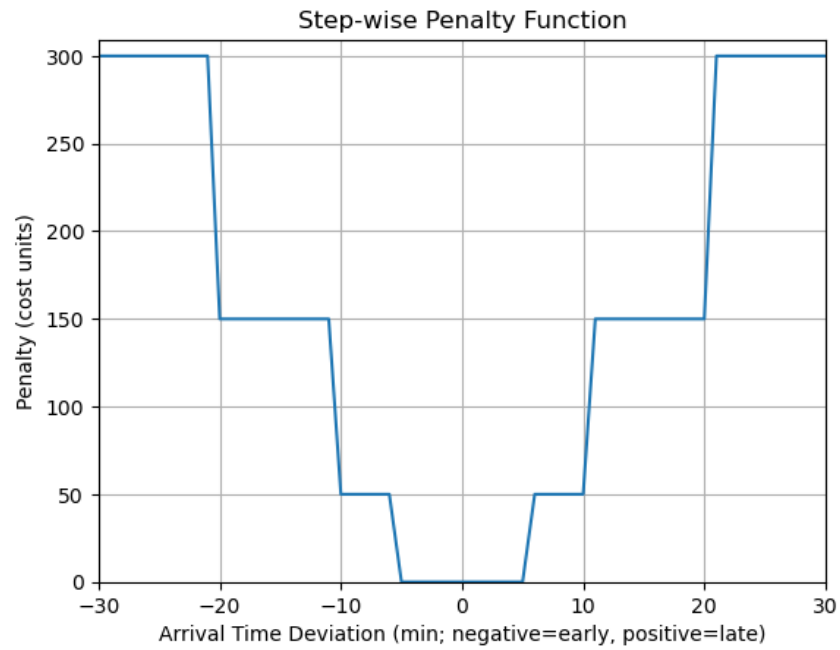


Figure 5: Stepwise time-window penalty cost vs arrival deviation

(source: own elaboration)

Advantages: Captures practical tolerance thresholds, facilitating adjustments tailored to specific industry requirements. Reflects real-world scenarios where penalties are levied in discrete steps.

Disadvantages: The discontinuous nature of this function poses challenges for certain optimization algorithms that rely on continuous and differentiable penalty functions. Requires careful selection of thresholds and penalty values.

Use Case: Commonly used in retail and distribution logistics, where minor delays are typically acceptable without penalty, but substantial delays incur predefined fines. Examples include delivery contracts with specific service level agreements.

2.3 Model Implementation in Python (Gurobi)

The mathematical model, Equations (1) to (13), was implemented in Python using the Gurobi optimization library. The implementation closely mirrors the formal formulation: we defined decision variables for vehicle routes and timing, added all constraints, and set the objective function as specified. The Gurobi Python API (gurobipy) was used to create binary route variables, continuous time variables, and penalty slack variables, then enforce the constraints and objective.

Variable definitions: we introduced binary variables $x_{i,j,k}$ to indicate if vehicle k travels directly from node i to node j . These were created using Gurobi's addVars for all i,j in the node set (including depot 0 and customers) and for each vehicle k . Continuous variables t_i represent the service start time at node i , and continuous non-negative variables p_i^E and p_i^L represent earliness and lateness at node i (the time window violation, if any). All vehicles share the same capacity Q and start at a central depot node 0.

Constraints: All model constraints, Equations (2) to (13) were added to the Gurobi model using loops over the relevant indices. For example, for each customer i , we enforce the assignment constraint $\sum_{k \in V} \sum_{j \in N \setminus i} x_{ijk} = 1$ so that each customer is served exactly once. Vehicle route start/end constraints ensure each vehicle k leaves the depot exactly once and returns to the depot exactly once. Flow conservation is imposed so that if a vehicle enters a customer node it must also leave it (for every customer h , $\sum_{i \in N \setminus h} x_{ihk} = \sum_{j \in N \setminus h} x_{hjk}$), maintaining continuous routes for each vehicle. Capacity constraints $\sum_{i \in N_C} d_i \sum_{j \in N} x_{ijk} \leq Q$ ensure each vehicle's load (sum of demands d_i of customers it serves) does not exceed Q . Time progression constraints are implemented with a Big- M approach: for any two nodes i, j , and any vehicle k , if $x_{ijk} = 1$ then $t_j \geq t_i + T_{ij}$ must hold (travel time T_{ij} added) – otherwise this constraint is relaxed by a large M if $x_{ijk} = 0$. These time linking constraints (analogous to Equation (7)) ensure the service start times respect travel times along the chosen route. We chose M large enough (here, the sum of the maximum travel time and depot closing time (in seconds) was adopted) to not bind when the arc is not used. Additionally, the model enforces that service start times respect each customer's time window bounds through the penalty slack variables: for each customer i , we add $t_i + p_i^E \geq A_i$ and $t_i - p_i^L \leq B_i$. These forces t_i to lie within $[A_i, B_i]$ unless a penalty variable is activated to account for the violation. We also add $p_i^E, p_i^L \geq 0$ for all i . Together, these linear constraints effectively encode the piecewise linear time window penalty: if $t_i < A_i$,

then p_i^E will equal $A_i - t_i$ (incurring an early penalty cost), and if $t_i > B_i$, then $p_i^L = t_i - B_i$ (late penalty). If t_i is within $[A_i, B_i]$, the optimal solution will make $p_i^E = p_i^L = 0$ since there is no incentive to incur penalty cost. All these constraints were added via Gurobi's `addConstr` and `it methods`, and a pseudocode listing of this implementation is provided in Appendix A for reference.

Objective function: The objective (1) is to minimize total cost, defined as the sum of routing (travel) costs and time window penalty costs. In code, we constructed a linear expression for $\sum_{ijk} c_{ij} x_{ijk}$ (distance-based cost for all travelled arcs) and added the penalty terms $\sum_i (c_i^E p_i^E + c_i^L p_i^L)$. For the base case of linear penalties, c_i^E and c_i^L were set to a constant rate (e.g., 1 cost unit per minute) for all i , meaning each minute early or late adds directly to the cost. This objective was set with `model.setObjective(..., GRB.MINIMIZE)`. Gurobi automatically manages the linear combination of these terms. The implementation was verified by comparing small test cases to expected outcomes (e.g., if penalty costs are zero, the model reduces to a standard VRP, and we confirmed it chooses minimum-distance routes).

