Università degli Studi di Brescia



Università di Brescia

Dipartimento di Ingegneria dell'Informazione
Dottorato di ricerca in Ingegneria dell'Informazione

CICLO
XXXVIII

Mathematical programming methods for manufacturing and distribution scheduling problems

Dottorando:
Lorenzo Tomasetti

Relatore:
Prof.ssa Renata Mansini

Settore scientifico disciplinare:
MAT/09 - Ricerca Operativa

A mia nonna Lilli,

che spero, ovunque si trovi, sia felice del suo primo nipote diventato dottore di ricerca.

Acknowledgements

I would like to express my sincere gratitude to everyone who, directly or indirectly, supported me throughout the past three years and helped me through the challenging yet rewarding experience of pursuing a PhD.

First and foremost, I would like to thank my supervisor, Renata Mansini, for giving me this opportunity and for guiding me throughout this journey. Thanks to her support and expertise, I have been able to grow both academically and personally and to experience some of the most rewarding and challenging years of my life.

I would also like to express my gratitude to the entire research group. In particular, I would like to thank Daniele, with whom I had the opportunity to collaborate on research articles and teaching activities. His guidance and support were fundamental in helping me better understand how to approach my work and develop as a researcher. I am equally grateful to Alessandro and Roberto, who, as the more senior members of the group, generously shared their knowledge and offered invaluable advice on both research and teaching.

Last but certainly not least, I would like to thank my fellow PhD candidates, with whom I shared all or part of this journey: Valentina, Filippo, Teresa, Annalisa, and Enrico. You made these long and demanding years truly unforgettable.

My deepest thanks go to my mother and my sister, who have seen me in every possible mental and physical state—from periods of intense stress to moments of low motivation. I know that my frequent absence and the limited time we were able to spend together were not easy for either of them. Nevertheless, I always knew that they supported me, and that was more than enough to keep me going. Even though you may not understand much of my research, I hope that you are proud of me.

I would also like to thank all my amazing friends, who were always there to celebrate with me, comfort me, and, when necessary, bring me back down to earth. Your friendship and support have meant more to me than I can express.

Finally, I am grateful to all the wonderful people from around the world whom I had the opportunity to meet along the way. Thank you for the ideas we exchanged, the conversations we shared, and the experiences we enjoyed together. Each of you contributed, in your own way, to making this journey so meaningful.

Abstract

This thesis investigates the application of advanced mathematical programming techniques to complex manufacturing and distribution scheduling problems, with a particular focus on real-world industrial contexts. Modern production and logistics systems are characterized by high complexity, multiple interacting components, and the need to balance conflicting objectives such as cost efficiency, quality, and environmental sustainability. To address these challenges, this work develops integrated optimization models that combine exact methods with heuristic and matheuristic approaches.

On the methodological side, the thesis adopts exact optimization techniques, including Branch-and-Cut, mainly to derive optimal solutions for small-size instances and to establish benchmark bounds for heuristic methods. For large-scale and computationally challenging problems, the focus shifts toward decomposition-oriented modeling and the design of tailored heuristics and matheuristics, including custom constructive heuristics, Adaptive Large Neighborhood Search, and Two-Phase Kernel Search, aimed at efficiently navigating complex solution spaces and producing high-quality solutions within practical computational limits.

From an application perspective, the first part of the thesis focuses on manufacturing systems, with a specific emphasis on the modeling and optimization of decision-making problems in the steelmaking production process.

Specifically, the contributions cover scrap blending under multiple objectives (cost, energy consumption, and quality), multi-period production planning with energy considerations aimed at cost reduction across casting sequences, and hybrid flow shop scheduling tailored to the steelmaking process. Novel mathematical formulations and solution algorithms are proposed, demonstrating significant improvements in computational performance and solution quality.

The second part extends these methodologies to distribution systems, specifically last-mile delivery. A multi-objective optimization model is developed to support the integration of autonomous delivery robots with traditional vehicles, explicitly capturing trade-offs between operational efficiency and environmental impact. The proposed approaches provide actionable managerial insights and highlight the potential of advanced optimization for supporting sustainable logistics.

Overall, the thesis contributes to the field of operations research by proposing innovative models and solution techniques that bridge theory and practice. It demonstrates how mathematical programming, enhanced by decomposition and heuristic strategies, can effectively address complex, multi-domain decision-making problems in modern industrial systems.

Sommario

Questa tesi studia l'applicazione di tecniche avanzate di programmazione matematica a complessi problemi di schedulazione nei sistemi di produzione e distribuzione, con particolare riferimento a contesti industriali reali. I moderni sistemi produttivi e logistici sono caratterizzati da elevata complessità, molteplici componenti interagenti e dalla necessità di bilanciare obiettivi tra loro contrastanti, quali efficienza economica, qualità e sostenibilità ambientale. Per affrontare tali sfide, il lavoro sviluppa modelli di ottimizzazione integrati che combinano metodi esatti con approcci euristici e matheuristici.

Dal punto di vista metodologico, la tesi adotta tecniche esatte di ottimizzazione, tra cui il Branch-and-Cut, principalmente per ottenere soluzioni ottime su istanze di piccole dimensioni e per fornire valori di riferimento utili alla valutazione delle euristiche. Per problemi di grandi dimensioni e computazionalmente complessi, l'attenzione si sposta verso strategie di modellazione ispirate alla decomposizione e alla progettazione di euristiche e matheuristiche sviluppate ad hoc, tra cui euristiche costruttive, Adaptive Large Neighborhood Search e Two-Phase Kernel Search, finalizzate a esplorare efficacemente spazi delle soluzioni complessi e a produrre soluzioni di elevata qualità in tempi computazionali contenuti.

Dal punto di vista applicativo, la prima parte della tesi si concentra sui sistemi manifatturieri, con particolare attenzione alla modellazione e ottimizzazione di problemi decisionali nel processo produttivo siderurgico.

In particolare, i contributi riguardano: il problema della carica del rottame in ottica multi-obiettivo (costo, consumo energetico e qualità), la pianificazione multi-periodo della produzione con considerazioni energetiche finalizzate alla riduzione dei costi relativi alle sequenze delle colate, e modelli di schedulazione di tipo hybrid flow shop adattati al processo siderurgico. Vengono proposte nuove formulazioni matematiche e algoritmi risolutivi, che dimostrano significativi miglioramenti in termini di prestazioni computazionali e qualità delle soluzioni.

La seconda parte estende tali metodologie ai sistemi di distribuzione, con particolare riferimento al problema dell'ultimo miglio. Viene sviluppato un modello di ottimizzazione multi-obiettivo per supportare l'integrazione di robot autonomi per le consegne con veicoli tradizionali, catturando esplicitamente il compromesso tra efficienza operativa e impatto ambientale. Gli approcci proposti forniscono indicazioni utili a supporto delle decisioni manageriali e mettono in evidenza il potenziale delle tecniche di ottimizzazione avanzata nel promuovere una logistica sostenibile.