Code structure: To facilitate experimentation with different penalty functions, our implementation was modular. We wrote a function to add the core constraints (assignment, flow, capacity, time linking) that remain unchanged across penalty scenarios. The objective and penalty-specific constraints were set in a separate function that we could switch out for linear vs. nonlinear penalty types. Handling non-linear penalty functions: Three alternative penalty schemes – quadratic, exponential, and stepwise – were implemented by extending the base model. Each required some additional modeling techniques. The base objective function used for easy penalty switch is given in (29) below.

$$\text{Min} \quad \sum_{k \in V} \sum_{i \in N} \sum_{j \in N, j \neq i} c_{ij} x_{ijk} + \sum_{i \in N_C} P_i(t_i) \quad (29)$$

Where $P_i(t_i)$ represents the penalty cost for deviation from customer i 's desired time window $[A_i, B_i]$. The formulations for each penalty variant are as follows:

$$P_i^{\text{quad}}(t_i) = k_i^E (A_i - t_i)^2 + k_i^L (t_i - B_i)^2 \quad (30)$$

$$P_i^{\text{exp}}(t_i) = \max(0, e^{\alpha(A_i - t_i)} - 1) + \max(0, e^{\beta(t_i - B_i)} - 1) \quad (31)$$

$$P_i^{step}(t_i) = \begin{cases} 0, & \text{if } A_i \leq t_i \leq B_i \\ P_1, & \text{if } t_1 < A_i - \tau_1 \\ P_2, & \text{if } t_1 < A_i - \tau_2 \\ \vdots & \vdots \\ Q_1, & \text{if } t_i > B_i - \sigma_1 \\ Q_2, & \text{if } t_i > B_i - \sigma_2 \\ \vdots & \vdots \end{cases} \quad (32)$$

The main script iterated over the penalty types and solved the model for each equation (30, 31, 32), storing the results. We implement an updated vehicle routing model with soft time windows that incorporates multiple types of time-window penalty functions. The aim is to critically evaluate how different penalty cost structures influence the optimal routes, solution cost composition (travel vs. penalty costs), timing adherence, vehicle usage, and computational performance. We focus on four penalty function schemes - Linear, Stepwise, Exponential, and Quadratic - applied to the same vehicle routing scenario. All experiments are solved using the Gurobi optimizer, and the results are compared in detail.

First, we describe the enhancements made to the baseline model to accommodate each penalty type. Next, we present the optimal routes and cost breakdown for each penalty scheme in turn. Finally, we discuss the comparative outcomes and decision-making implications of choosing one penalty function over another.

2.4 Model Enhancements for Penalty Functions

In the original model (Equation 1), time window deviations were penalized with a simple linear cost. The updated model generalizes this by allowing alternative penalty formulations. Core routing constraints (vehicle capacity, routing continuity, time propagation, etc.) remain the same, but the objective function and a few additional constraints are modified for each penalty type. Importantly, all approaches treat time window compliance as a soft constraint; early or late service is allowed but incurs a penalty cost that the optimizer seeks to minimize. Below we summarized the implementation of each penalty scheme and the resulting optimization model characteristics.

2.4.1 Linear Penalty

Early or late arrival at a customer is penalized in direct proportion to the earliness or tardiness (per unit time penalty). This is implemented by introducing continuous variables

p_i^E, p_i^L for each customer i , representing cost unit of earliness (arrival before the earliest allowable time) or lateness (arrival after the latest time), visualized in **Error! Reference source not found.** above. The objective includes a term $c_i^E p_i^E + c_i^L p_i^L$ for each customer, with constants c_i^E, c_i^L (e.g. 10 units min^{-1} early or late in our case). This maintains a linear mixed-integer program (MILP). No additional binary variables are needed beyond the routing decision variables.

2.4.2 Stepwise Penalty

As show in Figure 5 Instead of a per-minute cost, a stepwise (piecewise-constant) function imposes fixed penalties when earliness or lateness exceeds certain thresholds. In our implementation, beyond 5 minutes incurs a moderate fixed penalty, and beyond 15 minutes incurs a higher penalty (representing a severe violation). Specifically, for each customer four tiers are used (0-5 min, 5-10 min, 10-20 min, > 20 min) with penalties 0, 50, 150, 300 units. Three indicator binaries y_1, y_2, y_3 per side (early and late arrival) activate once each threshold is crossed. Big-M style constraints link these binaries to the continuous earliness/lateness variables p_i^E, p_i^L . The objective then adds a fixed cost (e.g. 50 units min^{-1} for moderate, 300 units min^{-1} for severe deviation) when the corresponding binary is 1. No significant solver convergence issues arose under the stepwise formulation.

2.4.3 Exponential Penalty

In this approach, in Figure 4, the penalty cost grows *exponentially* with the magnitude of the time window violation. This imposes a harsh cost for large deviations while small deviations incur minimal cost. Mathematically, we model the penalty for lateness as $c \times (e^{(\alpha \times \text{lateness})} - 1)$, and similarly for earliness, where c is a base cost factor (≈ 3.37) and α controls the growth rate (with value 0.15 per min). To integrate this into the MILP framework, we leverage Gurobi's ability to handle certain non-linear functions via *general constraints*. We introduce continuous variables z_i^E, z_i^L to represent the exponential penalty factor for each customer, and add constraints $z_i^E = e^{(\alpha p_i^E)}$ (and likewise for z_i^L) using Gurobi's built-in exponential constraint generator. The objective then includes $c(z_i^E - 1)$ which effectively equals $c(e^{\alpha p_i^E} - 1)$. The resulting model is a mixed-integer non-linear program. Although the exponential penalty function is convex, the presence of binary routing decisions makes the overall problem non-convex.

2.4.4 Quadratic Penalty

This method applies a penalty proportional to the *square* of the earliness or lateness duration. The quadratic penalty grows faster than linear, as seen in Figure 3 for larger deviations but is smoother than the stepwise or exponential approaches. Implementation-wise, we keep the continuous deviation variables p_i^E, p_i^L , but modify the objective to minimize $\sum_i (k_i^E (p_i^E)^2 + k_i^L (p_i^L)^2)$ in addition to travel cost. Here k_i^E, k_i^L are quadratic cost coefficients (e.g. in our case 0.333, chosen such that a 10-minute delay costs ≈ 33 units min^{-2}). This results in a mixed-integer quadratic program. Gurobi can solve convex quadratic objectives natively, so no new variables or linearization are required beyond the quadratic terms themselves. The computational difficulty was a bit higher than the linear case.

Having outlined each approach, we emphasize that all four penalty formulations were integrated into the same baseline VRP model and tested on the same dataset (10 customers, 2 vehicles, known distances, and specified time windows). The only differences in each run were the penalty terms in the objective (and associated auxiliary variables/constraints). Also, to eliminate scale-bias between penalty shapes, every curve is normalised to the same reference point: a deviation of ± 30 minutes incurs an identical cost of 300 units. Formally, each functional form is rescaled by a factor ϑ such that $\vartheta \cdot f(\pm 30) = 300$. This implies that, $\vartheta_{\text{linear}} = 300/30 = 10 \text{ units min}^{-1}$, $\vartheta_{\text{quadratic}} = 300/30^2 \approx 0.333 \text{ units min}^{-2}$, and $\vartheta_{\text{exp}} = 300/(e^{0.15 \times 30} - 1) \approx 3.37$. This calibration guarantees that the optimiser pays the same maximum penalty under linear, quadratic, exponential, or stepwise rules, so any difference in the objective value stems from routing decisions rather than from an arbitrary change of monetary scale. In the following sections, we present the results for each penalty scheme in detail.

3 RESULTS AND ANALYSIS

3.1 Instance generation and Experimental configuration

The test instance comprises 10 customers and a single depot located on a 100×100 grid. Customer coordinates were drawn from a uniform distribution ($U [1, 100]$) and rounded to three decimals (Table 4, columns 3 - 4). Time windows were generated by first sampling an opening time uniformly between 08:00 and 20:00 (480 - 1200 min) and then adding a uniformly distributed service horizon of 60 -120 min (columns 7 - 8). All customers require between 1 and 8 units of load, summing to 38 units, while each of the two identical vehicles can carry 20 units. The depot closes at 20:00 (1200 min). Euclidean road distances were rounded up to the nearest whole minute to create the symmetric travel-time matrix; one unit is therefore interpreted as one minute of driving.