Nel complesso, la tesi contribuisce al campo della ricerca operativa proponendo modelli e tecniche risolutive innovativi che colmano il divario tra teoria e applicazioni reali. Essa dimostra come la programmazione matematica, integrata con strategie di decomposizione ed euristiche, possa affrontare efficacemente problemi decisionali complessi e multi-dominio nei moderni sistemi industriali.

Contents

Abstract iii

Sommario iv

1 Introduction 1

I Manufacturing 4

2 Bi-Objective Optimization in Steelmaking: Balancing Scrap Costs, Energy Consumption, and Steel Quality 5
- 2.1 Introduction . . . 6
- 2.2 Literature Review: Single-Period Blending . . . 8
- 2.3 Problem Description . . . 11
  - 2.3.1 Modeling Assumptions . . . 12
- 2.4 Mathematical Formulation . . . 13
- 2.5 Solution Methodologies for Multi Objective Programs . . . 15
- 2.6 Experimental Analysis . . . 15
  - 2.6.1 Instances . . . 15
  - 2.6.2 Trade-off results when f_P worsening is not allowed . . . 17
  - 2.6.3 Trade-off results when f_P worsening is allowed . . . 18
- 2.7 Conclusions . . . 20

3 A Combinatorial Approach to Steelmaking Cost Reduction: Liquid Heel and Energy Considerations in Casting Sequences 22
- 3.1 Introduction . . . 23
- 3.2 Literature Review: Multi-Period Blending . . . 25
- 3.3 Problem description . . . 27
  - 3.3.1 Modeling Assumptions . . . 29
- 3.4 Mathematical model . . . 30
  - 3.4.1 Managing subtour in the Hamiltonian path . . . 34
  - 3.4.2 Valid inequalities tightening the bounds on q_i^e . . . 34
- 3.5 Two-Step Constructive heuristic . . . 35
  - 3.5.1 Ranking methods . . . 36
  - 3.5.2 Aggregation methods . . . 39
  - 3.5.3 Heuristic sorting . . . 39
  - 3.5.4 Computing feasibility . . . 40

3.6 An ALNS Algorithm for the MHSSP-LH . . . 41
3.6.1 Introduction to ALNS . . . 41
3.6.2 The proposed ALNS Algorithm . . . 42
3.7 Experimental Results . . . 46
3.7.1 Instances description . . . 46
3.7.2 Numerical Tolerances and Precision Considerations . . . 48
3.7.3 Company Solution Definition . . . 49
3.7.4 Constructive Heuristic analysis . . . 50
3.7.5 Solving the complete model . . . 59
3.7.6 ALNS Evaluation . . . 63
3.8 Conclusions . . . 73

4 A Comprehensive Hybrid Flow Shop Scheduling Model for Steelmaking and Ingot Casting 74
4.1 Introduction . . . 75
4.1.1 Steelmaking Process Based on the Electric Arc Furnace . . . 76
4.1.2 From physical process to mathematical models . . . 79
4.2 Literature Review . . . 80
4.3 Problem description . . . 84
4.3.1 Modeling Assumptions . . . 87
4.4 Mathematical model . . . 88
4.4.1 The Hybrid Flow Shop Formulation . . . 88
4.4.2 Ingot Management . . . 91
4.4.3 Inter-machine eligibility . . . 93
4.4.4 Valid Inequalities . . . 93
4.4.5 Symmetry Breaking . . . 94
4.4.6 Lower bounds . . . 95
4.4.7 Upper bounds . . . 96
4.4.8 Basic and Advanced models . . . 97
4.5 Experimental Results . . . 98
4.5.1 Instances generation . . . 99
4.5.2 Analysis of the strengthening mechanisms . . . 101
4.5.3 Upper bounds . . . 103
4.5.4 Lower bounds . . . 106
4.5.5 Models comparison . . . 108
4.6 Conclusions . . . 113

II Distribution scheduling 115

5 Towards Carbon-Neutral Last-Mile Delivery: Balancing Environmental Impact and Service Quality with Autonomous Robots 116
5.1 Introduction . . . 117
5.2 Related literature . . . 119
5.3 Problem Definition and Mathematical Formulation . . . 121
5.3.1 Modeling Assumptions . . . 124
5.4 Solution Method . . . 125
5.4.1 Bi-Objective approach . . . 125
5.4.2 Two-Phase Kernel Search . . . 126

5.5 Experimental Analysis 128
5.5.1 Benchmark instances 129
5.5.2 Managerial Insights and Sensitivity Analysis 131
5.5.3 2P-KS performance analysis 136
5.5.4 Analysis of a realistic scenario 138
5.6 Conclusions and Future Work 141

6 Conclusions **143**

List of Algorithms

3.1 RankDenseSquared(e, H) . . . 38
3.2 RankDenseSquaredWildcards(e, H) . . . 38
3.3 AggregateBySum$(H, \mathscr{R})$ . . . 39
3.4 AggregateByNormalizedSum$(H, \mathscr{R})$ . . . 40
3.5 HeurSorting 1 (H) . . . 40
3.6 HeurSorting 2 (H) . . . 40
3.7 HeurSorting 3 (H) . . . 40
3.8 HeurSorting 4 (H) . . . 40
3.9 BestHeuristic$(\mathscr{P}, H)$ . . . 41
3.10 Adaptive Large Neighborhood Search . . . 42
3.11 ALNS within ISA Framework . . . 44
4.1 Constructive heuristic for the computation of UB_2 . . . 98
5.1 2P-KS$(\mathscr{I}, (b_1, \Delta_1, t_1), (b_2, \Delta_2, t_2), (b_3, \Delta_3, t_3), \omega, TH, T_{lim})$ . . . 127

List of Figures

2.1 Scrapyards charge example. . . . 17
2.2 Costs evolution using $(f_1|f_2, w)$ . . . 19
2.3 Square deviation evolution using $(f_2|f_1, w)$ . . . 19

3.1 Problem sequencing example. . . . 28
3.2 Graph based explanation of the liquid heel. . . . 33
3.3 Schematics of resource flow from node b to node u. . . . 33
3.4 Instances overlapping ranges. . . . 47
3.5 Cost reduction of best constructive heuristics compared to the company solution. . . . 51
3.6 Execution time for company solution and constructive heuristics. . . . 55
3.7 Gap percentage for company solution and constructive heuristics. . . . 56
3.8 $D1$ ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic. . . . 64
3.9 $D2$ ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic. . . . 64
3.10 $D3$ ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic. . . . 65
3.11 $D4$ ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic. . . . 65
3.12 $D6$ ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic. . . . 66
3.13 ALNS repair operator performance. . . . 68
3.14 ALNS destroy operator performance. . . . 69

4.1 Steelmaking process. . . . 78
4.2 Machines and resources available in the HFS model. . . . 85
4.3 Possible Flows. . . . 85
4.4 Pits and ingots molds example. . . . 86
4.5 Component analysis: Gap (%) . . . 102
4.6 Component analysis: Time To Best. . . . 103
4.7 Classical Gantt chart of the optimal solution. . . . 112
4.8 Machine and ladle utilization in the optimal solution. . . . 112
4.9 Solidification pit allocation in the optimal solution. . . . 113