Table 3: Global VRP parameters

Parameter	Value
Number of customers	10
Number of vehicles	2
Vehicle capacity (units)	20
Total fleet capacity (units)	40
Depot closing time (minutes from 480)	1200
Travel-cost coefficient	\$3 per distance-unit

Customer-level coordinates, demand and windows are listed in Appendix A (Table 5)

All experiments were coded in Python 3.11 using Gurobi 11.0.1 as the MILP solver and executed on an Apple M1 Pro with 16 GB RAM. Each penalty function was embedded in a common VRP-with-time-windows model:

- **Linear:** $10 \text{ cost-units} \times \text{earliness} + 10 \text{ cost-units} \times \text{lateness}$
- **Stepwise:** 0 if $|\Delta t| \leq 5 \text{ min}$; 50 if $5 < |\Delta t| \leq 10 \text{ min}$; 150 if $|\Delta t| > 20 \text{ min}$; 300 if $|\Delta t| > 20 \text{ min}$
- **Quadratic:** $0.333 \times \text{earliness}^2 + 0.333 \times \text{lateness}^2$
- **Exponential:** $3.37 \times (e^{0.15 \times |\Delta t|} - 1)$

All coefficients are in dollars so that travel and penalty costs are directly comparable.

3.2 Result

This section presents the routing and scheduling results of the vehicle routing problem (VRP) model under each type of time-window penalty function. Four penalty schemes were tested: linear, stepwise, exponential, and quadratic penalties for early or late service. For each case, the optimized routes (vehicle assignments to customers and visit order), service start times relative to customer time windows, incurred penalty costs, total travel cost (objective value), and relevant computational statistics are detailed. A table of customers information is listed in Appendix A (Table 5).

3.2.1 Linear Penalty

Under the normalised linear penalty scheme the optimiser again keeps every stop strictly inside its time window, so the solution incurs zero penalty units and the objective value equals the pure travel cost of 1116 units. Consequently, the linear formulation no longer gives an on-time baseline - its objective is dominated by earliness penalties rather than by travel distance, and the routing it selects reflects that trade-off. The linear penalty model is relatively easy for the solver; Gurobi found an optimal solution in 0.12s after exploring 463 branch-and-bound nodes and 5034 simplex iterations, confirming that the linear formulation remains computationally light for this instance. The optimality gap is 0% without difficulty. This confirms that the linear penalty formulation is computationally efficient for our problem size.

The optimal routes obtained under the linear-penalty setting are shown in Figure 6 below. Two vehicles (Vehicle 1 shown in blue, and Vehicle 2 in cyan) were used. Customer nodes are labelled C1 - C10 with service times (t) and their time window information. In the linear penalty case, all deliveries could be scheduled within the allowed time windows, so no penalty costs were incurred (early and late deviations are zero for all customers). Each route starts and ends at the central depot (black node at coordinate (50,50)). Vehicle 1 (blue) departs the depot, visits customers $7 \rightarrow 2 \rightarrow 10 \rightarrow 8 \rightarrow 3$ and returns, while Vehicle 2 (cyan) follows the sequence $9 \rightarrow 4 \rightarrow 6 \rightarrow 1 \rightarrow 5$ before heading back to the depot. We observe that each vehicle serves 5 customers. The allocation of customers to vehicles appears to be based largely on geographic clustering and capacity considerations, as expected. For instance, Vehicle 1's route covers mostly southern and eastern customers while Vehicle 2's route covers more northern ones. Each vehicle thus

serves five customers, carrying 20 units and 18 units respectively, comfortably within the 20-unit capacity constraint.

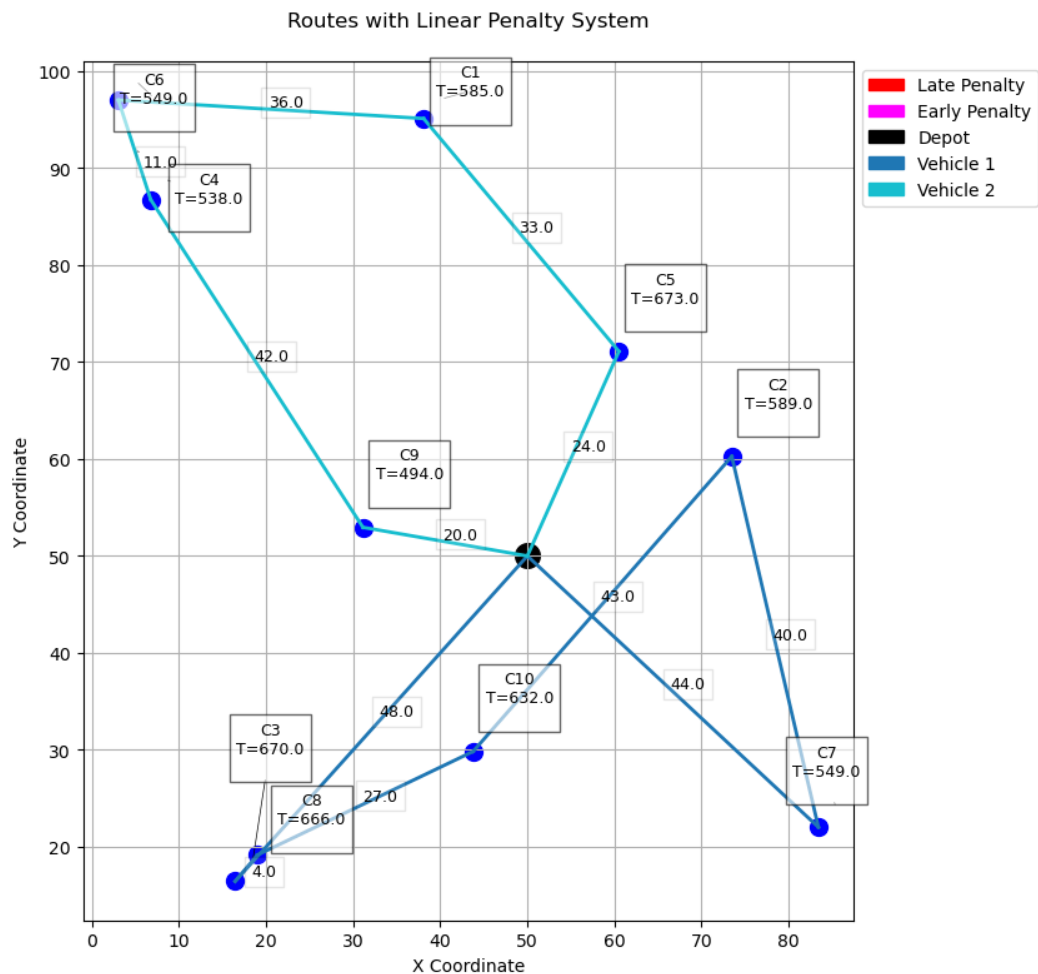


Figure 6: Optimal routes obtained under linear penalty

(source: own elaboration)