5.1 Example of a Set B1.5 instance. . . . 130

5.2 Density of conflicts in the analyzed sets, comparison between 1 km ($A1$, $B1$) and 1.5 km ($A1.5$, $B1.5$). 130
5.3 Worsening of one objective function while allowing different percentage worsening of the other 132
5.4 Worsening of one objective function, in absolute value, while allowing different percentage worsening of the other 132
5.5 Objective function f_1 components 133
5.6 Objective function f_1 components 134
5.7 Percentage change of distance traveled by the truck and time traveled by the robots when optimizing f_1 with and without constraining f_2 to its optimal value 135
5.8 Example of a Pareto-efficient frontier 136
5.9 Time To First results 137
5.10 Time To Best results 139
5.11 Improvement between the first and second phase of 2P-KS. 139
5.12 Optimal solutions for one realistic scenario 140
5.13 Function f_1 components for the realistic scenarios 141
5.14 Kilometers traveled by components 141

List of Tables

2.1 Comparison of optimization-based single-heat scrap charging models. 11
2.2 Sets 12
2.3 Indices 12
2.4 Parameters 12
2.5 Decision variables summary 13
2.6 Costs distribution 17
2.7 Square deviation from $\hat{q}_e$ 18

3.1 Comparison of optimization-based multi-heat scrap charging models. 27
3.2 Sets 29
3.3 Indices 29
3.4 Parameters 29
3.5 Decision variables summary 31
3.6 Instances summary description. 47
3.7 Chemical composition related to the steel. 48
3.8 Cost comparison: real historical cost vs optimized company solution. 49
3.9 $D1$ Company and Constructive heuristic solution. 52
3.10 D2 Company and Constructive heuristic solution. 52
3.11 D3 Company and Constructive heuristic solution. 53
3.12 $D4$ Company and Constructive heuristic solution. 53
3.13 $D6$ Company and Constructive heuristic solution. 53
3.14 Feasibility results of company solution and heuristics. 54
3.15 Best solution occurrences of heuristics and number of instances where heuristics outperform the company solution. 54
3.16 $D1$ final gap and execution time for heuristics and company solution 56
3.17 $D2$ final gap and execution time for heuristics and company solution 57
3.18 $D3$ final gap and execution time for heuristics and company solution 57
3.19 $D4$ final gap and execution time for heuristics and company solution 57
3.20 $D6$ final gap and execution time for heuristics and company solution 58
3.21 Average results across all instance sets. 60
3.22 Results for the complete model over $D1$ instances. 60
3.23 Results for the complete model over $D2$ instances. 61
3.24 Results for the complete model over $D3$ instances. 61
3.25 Results for the complete model over $D4$ instances. 62
3.26 Results for the complete model over $D6$ instances. 62
3.27 ALNS cost and time summary for benchmark set $D1$ 70
3.28 ALNS cost and time summary for benchmark set $D2$ 71

3.29 ALNS cost and time summary for benchmark set $D3$ . . . 71
3.30 ALNS cost and time summary for benchmark set $D4$ . . . 71
3.31 ALNS cost and time summary for benchmark set $D6$ . . . 72

4.1 Chronological overview of models for steelmaking scheduling. . . . 84
4.2 Model sets . . . 87
4.3 Indices . . . 87
4.4 Model parameters . . . 87
4.5 Decision Variables . . . 89
4.6 Comparison between the basic and the advanced MILP formulations . . . 97
4.7 Processing times statistics at each production stage (in minutes). . . . 100
4.8 Average comparison between UB_1 and UB_2 across the benchmark sets. . . . 104
4.9 Comparison between UB_1 and UB_2 . . . 105
4.10 Percentage improvements between lower bounds (LB_1, LB_2, LB_3) . . . 106
4.11 Linear relaxation and Lower Bounds values comparison . . . 107
4.12 Summary of average results across instance sets. . . . 109
4.13 Basic and advanced model comparison for $D1$ instances. . . . 110
4.14 Basic and advanced model comparison for $D2$ instances. . . . 110
4.15 Basic and advanced model comparison for $D3$ instances. . . . 111
4.16 Basic and advanced model comparison for $D4$ instances. . . . 111
4.17 Basic and advanced model comparison for $D6$ instances. . . . 111

5.1 Comparison of optimization-based last-mile delivery models. . . . 121
5.2 Sets . . . 122
5.3 Indices . . . 122
5.4 Parameters . . . 122
5.5 Decision variables summary . . . 123
5.6 Common instance parameters. . . . 131
5.7 Set specific parameters. . . . 131
5.8 Absolute difference of gCO_2e from $\rho_{max} = 3$ to $\rho_{max} = 5$. . . . 133
5.9 Gurobi and 2P-KS on instances from B1 set . . . 136
5.10 Gurobi and 2P-KS on instances from B1.5 set . . . 137

Chapter 1

Introduction

Manufacturing and distribution systems constitute the backbone of modern industrial and logistics networks. Their efficient management is crucial for ensuring competitiveness, sustainability, and responsiveness to market demands. However, optimizing such systems is inherently challenging due to their complexity, the presence of multiple interacting components, and the need to account for heterogeneous decision processes.

In practice, industrial optimization problems rarely belong to a single well-defined class. Instead, they often combine elements from different areas of operations research, such as cutting and packing, scheduling, blending, inventory management, and routing. This is particularly evident in the applications considered in this thesis, where manufacturing processes and distribution systems require the integration of multiple modeling paradigms. As a consequence, classical approaches based on isolated models are often insufficient, and there is a need for more flexible and integrated optimization frameworks.

This thesis embraces this complexity by addressing a variety of problems arising in both manufacturing and distribution contexts. The breadth of the considered applications reflects the interdisciplinary nature of modern operations research, where different methodological tools must be combined to effectively tackle real-world challenges. For this reason, the thesis is intentionally structured across multiple domains, highlighting both the diversity of the addressed problems and the unifying role of mathematical programming techniques.

From a methodological perspective, the thesis focuses on the development and application of advanced optimization approaches, combining exact methods with heuristic and matheuristic techniques. A central theme is the design of models that balance realism and tractability, together with solution methods that exploit problem structure.

On the exact side, Branch-and-Cut techniques are employed. This approach enhances mixed-integer formulations by incorporating problem-specific valid inequalities directly within the branch-and-bound process, thereby tightening the linear relaxation and improving computational performance. For instance, in Chapter 4, the model is solved using a Branch-and-Cut framework enriched with valid inequalities and symmetry-breaking constraints.