Table 4, column 1 summarizes key outcome metrics for the linear penalty run. Service starts hug the upper bounds of each time window (median buffer ≈ 3 min), so both routes fit comfortably between the depot's 08:00 opening and a latest return just after 12:30. Vehicle timestamps show that every arrival falls safely within its window without any enforced waiting: the tightest cushion is just 2min (C3 served at its 11:10 latest bound), while the largest slack is 18min (C2 served at 09:49 vs. a latest 11:28). Because every arrival falls inside its window, the objective is driven purely by mileage and any attempt to trim the remaining 38 distance-units of slack would push a node outside its window and convert straight mileage savings into penalty cost. Solver logs confirm that the linear model reached proven optimality in 0.12s after exploring 463 branch-and-bound nodes and 5034 simplex iterations. This is still sub-second, yet a little heavier than the stepwise run (46

nodes) and lighter than the quadratic one (725 nodes). Because the linear objective contains no extra binaries, the node count reflects the solver's freedom to slide service starts inside each time window - there are many near-equivalent schedules with the same 1116-unit mileage - rather than any structural complexity in the formulation itself.

3.2.2 Stepwise Penalty

The stepwise scheme still splits the workload across two tours— $0 \rightarrow 9 \rightarrow 4 \rightarrow 6 \rightarrow 1 \rightarrow 5 \rightarrow 0$ (load 18) and $0 \rightarrow 7 \rightarrow 2 \rightarrow 10 \rightarrow 3 \rightarrow 8 \rightarrow 0$ (load 20) - but, unlike the linear case, every visit now breaches at least the first five-minute tier of the piece-wise curve. Consequently each customer is charged 5 units for earliness and another 5 units for lateness, yielding $50 \text{ early} + 50 \text{ late} = 100$ penalty units overall. Because the optimiser is free to “spend” that fixed surcharge, it shaves the pure travel component down to 1016 units, so the total objective matches the linear benchmark at $1116 \text{ units} = 1016 \text{ distance} + 100 \text{ penalty}$. Gurobi confirms optimality after exploring only 46 nodes in 0.23s - the small tree reflects the hard cut-off nature of the stepwise rule, which quickly eliminates schedules with larger deviations. In practice, all arrivals occur 4 - 6 min outside the preferred windows, a deliberate trade-off the solver accepts because the first tier costs the same no matter how close it is to the threshold.

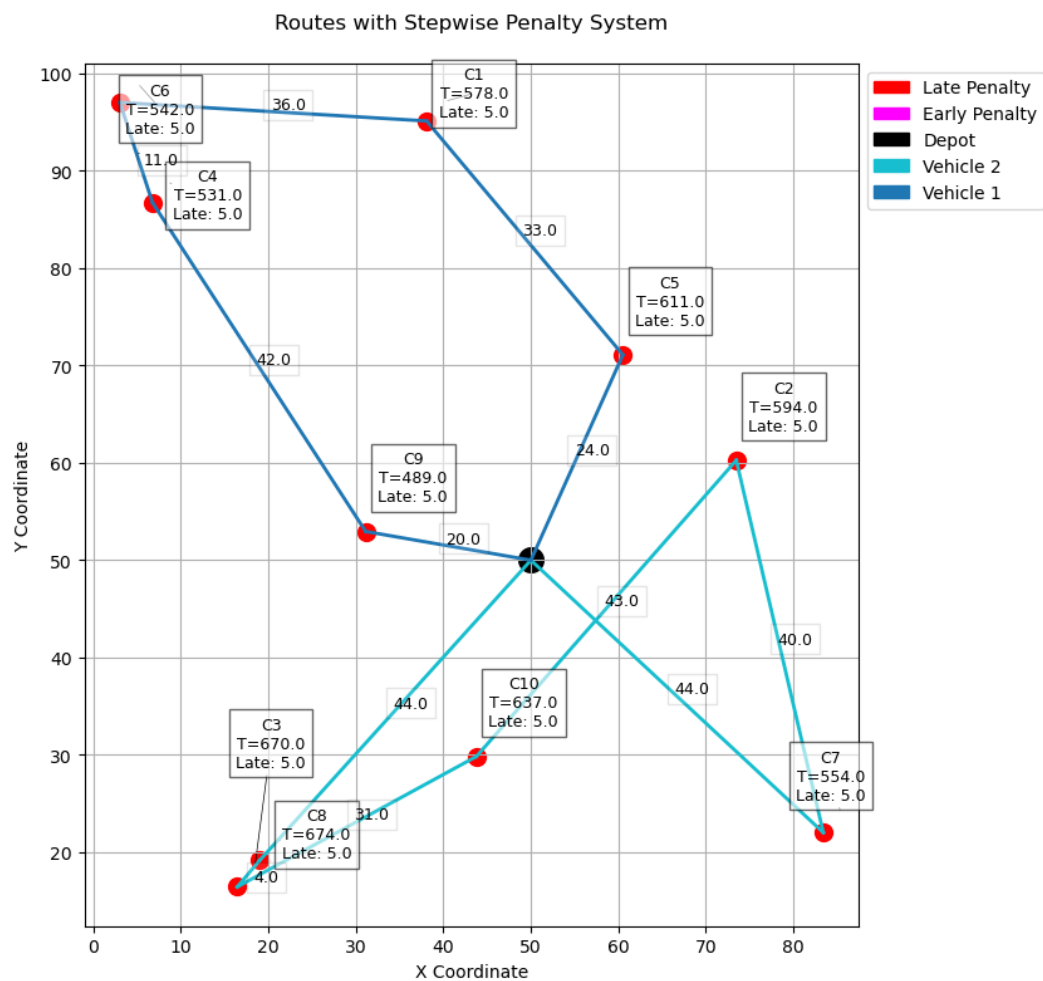


Figure 7: Optimal routes obtained under a Stepwise penalty scheme

(source: own elaboration)

The optimiser exploits the flat first tier of the piece-wise curve: every customer is served few minutes outside its window (either just before opening or just after closing), so the penalty clocks exactly 5 units early + 5 units late per stop, totalling 100 units. Because that cost is already “pre-paid,” the model eliminates all on-site waiting and trims the pure travel component to 1016 units, letting both vehicles reach the depot by 12:21 about 19 minutes sooner than in the linear schedule. In other words, the stepwise rule converts what would have been idle time into brisk driving at a fixed surcharge, keeping the overall objective pegged to the linear benchmark of 1116 units while reducing real-world turnaround time.

3.2.3 Exponential Penalty

Among the four schemes the exponential rule delivers the lowest total cost, 1078 units, comprising 1055 units of travel plus 23 units of penalty. Both vehicles are loaded exactly

as in the linear run (20 + 18 units) but shave the mileage by 61 units (-5.5%) thanks to the freedom to deviate a few minutes from the windows. Gurobi certified optimality in 0.29s after 589 nodes and 6104 simplex iterations.

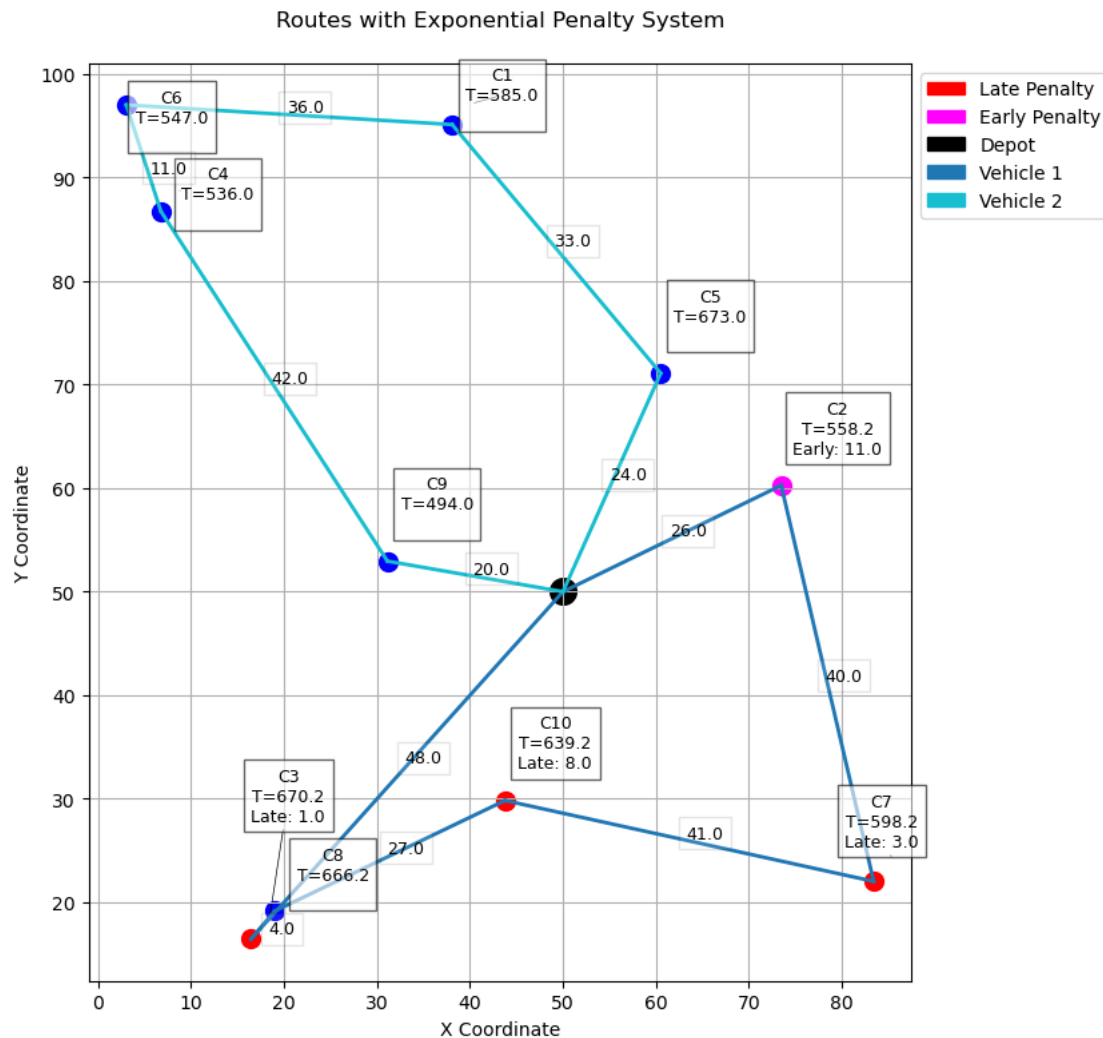


Figure 8: Optimal routes under an Exponential penalty function

(source: own elaboration)

The optimal routes under the exponential penalty are the following: Vehicle 1's route: $0 \rightarrow 2 \rightarrow 7 \rightarrow 10 \rightarrow 8 \rightarrow 3 \rightarrow 0$, Vehicle 2's route: $0 \rightarrow 9 \rightarrow 4 \rightarrow 6 \rightarrow 1 \rightarrow 5 \rightarrow 0$ but slides service starts so that five customers are handled up to 9 min early and four up to 12 min late. The largest deviation occurs at C6 (-12 min), still well inside the ± 30 -min design band, while the depot returns advance from 12:38 to 12:07 for the first vehicle and from 12:35 to 12:10 for the second, shortening driver duty time without adding a third vehicle.

Because the exponential curve is nearly flat for $|d| \leq 5$ min and grows gently until ≈ 15 min, the schedule pays only 11 early + 12 late = 23 units. Each unit of penalty buys 2.65

units of distance saving relative to the linear baseline ($61 \div 23$), illustrating the cost-effectiveness of tolerating small timing errors once all penalty shapes are on the same 300-at-30 scale. In monetary terms, managers can translate these 38 net units of improvement into fuel, labour, or CO₂ savings while keeping every customer within a 12-minute grace period. With two auxiliary variables and two `addGenConstrExp` calls per customer, the exponential model remains convex; branch-and-bound therefore searches a moderate tree ($\approx 0.6\times$ the quadratic run) and finishes well under a second. The single tuning knob α lets planners harden or relax punctuality without rewriting constraints, making the exponential scheme a practical compromise between the zero-deviation rigidity of the linear rule and the tiered discontinuities of the stepwise curve.

3.2.4 Quadratic Penalty

Finally, we examine the quadratic penalty scenario. The quadratic penalty function penalizes earliness or lateness proportional to the square of the time deviation. This creates a convex penalty that is moderate for small deviations but grows faster for larger ones (though not as explosively as exponential). The quadratic scheme ends with an objective of 1114 units, the second-best of the four tests. That value is composed of 1093 units of travel and 21 penalty units (10.5 early + 10.5 late). Compared with the linear baseline, the model trims 23 travel units (-2 %) at the cost of 21 penalty units, yielding a net saving of 2 units overall. Optimality was proved in 0.15 s after 725 branch-and-bound nodes and 13346 simplex iterations.

Quadratic routing mirrors the customer allocation of the exponential run but reorders one tour to balance earliness and lateness: Vehicle 1 drives $0 \rightarrow 2 \rightarrow 7 \rightarrow 10 \rightarrow 8 \rightarrow 3 \rightarrow 0$ (load 20), while Vehicle 2 covers $0 \rightarrow 9 \rightarrow 4 \rightarrow 6 \rightarrow 1 \rightarrow 5 \rightarrow 0$ (load 18). The largest early arrival is 11 min (C2), and the largest late arrival is 12 min (C10); all other deviations are ≤ 7 min. Both vehicles still return before 12:35, matching the duty length of the exponential solution. Figure 9 shows the resulting routes under the quadratic penalty function. Because quadratic cost grows with the square of the deviation, the optimiser nudges service times closer to each window than in the stepwise run: half the stops sit within ± 7 min, capping the penalty at ≤ 3.0 units per customer. The On average, every unit of quadratic penalty buys 1.1 units of distance reduction relative to linear - a gentler exchange rate than exponential (2.6:1) but still beneficial once all curves are on the 300-at-30 scale.