In addition to exact methods, the thesis places strong emphasis on the development of heuristic and matheuristic approaches, which are essential for tackling large-scale and highly constrained instances. Several ad-hoc heuristics are proposed, specifically tailored to the structure of the considered problems. These include constructive procedures, based on ranking and aggregation methods as explained in Chapter 3 or like in Chapter 4 based on classical properties of the underlying problem. Such approaches are designed to provide high-quality solutions and/or to support exact methods by improving bounds and initialization.

More advanced metaheuristic frameworks are also developed and adapted. In particular, an Adaptive Large Neighborhood Search (ALNS) algorithm is introduced in Chapter 3, while a Two-Phase Kernel Search approach is proposed in Chapter 5. These matheuristics combine the strengths of mathematical programming and heuristic search, enabling the efficient exploration of large solution spaces.

Another important aspect of this work is the consideration of multi-objective optimization problems. In real-world applications, decision makers must often balance conflicting criteria, such as cost, quality, and environmental impact. This is explicitly addressed in Chapter 2, where a bi-objective model is developed for scrap selection in steelmaking, and in Chapter 5, where environmental and service-level objectives are jointly optimized in a distribution context.

While a significant part of the thesis is devoted to manufacturing systems—particularly within the steelmaking industry—the second part focuses on distribution scheduling problems. In recent years, the rapid growth of e-commerce and urban logistics has made last-mile delivery a critical component of supply chains. In this context, new technologies such as autonomous delivery robots are emerging, introducing additional layers of complexity in routing and scheduling decisions.

Chapter 5 addresses these challenges by proposing a novel optimization model for last-mile delivery systems that integrate traditional vehicles and autonomous robots. The problem is formulated as a multi-objective optimization model, capturing both operational efficiency and environmental impact. This part of the thesis highlights how matheuristics methodological tools can be extended and adapted to distribution systems, reinforcing the unifying role of mathematical programming.

Overall, the thesis covers a wide range of problem classes and application domains. This diversity is not incidental but reflects the need to address complex, real-world systems that span multiple areas of operations research. The common thread across all contributions is the use of advanced optimization techniques to model, analyze, and solve challenging decision-making problems.

Notation Throughout this thesis, standard notation from mathematical programming and combinatorial optimization is adopted. Whenever necessary, chapter-specific notation is introduced and clearly defined.

Contributions

The main contributions of this thesis are as follows:

1. The design of novel mathematical models for manufacturing systems, with a focus on steelmaking processes, including blending, sequencing, and scheduling problems (Chapters 2 to 4).

2. The development of advanced exact optimization approaches based on Branch-and-Cut (Chapter 4).

3. The analysis of distribution scheduling problems, with applications to last-mile delivery systems involving autonomous technologies (Chapter 5).

4. The introduction of multi-objective optimization frameworks (Chapters 2 and 5) to analyze trade-offs between cost, quality, and environmental impact.

5. The development of ad-hoc heuristic methods, including constructive and feasibility heuristics tailored to specific industrial problems (Chapters 3 and 4).

6. The design of matheuristic approaches, such as Adaptive Large Neighborhood Search (Chapter 3) and Two-Phase Kernel Search (Chapter 5), to efficiently solve large-scale instances.

7. The integration of linear and quadratic programming models to better represent real-world phenomena (Chapter 2).

Structure of the Thesis

This thesis is organized into two main parts and six chapters. The first chapter introduces the research context and objectives of the thesis, while the final chapter summarizes the main findings and outlines directions for future research.

Part I focuses on manufacturing optimization problems and presents models and solution approaches for industrial production systems, particularly in the steelmaking sector.

Part II focuses on distribution scheduling problems, addressing last-mile delivery systems and the integration of emerging technologies such as autonomous robots.

The thesis is divided into structured chapters as follows.

In Chapter 2, a bi-objective optimization model is introduced for scrap selection in steelmaking.

In Chapter 3, a multi-period combinatorial model for the casting sequence is developed, together with ad-hoc heuristics and an ALNS-based solution approach.

In Chapter 4, a hybrid flow shop scheduling model, for managing the interaction of machines in a plant-wide system for steelmaking, is proposed and solved using advanced mixed-integer programming techniques.

Finally, in Chapter 5, a distribution scheduling problem is addressed, focusing on last-mile delivery with autonomous robots and multi-objective optimization.

Each chapter is self-contained and includes a literature review, problem definition, and methodological contributions.

This thesis aims to demonstrate how advanced mathematical programming techniques can effectively address complex decision-making problems across multiple domains, bridging the gap between theory and real-world industrial applications.

Part I

Manufacturing

Chapter 2

Bi-Objective Optimization in Steelmaking: Balancing Scrap Costs, Energy Consumption, and Steel Quality

The content of the following chapter is adapted and extended from Daniele Manerba, Renata Mansini, Lorenzo Tomasetti, and Roberto Zanotti. Bi-objective optimization in steelmaking: Balancing scrap costs, energy consumption, and steel quality. IFAC-PapersOnLine, *59(10): 1706–1711, 2025, presented at the 11th IFAC Conference on Manufacturing Modelling, Management and Control – IFAC MIM2025.*

Contents

2.1 Introduction **6**
2.2 Literature Review: Single-Period Blending **8**
2.3 Problem Description **11**
2.3.1 Modeling Assumptions 12
2.4 Mathematical Formulation **13**
2.5 Solution Methodologies for Multi Objective Programs **15**
2.6 Experimental Analysis **15**
2.6.1 Instances 15
2.6.2 Trade-off results when f_P worsening is not allowed 17
2.6.3 Trade-off results when f_P worsening is allowed 18
2.7 Conclusions **20**

In this chapter, we present a bi-objective optimization approach for steelmaking, with a focus on scrap-loading. The first objective minimizes the costs associated with raw materials and the melting energy, while the second one minimizes deviation from the required chemical composition of the final product. We analyze key metrics of scrapyard composition, such as size, similarity, and variance, and assess their impact on production outcomes. The results show that scrapyard configuration plays a crucial role in both cost and quality, with larger

and optimized scrapyards leading to improved efficiency. By examining trade-offs using a lexicographic approach, this work provides valuable insights for bridging theoretical models with practical strategies aimed at achieving sustainable and cost-effective steel production.

2.1 Introduction

Scrap-based steelmaking is increasingly central to modern production systems, both for economic competitiveness and for sustainability objectives as shown by World Steel Association (2021). In particular, the electric arc furnace (EAF) route relies heavily on scrap as a primary input material, making the *scrap mix* (or scrap selection) decision a critical operational lever. While scrap enables flexible and potentially lower-emission production pathways, it also introduces a distinctive set of complexities: heterogeneous chemical composition, volatile availability and pricing, handling and logistics constraints in the scrapyard, and non-trivial interactions with process energy consumption.