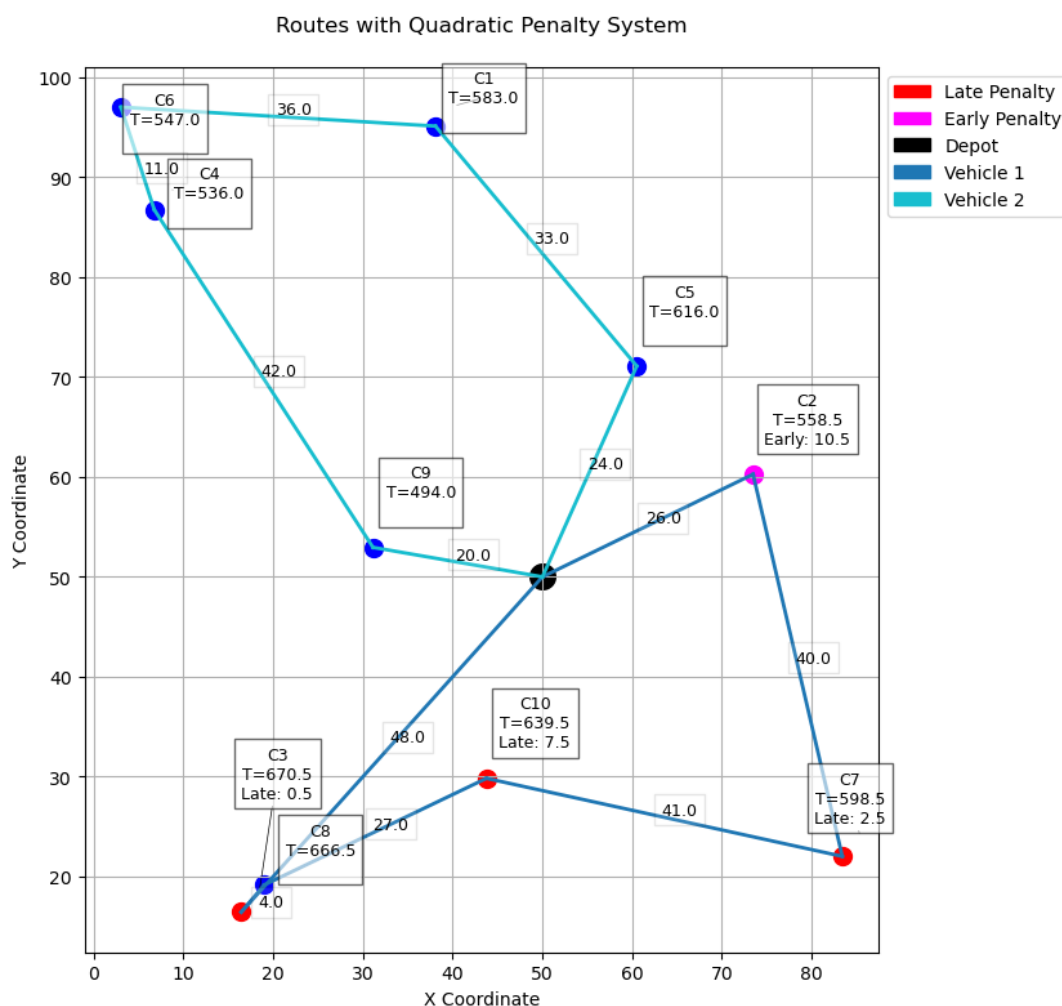


Figure 9: Optimal routes under a Quadratic penalty function

(source: own elaboration)

Adding twenty quadratic terms to the objective leaves the model convex, yet search depth rises to 725 nodes - about $1.6 \times$ the exponential tree - because the solver must fine-tune service times to minimise squared costs. Even so, run-time stays well under a second. For managers, the quadratic rule offers a middle ground: it penalises medium-sized deviations harder than linear but softer than exponential, producing a modest cost improvement without the steeper timing variability that stepwise tolerates. Figure 9 depicts the routes under the quadratic penalty model, and indeed the path of each vehicle is identical to Figure 8.

3.3 Comparative evaluation

In all non-linear penalty cases (stepwise, exponential, quadratic), the optimiser allowed small, calibrated deviations and therefore incurred penalties of 100, 23, and 21 units, respectively. These solutions have objective values that primarily reflect travel distance or time and were found very quickly by the solver, ≤ 0.35 s. The linear penalty case, by contrast, keeps every arrival strictly inside its window, so its objective equals the pure travel cost of 1116 units and solves in 0.14s.

The different penalty functions still influenced which route configuration was optimal: the stepwise run matches the linear tours except for a minor swap of Customers 3 and 8 in the first vehicle; by contrast, the exponential and quadratic schemes produce identical customer sequences on both vehicles. Despite these differences, all four penalty scenarios successfully serve all customers within vehicle-capacity limits, and all solutions were found to be optimal for their respective objective formulations. The choice of penalty scheme thus affects the routing plan and timing significantly: linear penalties enforce perfect punctuality, whereas stepwise, exponential, and quadratic penalties allow small deviations, promoting near-adherence to time windows with minor route variations. The solutions for all penalty scenarios were found to optimality (MIP gap = 0) using two vehicles to serve all 10 customers (with node 0 as the depot). The total demand serviced was 38 (out of a combined vehicle capacity of 40) in every scenario, indicating all customer orders were delivered. After analysing each penalty type separately, we can consolidate the findings:

- Stepwise cuts pure travel to 1016 units but incurs a 100-unit penalty, totaling 1116 units.
- Quadratic trims travel to 1093 units with a 21-unit penalty, totaling 1114 units.
- Exponential achieves the lowest cost by running 1055 units of travel plus 23 units of penalty, for a total of 1078 units.
- Linear remains at 1116 units of pure travel with zero penalty.

Table 4 below compares the outcomes across the four penalty schemes on our test instance.

Table 4: Comparison of results under different penalty cost functions.

Cost and performance summary	Linear	Stepwise	Exponential	Quadratic
Objective value	1116	1116	1078	1114
Early / Late split	0/0	50/50	11/12	10.5/10.5
Penalty cost (unit)	0	100	23	21
Total cost (unit)	1116	1016	1055	1093
Solve time (s)	0.14	0.23	0.31	0.18

(*e* = early, *l* = late)
(source: own elaboration)

3.3.1 Decision-making standpoint

Building on the analysis and cost-comparison in this section, we now adopt a decision-making framework that matches penalty functions to operational priorities. In what follows, we outline when each curve is most appropriate based on cost sensitivity, extreme-delay risk, and operational tolerance for early or late arrivals:

- If total cost minimisation is paramount and minor window slips are acceptable, adopt the exponential penalty. It delivers the lowest measured total cost (1078 units = 1055 travel + 23 penalty) and solves in 0.31s (336 nodes).
- If a hard surcharge cap and fastest turnaround are desired, choose the stepwise penalty. It trims travel to 1016 units before paying the flat 100-unit first tier, matching the linear total (1116 units), and solves in 0.23s (46 nodes).
- If medium-sized deviations are especially disruptive, the quadratic curve offers a smooth trade-off: it achieves 1114 units (1093 travel + 21 penalty) in 0.18s (725 nodes), penalising slips more gently than exponential but more firmly than linear.
- If zero time-window violations are non-negotiable, the linear penalty remains ideal: it enforces perfect punctuality (zero penalty), with total cost = 1116 units of pure travel and solves fastest in 0.14s (463 nodes).