At the operational level, scrap selection is often carried out at the granularity of a *single heat* (or single casting), i.e., determining the set and quantities of raw materials to load into the EAF to produce a target amount of liquid steel with prescribed chemical specifications. The decision is inherently constrained by multi-element quality requirements (minimum and maximum thresholds for several chemical elements), furnace capacity limits, and practical loading constraints as in Kim and Lewis (1987); Rong and Lahdelma (2008). Even when the nominal chemical composition of each scrap type is available, the realized composition may deviate due to within-grade variability, contamination, and measurement errors, which can result in off-spec heats, corrective alloying can be a solution procedure but in the worst case, the results may be the disruption of production Rong and Lahdelma (2008); Sakallı et al. (2011). Not managing these quality failures effectively translates directly into costs: additional ferroalloys, extra melting time, increased electricity consumption, reduced productivity, and potential downstream disruptions.

Beyond uncertainty, scrap selection is also shaped by *implementability*. As described by Kowalik (2019) scrap types are not always divisible at arbitrary resolution: they may be loaded in discrete quantities due to crane operations, buckets, containers, or fixed-size portions, and their movement from yard to furnace is subject to operational frictions. Moreover, yard organization and internal logistics can impose binding limits on which combinations are feasible in practice, even if they appear feasible in a purely chemical blending model as explained by Bernatzki et al. (1998). Consequently, the gap between an analytically optimal blend and an operationally feasible blend can be substantial unless these constraints are explicitly modeled.

A further layer of complexity arises from the coupling between scrap choice and energy-related costs. The energy required to melt a batch depends not only on the total mass but also on material properties and process parameters. As electricity pricing becomes more volatile and tariffs increasingly reflect time-of-use structures, the economic impact of scrap selection can no longer be evaluated by material cost alone as explained by Yang et al. (2018). This motivates optimization frameworks that integrate scrap procurement costs with energy consumption, and that quantify trade-offs between short-term cost reductions and the ability to meet stringent chemical targets.

Within this context, this chapter focuses on the single-heat scrap selection problem. The proposed Cost–Quality Scrap Selection Optimization Problem (CQ-SSOP) models scrap loading as a structured decision problem in which cost efficiency and chemical performance are simultaneously relevant objectives. Specifically, the framework integrates (i) raw ma-

terial and theoretical melting energy costs, (ii) explicit chemical composition constraints and target-driven quality metrics, and (iii) operational constraints reflecting the discrete and bounded nature of scrap handling. This perspective differs from classical single-heat blending formulations that optimize cost under hard quality constraints as in Kim and Lewis (1987); Sakalli and Birgoren (2009) and from uncertainty-centered approaches that focus primarily on feasibility under variability as in Rong and Lahdelma (2008); Sakallı et al. (2011). Instead, it treats chemical quality as an optimization objective in its own right, enabling a controlled exploration of cost–quality trade-offs relevant to decision makers in real production settings.

Ultimately, the practical significance of this modeling decision is further proven by the function of scrapyard composition as an internal factor affecting performance. The available set of scrap types, their similarity to the intended production portfolio, and their internal chemical dispersion can affect both achievable quality and total cost. By making these relationships explicit, the chapter provides a foundation for decision support that goes beyond computing a single blend: it supports a deeper understanding of how scrapyard design and selection policies shape operational outcomes in single-heat steel production.

This chapter offers several key contributions, the Cost–Quality Scrap Selection Optimization Problem (CQ-SSOP) proposed consolidates these dimensions within a unified single-heat model. Its methodological contributions can be summarized as follows:

(i) A bi-objective formulation that simultaneously minimizes total production cost and deviation from the target chemical composition.

(ii) A mixed-integer representation explicitly modeling discrete scrap handling constraints at furnace level.

(iii) An explicit consideration of the liquid heel component of the melting process.

(iv) Direct integration of melting-related energy costs within the objective function.

(v) A structured trade-off exploration based on a lexicographic optimization scheme allowing controlled deterioration of prioritized objectives.

(vi) A systematic analysis of scrapyard structural characteristics (size, similarity, and chemical dispersion) as determinants of economic and chemical performance.

By integrating these components within a single formulation, the model advances the single-heat literature from isolated modeling refinements toward a structurally comprehensive decision-support framework, which can help decision makers to take high level decision about the plant production.

Extensive computational experiments run on synthetic instances constructed over the realistic setting prove that our approach can give helpful insight for the high level decision maker.

The remainder of this chapter is structured as follows. Section 2.2 outlines the relevant literature related to single heat blending problems. Section 2.3 presents the problem under investigation and in the Section 2.4, the bi-objective mixed-integer quadratic programming formulation of the defined problem is presented. Section 2.5 details the solution methodology regarding the resolution of the multi-objective problem. Section 2.6 focuses on our experimental analysis, describing the realistic instances used, the tests performed, and the results obtained. Finally, Section 2.7 discusses conclusions and potential directions for future research.

2.2 Literature Review: Single-Period Blending

The challenges associated with blending and charge calculation emerge in metallurgical processes that necessitate the integration of various input materials, including scrap classes and ferroalloys, to generate a melt that complies with chemical specifications while minimizing costs.

Literature related to the optimization of *single-heat* (single-furnace charge) or *single-casting* decisions, specifically addresses the process of optimizing one melt or charge at a time. This deals with the operational decision-making process that coincides with a melt in foundries and steelmaking. It involves the selection of quantities of available materials to ensure that the resulting composition is within a specified lower and upper limits for various elements. Additionally, it must consider charge mass requirements and process-specific practicalities, including but not limited to special material handling with finite precision.

The mathematical structure of blending is one of the oldest LP problem addressed in the literature, that is however known as the diet problem defined by Stigler (1945), which established the paradigm of minimizing cost subject to nutritional or, in the case of metallurgical process, chemical constraints. In metallurgy, the deterministic single-heat blending model is typically formulated as a linear program: decision variables represent the mass of each input material in the charge; constraints enforce the total charge mass, element-by-element composition bounds (lower/upper limits), and sometimes yield; and the objective minimizes total material cost. This formulation remains attractive because it is transparent, computationally efficient, and easily integrated into decision-support tools.

The industrial relevance of deterministic LP for charge calculation is demonstrated by large-scale applications in foundry operations, where least-cost charging was implemented and tested on real data at significant scale since the first work of Kim and Lewis (1987). While is often positioned as an operational research application rather than a methodological contribution, its enduring value is that it validated LP-based charge calculation as a practical mechanism for real foundry melting operations, thereby motivating later studies that incorporate uncertainty, discreteness, and integration with other shop-floor decisions.

Beyond metallurgy-specific contexts, general blending modeling in process manufacturing has also contributed important perspectives on how blending models are deployed and maintained, including issues such as data management, organizational embedding, and the relationships between model fidelity and usability as explained by Ashayeri et al. (1994). These broader lessons translate directly to single-heat scrap charging: even a theoretically sound model can fail operationally if it demands data that the plant cannot reliably measure or if the solution logic cannot be explained to practitioners.

A substantial body of work on single-heat blending is developed, inside similar process industries as in the brass casting context, where scrap and return variability is high and pure metal additions can be expensive. A representative contribution is the development of a spreadsheet-based decision support tool for the brass casting blending problem by Sakalli and Birgoren (2009). The underlying optimization is consistent with the deterministic LP paradigm, but the principal contribution is its operational integration. The model is embedded into a spreadsheet environment and structured around the practical workflow of preparing a melt. From a single-heat perspective, this highlights a central industrial reality: the optimization is rarely executed once by engineers but often is done by expert workers. This creates the need of abstracting complex information into a simpler decision support system that can be used daily by expert workers that may lack strong mathematical background. While deterministic LP formulations remain the operational backbone of single-heat

charging, practical melting operations are rarely free from disturbances. Deviations between planned and realized chemistry, variability in scrap composition, and measurement imprecision frequently require corrective actions during or after the initial charge preparation. This motivates a more formal treatment of correction decisions within the single-heat framework.

A mathematical perspective on such adjustments is provided by best corrected mixture models, which explicitly determine the minimal modification required to restore feasibility when an initial mixture violates target constraints as in Dubeau and Ntigura Habingabwa (2017). In the context of single-heat blending, this clarifies that the correction step can itself be formulated as an optimization problem. Even when not explicitly solved in practice, the existence and cost of corrective actions influence how conservative the initial blending decision should be. If inexpensive and rapid corrections are available, less conservative initial charges may be economically justified.

Beyond corrective adjustments, a major stream of research addresses uncertainty in input material chemistry. Scrap composition is often subject to significant variability, especially when materials are sourced externally or aggregated into heterogeneous classes. Stochastic optimization approaches therefore reformulate the blending problem using chance constraints, requiring that elemental specifications be satisfied with a prescribed probability. A case in brass casting by Sakallı et al. (2011) demonstrate how probabilistic feasibility can replace rigid deterministic bounds, making the cost–risk trade-off explicit and reducing reliance on ad hoc safety margins.

However, probabilistic models assume the availability of reliable distributional information. In practice, scrap chemistry may be characterized by limited samples or expert estimates rather than statistically robust datasets. To address this limitation, models have been proposed that consider uncertainty as in Sakallı and Baykoç (2011). For single-heat blending, this issue is relevant because externally sourced scrap or infrequently used materials may exhibit substantial informational uncertainty. Incorporating uncertainty prevents overconfidence in parameter estimates and yields more realistic feasibility assessments.

An alternative representation of uncertainty is provided by fuzzy chance-constrained programming, where uncertain parameters are modeled as fuzzy numbers and constraint satisfaction is enforced at a specified confidence or possibility level as done by Rong and Lahdelma (2008). The relevance to single-heat models lies in the ability to (i) encode imprecise chemistry ranges and (ii) tune conservatism via the chosen satisfaction level. Compared to purely stochastic chance constraints, fuzzy formulations can be easier to calibrate when only expert knowledge or limited sampling is available, though they also introduce modeling choices regarding membership functions and interpretation.

More recently, robust optimization has emerged as a distribution-free approach that guarantees feasibility for all realizations within a predefined uncertainty set. In scrap charge optimization, robust models are particularly attractive when the cost of off-specification production is high and reliable probabilistic characterization is unavailable as in Yang et al. (2018). Rather than controlling violation probability, robust formulations enforce constraint satisfaction under bounded deviations of elemental concentrations. For single-heat decisions, this provides a transparent and conservative mechanism to ensure compliance under worst-case variability.

A persistent limitation of classical blending LP is the assumption of continuous divisibility of materials. In practice, many materials are available or charged in discrete units: standardized ingots, bundles, bins, or fixed-size portions. Ignoring indivisibility can produce solutions that are optimal on paper but suboptimal or in the worst case infeasible or costly in practice. Operational constraints also arise from scrapyard logistics, charging equipment

limitations, and rules on mixing certain scrap types, which easily arise in a production process where the safety measures must always be placed first and cannot be ignored.

A prominent illustration of these realities in steel production is the formulation of optimal scrap combination models that account for both chemical constraints and transportation/handling limitations as in Bernatzki et al. (1998). Even though such models may include broader operational features, the key single-heat insight is that feasibility can depend as much on material movement and unit capacities as on chemical balance. Therefore, charging should directly consider chemical feasibility while managing the practical implementation under discrete and logistical constraints.

Discrete charging also appears in die casting, where alloy may be supplied as solid ingots and charging decisions are effectively quantized as in Park and Yang (2011). While that work addresses process efficiency with scrap recycling and ingot adjustment, it reinforces a general single-heat principle: if material units are indivisible, then either the optimization must explicitly incorporate integer variables or the solution process must include a systematic rounding/repair step that preserves feasibility and cost-effectiveness.

A structured way to integrate the indivisibility is to model fixed-size portions directly in the optimization formulation. An example of this ca be seen by Kowalik (2019) who developed a chance-constrained blending with components available in fixed-size portions explicitly targeting this setting and demonstrating how uncertainty and discreteness interact. For single-heat charging, this is particularly relevant when procurement and charging occur in standardized lots and when element variability implies that discrete choices can materially change risk. The methodological takeaway is that discrete modeling is not a secondary detail: it can qualitatively change the shape of the feasible region.

Single-heat blending decisions are repeated frequently, and their economic impact depends on both input prices and material quality. Consequently, decision support tools that quantify the economic value of scrap mixes and compare alternatives can be as important as the optimization model itself. Tools developed for comparing the economic value of different scrap mixes in steel production highlight how blending optimization can be embedded in a broader decision-support context involving simulation, scenario evaluation, and reporting as done by Fornai et al. (2013). From a single-heat perspective, these tools support tasks such as: evaluating new scrap suppliers, understanding the marginal value of reducing variability in a scrap class, and performing sensitivity analysis on alloying additions.

These contributions reinforce an important design principle: single-heat charging models should be accompanied by diagnostic outputs and or KPI (shadow prices, binding constraints, risk measures, sensitivity to chemistry and cost parameters) to facilitate trust and adoption. In industrial settings, users often need not only a recommended charge, but also an explanation of which constraints drive the decision and how robust the recommendation is to expected variability.

Across the single-heat literature, two intertwined modeling dimensions consistently structure the evolution of the field. On the one hand, early deterministic LP formulations provide a computationally efficient and operationally attractive baseline for cost minimization under strict chemical constraints as shown by Kim and Lewis (1987); Sakalli and Birgoren (2009). On the other hand, subsequent contributions introduce uncertainty-aware formulations that reinterpret feasibility in probabilistic, or robust terms, allowing explicit control of off-specification risk as in Sakallı et al. (2011); Rong and Lahdelma (2008); Sakallı and Baykoç (2011); Yang et al. (2018).

At the same time, the assumption of continuous material divisibility—central to classical blending models—has progressively been challenged by industrial realities. Discrete

charging units, logistical limitations, and handling constraints have motivated integer and mixed-integer extensions as in Bernatzki et al. (1998); Kowalik (2019); Park and Yang (2011), demonstrating that plant constraints and discretization of the input materials are not merely a computational refinement but a structural modeling requirement. Together, these developments show a gradual enrichment of the single-heat framework, moving from purely cost-driven continuous models toward formulations that incorporate risk control and operational feasibility.