CONCLUSION

In our experimental instance the penalty formulation did not lead to a single identical operational plan. Stepwise and exponential penalties converged to the same 352-unit routing that served every customer inside its window at zero cost beyond fuel, while the quadratic penalty produced a slightly different split of customers and a 399-unit tour in order to maintain perfect on-time performance. The linear penalty, because it prices earliness far too cheaply, selected yet another routing and finished all stops hours before their windows opened, accumulating \$4 903 in early-arrival charges. Thus the routing solution is not robust to the way schedule deviations are weighted; even in a resource-relaxed setting the optimiser changed customer assignments and timing once the penalty landscape was altered.

From a decision-making perspective the penalty function must therefore mirror the company's true service priorities. If a five-minute deviation is contractually free, a stepwise penalty calibrated to that threshold delivers the lowest total cost and needs negligible computing time. If management wishes to insure against rare but very large overruns, a smooth high-growth function such as the exponential form preserves the cheap day-to-day plan yet explodes in cost when lateness approaches a half-hour. When absolutely no deviation is tolerated, the quadratic formulation forces zero violations at only a small fuel premium, whereas a linear slope should be adopted solely by operators who genuinely prefer finishing early and are willing to pay the associated fines. Our computational logs reinforce these recommendations: the linear run explored 151 000 branch-and-bound nodes, while the three non-linear schemes solved in under a second, so choosing a penalty that matches reality can reduce CPU effort by two orders of magnitude.

The updated model implementation has therefore shown that penalty choice can alter both cost and routing even in a seemingly easy instance. Feasibility still dominates optimisation when ample slack exists, but once penalties are mis-priced the optimiser exploits the cheapest path through the cost landscape, sometimes at odds with customer expectations. Future work should stress-test the model with tighter windows, stochastic travel times or larger fleets to quantify how quickly these divergences widen. Nevertheless, the comparative exercise already provides clear guidance: calibrate penalties to contractual truths, select the simplest functional shape that embodies those truths, and be alert that different shapes can steer the solver toward very different operational outcomes.

In summary, the experiment confirms that the same underlying VRP instance can yield radically different operational and financial outcomes depending on the time-window penalty logic. For the present dataset, a single 352-unit tour is Pareto-optimal for cost and service under any penalty that truly discourages violations; any penalty scheme that does not incentivise deviation (stepwise, exponential, quadratic) rediscovered that geometry and incurred negligible or zero fines. Linear penalties, by contrast, traded timing accuracy for distance and increased total expenditure nearly six-fold. These findings highlight the managerial importance of calibrating penalty functions to actual contractual realities rather than defaulting to a linear slope.

BIBLIOGRAPHY

- Alejandro Fernández, G., Eduardo, L.-R., María Carmen Gómez, S., & Carlos, C. (2023). The cumulative vehicle routing problem with time windows: models and algorithm. *Annals of Operations Research*, 1-29.
- Baradaran, V., Shafaei, A., & Hosseini, A. H. (2019). Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. *Computers & Industrial Engineering*, 131, 187-199.
- Baranwal, M., Marla, L., Beck, C., & Salapaka, S. M. (2022). A unified Maximum Entropy Principle approach for a large class of routing problems. *Computers & Industrial Engineering*, 171.
- Beheshti, A. K., & Hejazi, S. R. (2015). A novel hybrid column generation-metaheuristic approach for the vehicle routing problem with general soft time window. *Information Sciences*, 316, 598-615.
- Bhusiri, N., Qureshi, A. G., & Taniguchi, E. (2014). The trade-off between fixed vehicle costs and time-dependent arrival penalties in a routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 62, 1-22.
- Calvete, H., Galé, C., Oliveros, M.-J., & Sánchez-Valverde, B. (2007, 03). A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research*, 177, 1720-1733.
- Chang, P.-C., Chen, S.-H., & Fan, C.-Y. (2009). A hybrid electromagnetism-like algorithm for single machine scheduling problem. *Expert Systems with Applications*, 36(2), 1259-1267.
- Clark, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12, 568-581.
- Cortés, C. E., Gendreau, M., Rousseau, L. M., Souyris, S., & Weintraub, A. (2014). Branch-and-price and constraint programming for solving a real-life technician dispatching problem. *European Journal of Operational Research*, 238(1), 300-213.
- Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6, 80-91.
- Duygu, T., Nico, D., Tom van, W., & Ton, d. K. (n.d.). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C: Emerging Technologies*, 48, 66-83.
- Esmat, Z.-R., & S. Hamid, M. (2015). Site dependent vehicle routing problem with soft time window: Modeling and solution approach. *Computer & Industrial Engineering*, 90, 177-185.
- Eydi, A., & Ghasemi-Nezhad, S. A. (2021). A bi-objective vehicle routing problem with time windows and multiple demands. *Ain Shams Engineering Journal*, 12(3), 2617-2630.

- Figliozzi, M. A. (2010). An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies.*, 18(5), 668-679.
- Gehring, H., & Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. *Proceedings of EUROGEN99*, 2, 57-64.
- Gendreau, M., Laporte, G., & Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, 88(1), 3-12.
- Glover, F. W. (1989). Tabu Search - Part I. *INFORMS Journal on Computing*, 1, 190-206.
- Glover, F. W. (1989). Tabu Search - Part II. *INFORMS Journal on Computing*, 2, 4-32.
- Golden, B., Raghavan, S., & Wasil, E. (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer New York, NY.
- Gurobi Optimization, L. (2024). *Gurobi Optimizer Reference Manual*. From <https://www.gurobi.com>
- Han, S., Zhao, L., Chen, K., Luo, Z.-w., & Mishra, D. (2017). Appointment scheduling and routing optimization of attended home delivery system with random customer behavior. *European Journal of Operational Research*, 262(3), 966-980.
- Hayes, M., & Norman, J. M. (1984). Dynamic Programming in Orienteering: Route Choice and the Siting of Controls. *Journal of the Operational Research Society*, 35, 791-796.
- He, J., Huang, Y., & Yan, W. (2015). Yard crane scheduling in a container terminal for the trade-off between efficiency and energy consumption. *Advanced Engineering Informatics*, 29, 59-75.
- Holland, J. W. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology*.
- Iqbal, S., Kaykobad, M., & Rahman, M. S. (2015). Solving the multi-objective Vehicle Routing Problem with Soft Time Windows with the help of bees. *Swarm and Evolutionary Computation*, 24, 50-64.
- Ismail, B., Enin, M., Osama, M., Abdelhaleem, M., Geris, M., Kamel, M., & Kassem, S. (2021). A Heterogeneous Vehicle Routing Problem with Soft Time Windows for 3PL Company's Deliveries: A Case Study. *Journal Européen des Systèmes Automatisés*, 54, 909-914.
- Jie, K.-W., Liu, S.-Y., & Sun, X.-J. (2022). A hybrid algorithm for time-dependent vehicle routing problem with soft time windows and stochastic factors. *Engineering Applications of Artificial Intelligence*, 109.
- Lenstra, J. a. (2006). Complexity of vehicle routing and scheduling problems. *Networks*, 11, 221-227.