Taken together, the literature shows a clear methodological progression: deterministic cost minimization, followed by uncertainty-aware feasibility control, and subsequently by discrete and operationally enriched formulations. Nevertheless, these dimensions are typically addressed in isolation. Most contributions remain single-objective, treat chemical quality as a strict constraint rather than a measurable performance indicator, and rarely integrate energy costs within the same optimization framework. Furthermore, the scrapyard composition is generally assumed to be exogenous data rather than analyzed as a strategic determinant of performance.

This chapter addresses these limitations by proposing the Cost–Quality Scrap Selection Optimization Problem (CQ-SSOP), a bi-objective mixed-integer quadratic programming model that jointly optimizes production cost and steel quality while explicitly incorporating discrete charging constraints, liquid heel effects, and energy consumption. In addition, the proposed framework systematically investigates how scrapyard characteristics influence the cost–quality trade-off, providing managerial insights that extend beyond the scope of previous single-heat scrap charging models.

Table 2.1 summarizes the positioning of this contribution relative to the principal optimization-based single-heat scrap charging models discussed above.

Table 2.1: Comparison of optimization-based single-heat scrap charging models.

Work	Uncertainty	Discrete Materials	Multi-Objective	Energy	Quality as Objective	Scrapyard Integration
Kim and Lewis (1987)	○	○	○	○	○	○
Bernatzki et al. (1998)	○	●	○	○	○	◐
Rong and Lahdelma (2008)	●	○	○	○	◐	○
Sakalli and Birgoren (2009)	○	○	○	○	○	○
Sakallı et al. (2011)	●	○	○	○	◐	○
Sakallı and Baykoç (2011)	●	○	○	○	◐	○
Yang et al. (2018)	●	◐	○	●	○	◐
Dubeau and Ntigura Habingabwa (2017)	○	○	○	○	◐	○
Park and Yang (2011)	○	●	○	○	○	○
Kowalik (2019)	●	●	○	○	◐	○
Fornai et al. (2013)	◐	○	○	○	○	◐
This work	○	●	●	●	●	●

Notation: ● = fully addressed; ◐ = partially addressed; ○ = not addressed.

2.3 Problem Description

Let $R = S \cup K$ represent the set of raw materials, where S denotes the scraps set and K represents the ferroalloys set. Let E be the set of various chemical elements present in the raw materials. For a given type of steel, each chemical element $e \in E$ has a predefined minimum $\underline{\beta^e}$ and maximum $\overline{\beta^e}$ percentage, as well as a target value $\hat{q}^e$. Each raw material $r \in R$ has known chemical compositions α_r^e for each element $e \in E$, along with a cost c_r, a liquid coefficient μ_r, and a specific energy parameter m_r. Together with the energy cost γ, these parameters determine the theoretical melting cost. The required production

amount is determined each time a new steel heat needs to be produced. To achieve the desired production quantity P, each material must be properly scaled by its liquid coefficient, which indicates how much raw input is converted into liquid steel. Since all materials are processed in an electric arc furnace (EAF), an important aspect is the furnace's maximum capacity, denoted as C, which must never be exceeded for operational and safety reasons. Another factor influencing the target chemical composition for a specific type of steel is the residual liquid heel already present in the EAF, denoted by λ^e, which accounts for the small amount of steel remaining from the previous casting. However, relying solely on this information overlooks important practical challenges in production. A key issue is that not all materials are handled in the same way. To address this, we introduce a parameter Δ, which measures the ease of moving materials from the scrapyard to the EAF, without affecting ferroalloys. While industrial cranes generally operate on a kilo scale, accurately loading precise amounts into the furnace can be challenging. These handling difficulties can lead to delays or unintended variations in scrap composition. Hence, to better align the model with real-world production, two additional parameters are introduced: M_1, representing the available scrap supply, and M_2, the minimum scrap quantity required in the EAF that ensures efficiency. While adding smaller scrap amounts can be cost-effective, it may increase loading times and reduce overall productivity.

The main objective of the problem is to minimize the total cost, which is the sum of the scrap cost and the energy cost associated with the melting process, while also optimizing the final steel chemical composition (quality). We refer to this problem as the Cost-Quality Scrap Selection Optimization Problem (CQ-SSOP).

Table 2.2: Sets

$R = S \cup K$	Set of raw materials (scrap and ferroalloys)
S	Set of scrap materials
K	Set of ferroalloys
E	Set of chemical elements

Table 2.3: Indices

r	Raw material
s	Scrap
k	Ferroalloy
e	Chemical element

Table 2.4: Parameters

α_r^e	Percentage of element e in material r
c_r	Cost of material r
μ_r	Liquid coefficient of material r
m_r	Specific energy parameter of material r
γ	Energy cost coefficient
β^e	Minimum allowed percentage of element e
$\overline{\beta}^e$	Maximum allowed percentage of element e
$\hat{q}^e$	Target percentage of element e
λ^e	Residual (heel) amount of element e in EAF
P	Required production quantity
C	Maximum capacity of the furnace
Δ	Scrap handling difficulty parameter
M_1	Maximum available scrap supply
M_2	Minimum required scrap quantity

2.3.1 Modeling Assumptions

The mathematical model presented in this chapter relies on a set of assumptions that balance industrial realism with mathematical tractability. Several of these assumptions replace uncertain or dynamic industrial phenomena with deterministic approximations. They are stated explicitly below together with their motivation.

Deterministic material composition. Each raw material is assumed to have a fixed and exactly known chemical composition for every element, together with a fixed unit cost and energy consumption. In practice, the composition of raw materials may vary due

to supplier variability, contamination, measurement errors, and the absence of continuous monitoring. These sources of uncertainty are neglected because the objective of this work is to investigate the interaction between production cost, steel quality and scrapyard composition rather than the impact of uncertainty over input data. Assuming deterministic material properties also keeps the resulting optimization model computationally tractable while preserving consistency across the operational decisions, that are the focus of this work.

Deterministic energy price. The electricity price is assumed to remain constant throughout the optimization horizon. Although electricity markets are characterized by time-varying prices, this simplification is reasonable while addressing the planning of a single heat, whose duration is relatively short compared with typical electricity pricing intervals. Consequently, treating the electricity price as constant provides a reasonable approximation while avoiding additional model complexity that falls outside the scope of this work.

These assumptions are consistent with the scope of the proposed model, which is intended to support strategic decision-making rather than detailed operational planning. Although the model relies on deterministic input parameters, the analyses derived from it remain relevant for understanding the principal trade-offs and decision drivers governing scrap selection, production cost, energy consumption, and steel quality.

2.4 Mathematical Formulation

In this section, we present a Mixed-Integer Quadratic Programming (MIQP) formulation for the CQ-SSOP. To this end, we introduce the following sets of decision variables. For each scrap $s \in S$, x_s represents the amount required for production, t_s denotes the integer quantity of scrap, and z_s is a binary variable, equal to 1 if scrap s is used, and 0 otherwise. For each ferroalloy $k \in K$, we define y_k as the amount used in the production process. Additionally, we introduce an auxiliary variable p, representing the raw materials that are transformed into molded steel in the EAF. A summary of the relevant variables is provided in Table 2.5.