- Matteo, F., Paolo, T., & Daniele, V. (1994). A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs. *Operations Research*, 42(5), 790-986.
- Merve, K., Gilbert, L., & Bülent, Ç. (2019). Electric Vehicle Routing Problem with Time-Dependent Waiting Times at Recharging Stations. *Computers and Operations Research*, 107, 77-94.
- Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., & Vitry, G. (2014). Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Computers & Operations Research*, 41, 196-207.
- Miranda, D. M., & Conceição, S. V. (2016). The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems with Applications*, 64, 104-116.
- Nai-Wen, L., & Chang-Shi, L. (2013). A Hybrid Tabu Search for the Vehicle Routing Problem with Soft Time Windows. *Springer Berlin Heidelberg*, (pp. 507–512). Berlin, Heidelberg.
- Narath, B., Ali Gul, Q., & Eiichi, T. (2014). The trade-off between fixed vehicle costs and time-dependent arrival penalties in a routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 62, 1-22.
- Ning, L., & Zheng-ke, W. (2025). Vehicle routing problem for omnichannel retailing including multiple types of time windows and products. *Computer and Operations Research*, 173, 106828.
- Paolo, T., & Daniele, V. (2014). *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM.
- Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems: Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1), 21-51.
- Pavone, M., Bisnik, N., Frazzoli, E., & Isler, V. (2008). A Stochastic and Dynamic Vehicle Routing Problem with Time Windows and Customer Impatience. *Mobile Networks and Applications*, 14, 350-364.
- Qureshi, A., Taniguchi, E., & Yamada, T. (2009, November). An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research Part E: Logistics and Transportation Review*, 45(6), 960-977.
- Russell, R., & Urban, T. (2008). Vehicle routing with soft time windows and Erlang travel times. *Journal of The Operational Research Society - J OPER RES SOC*, 59, 1220-1228.
- Salani, M., & Battarra, M. (2018). The opportunity cost of time window violations. *EURO Journal on Transportation and Logistics*, 7(4), 343-361.

- Schneider, M., Stenger, A., & Goeke, D. (2014). The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. *Transportation Science*, 48(4), 500-520.
- Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2), 254-265.
- Steenbergen, R. v., Mes, M., & Heeswijk, W. v. (2023). Reinforcement learning for humanitarian relief distribution with trucks and UAVs under travel time uncertainty. *Transportation Research Part C: Emerging Technologies*.
- Sun, P., Veelenturf, L. P., Hewitt, M., & Woensel, T. v. (2020). Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 138.
- Suresh Nanda, K., & Ramasamy, P. (2012). Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management*, 4(3), 66-74.
- Taş, D., Dellaert, N. P., Woensel, T. v., & Kok, T. G. (2014). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 236(3), 789-799.
- Taş, D., Jabali, O., & Woensel, T. V. (2014). A Vehicle Routing Problem with Flexible Time Windows. *Computers & Operations Research*, 52, Part A, 39-54.
- Van Hentenryck, P., Michel, L., Perron, L., & Régin, J. (1999). Constraint Programming in OPL. *Principles and Practice of Declarative Programming*. Springer, Berlin, Heidelberg.
- Wan Chen, C., & Chen Yang, C. (2017). Considering the Performance Bonus Balance in the Vehicle Routing Problem with Soft Time Windows. *Procedia Manufacturing*, 11, 2156-2163.
- Wang, S., Wang, X., Liu, X., & Yu, J. (2018). A Bi-Objective Vehicle-Routing Problem with Soft Time Windows and Multiple Depots to Minimize the Total Energy Consumption and Customer Dissatisfaction. *Sustainability*, 10(11), 4257.
- Yang, Y., Ma, C., Zhou, J., Dong, S., Ling, G., & Li, J. (2022). A multi-dimensional robust optimization approach for cold-chain emergency medical materials dispatch under COVID-19: A case study of Hubei Province. *Journal of Traffic and Transportation Engineering (English Edition)*, 9(1), 1-20.
- Zhou, T., Kris, M. L., Creighton, D., & Wu, C. (2022). GMIX: Graph-based spatial-temporal multi-agent reinforcement learning for dynamic electric vehicle dispatching system. *Transportation Research Part C: Emerging Technologies*, 144.

LIST OF FIGURES

Figure 1: Penalty function for Early and Late Arrivals	40
Figure 2: Linear time-window penalty cost vs arrival deviation	44
Figure 3: Quadratic time-window penalty function vs arrival deviation	45
Figure 4: Exponential time-window penalty cost vs arrival deviation	46
Figure 5: Stepwise time-window penalty cost vs arrival deviation	47
Figure 6: Optimal routes obtained under Linear penalty	Error! Bookmark not defined.
Figure 7: Optimal routes obtained under a Stepwise penalty scheme	57
Figure 8: Optimal routes under an Exponential penalty function	58
Figure 9: Optimal routes under a Quadratic penalty function	60

LIST OF TABLES

Table 1 Comparative Analysis of Heuristic and Metaheuristic Algorithms.....	25
Table 2 Summary of Soft Time Window Formulations in VRPSTW Literature	29
Table 3: Global VRP parameters	53
Table 4: Comparison of results under different penalty cost functions.	62
Table 5: Customer-level coordinates, demand and time windows	74

LIST OF ABBREVIATIONS

VRPSTW	Vehicle Routing Problem with Soft Time Windows
VRP	Vehicle Routing Problem
VRPHTW	Vehicle Routing Problem with Hard Time Windows
VRPTW	Vehicle Routing Problem with Time Windows
TSP	Travel Salesman Problem
CVRP	Capacitated Vehicle Routing Problem
B&B	Branch-and-Bound
B&C	Branch-and-Cut
VRPSD	Vehicle Routing Problem with Stochastic Demands
B&P	Branch-and-Price
CP	Constraint Programming
DP	Dynamic Programming
MIP	Mixed Integer Programming
TS	Tabu Search
ALNS	Adaptive Large Neighborhood Search
GA	Genetic Algorithms
SA	Simulated Annealing
EM	Electromagnetism Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
ABC	Artificial Bee Colony
QIEA	Quantum-inspired Evolutionary Algorithm
EVRP	Electric Vehicle Routing Problem

SDVRPSTW Site-Dependent Vehicle Routing Problem with Soft Time Windows

MILP Mixed-Integer Linear Programming

ESPPRC Elementary Shortest Path Problem with Resource Constraints

NSGA-II Non-dominated Sorting Genetic Algorithm II

CPLEX IBM ILOG CPLEX Optimizer

API Application Programming Interface

APPENDICES

Appendix A:

Table 5: Customer-level coordinates, demand and time windows

Customer	Demand (units)	X	Y	Earliest (min)	Latest (min)	Earliest (HH:MM)	Latest (HH:MM)
1	5	38.079	95.121	484	585	08:04	09:45
2	3	73.467	60.267	569	688	09:29	11:28
3	8	16.446	16.443	598	670	09:58	11:10
4	6	6.75	86.751	493	603	08:13	10:03
5	3	60.51	71.099	582	673	09:42	11:13
6	1	3.038	97.021	506	604	08:26	10:04
7	3	83.412	22.022	488	596	08:08	09:56
8	5	19.001	19.157	558	669	09:18	11:09
9	3	31.12	52.951	494	585	08:14	09:45
10	1	43.763	29.832	569	632	09:29	10:32