Table 2.5: Decision variables summary

x_s	Continuous variable representing the amount of scrap $s \in S$ used for production
t_s	Integer variable representing the discrete quantity of scrap $s \in S$ loaded into the EAF
z_s	Binary variable equal to 1 if scrap $s \in S$ is used, 0 otherwise
y_k	Continuous variable representing the amount of ferroalloy $k \in K$ used for production
p	Continuous auxiliary variable representing the total amount of raw material converted into liquid steel in the EAF

Then, the problem can be formulated as follows:

$$f_1 = \min \sum_{s\in S}(c_s + m_s \cdot \gamma)x_s + \sum_{k\in K}(c_k + m_k \cdot \gamma)y_k \tag{2.1}$$

$$f_2 = \min \sum_{e\in E}\left(\frac{\sum_{s\in S}\alpha_s^e \cdot x_s + \lambda^e - \hat{q}^e}{\theta^e}\right)^2 \tag{2.2}$$

$$\text{s.t.} \sum_{s\in S}\alpha_s^e \cdot x_s + \sum_{k\in K}\alpha_k^e \cdot y_k + \lambda^e \le \overline{\beta^e} \cdot p, \quad \forall e \in E \tag{2.3}$$

$$\sum_{s\in S}\alpha_s^e \cdot x_s + \sum_{k\in K}\alpha_k^e \cdot y_k + \lambda^e \ge \underline{\beta^e} \cdot p, \quad \forall e \in E \tag{2.4}$$

$$p = \sum_{s\in S}\mu_s \cdot x_s + \sum_{k\in K}\mu_k \cdot y_k \tag{2.5}$$

$$p \ge P \tag{2.6}$$

$$\sum_{s\in S} x_s + \sum_{k\in K} y_k \le C \tag{2.7}$$

$$x_s \le M_1 \cdot z_s, \qquad \forall s \in S \tag{2.8}$$

$$x_s \ge M_2 \cdot z_s, \qquad \forall s \in S \tag{2.9}$$

$$x_s = \Delta \cdot t_s, \qquad \forall s \in S \tag{2.10}$$

$$z_s \in \{0,1\},\ x_s \ge 0,\ t_s \in \mathbb{Z}_0^+, \qquad \forall s \in S \tag{2.11}$$

$$y_k \ge 0, \qquad \forall k \in K. \tag{2.12}$$

The objective function (2.1) minimizes the total cost, which includes both the market price of the raw materials and the energy costs associated with melting scraps and ferroalloys. The objective function (2.2) measures the deviation from the predetermined quality target point $\overline{q}^e$ for each chemical element in the scrap. This deviation is scaled by a factor of θ^e to normalize the impact of different chemical ranges across elements. The squared deviation is then minimized to reduce the discrepancy between the actual and desired chemical composition. Constraints (2.3) and (2.4) enforce the upper and lower limits for each chemical element required to produce the specified steel type. These constraints account for contributions from scraps, ferroalloys, and the residual liquid mold in the furnace. Since the chemical composition requirements depend on the total steel produced, these limits are scaled based on the actual production volume. Constraint (2.5) ensures that the total amount of scrap and ferroalloy used is consistent with the desired steel production amount, which is essential for meeting the chemical requirements dictated by constraints (2.3) and (2.4). Constraints (2.6) and (2.7) impose weight restrictions: the first guarantees a minimum steel production quantity, while the second ensures that the total load does not exceed the EAF capacity. Constraints (2.8) and (2.9) regulate the quantity of selected scrap $s \in S$. If scrap s is selected, its quantity must be within the predefined bounds M_1 (maximum) and M_2 (minimum), reflecting real-world handling limitations in the production process. Constraint (2.10) accounts for the practical conditions of the production environment, allowing for more precise resource management. Finally, the variable domains and integrality restriction are specified in Constraints (2.11) and (2.12).

2.5 Solution Methodologies for Multi Objective Programs

Solving a multi-objective program requires specialized methods to identify Pareto-optimal solutions. Some approaches, such as the lexicographic method, follow an *a-priori* strategy, where the decision makers specify their preferences by establishing a strict priority order among objectives (see, for instance, Bonomi et al., 2024, 2025). In contrast, a-posteriori methods, including the weighted sum approach, the ϵ-constraint method, and NSGA-inspired heuristics, generate or approximate the set of Pareto-optimal solutions, allowing decision makers to explore trade-offs before making a selection.

In this work, we employ a two-step lexicographic optimization approach. The method proceeds as follows. In the first step, one objective function (f_P) is optimized, disregarding the other one. In the second step, the non-prioritized objective function (f_{NP}) is optimized while imposing a constraint on the deterioration of the first objective. Let f_P^* be the optimal value obtained in the first step. In the second step f_{NP} is optimized under the additional constraint:

$$f_P \leq f_P^*(1 + w)$$

where $0 \leq w \leq 1$ represents the maximum allowed percentage degradation of f_P with respect to f_P^*. We denote this general two-step optimization procedure as $(f_{NP}|f_P, w)$.

In our specific case, either f_1 or f_2 can be prioritized, allowing us to either minimize total cost first and maximize steel quality second, or reverse the order. Furthermore, by adjusting the w parameter, we can control trade-offs between the two objectives. Since f_2 is a quadratic function, the model formulation changes depending on its priority. When f_2 is the non-prioritized objective, the model remains quadratic in the objective function. Conversely, when f_2 is prioritized, it introduces a quadratic constraint. Although the inclusion of quadratic elements increases the complexity of the problem, all resulting models can still be efficiently solved using state-of-the-art commercial solvers.

2.6 Experimental Analysis

This section presents the experiments conducted to derive managerial insights. In particular, Section 2.6.1 describes the generation of benchmark instances, while Sections 2.6.2 and 2.6.3 analyze the results obtained under two scenarios: one where no deterioration of the prioritized objective is allowed, and another where a controlled deterioration is permitted.

The mathematical model was implemented in Java 21. All experiments were conducted on a Manjaro 23.1 system equipped with a Ryzen 5 5600U 12-core processor and 42 GB of RAM, using Gurobi 10.0.1 as MIQP solver. All available threads were used during the experiments.

2.6.1 Instances

The instance generation process begins by defining the reference quality point $\hat{q}^e$ for each chemical element as the midpoint between its minimum and maximum requirements when steel production is at its minimum (i.e., when $p = P$). To normalize these values, we use the parameter θ^e, which represents the difference between the maximum and minimum quantities under minimal production. Additionally, the liquid mold composition, λ^e, is defined as the mean composition across all steel types. The parameter δ is defined as 100 kg, while M_1 is set at 50000 kg and M_2 at 2000 kg.

A key aspect of the instance generation is the variety of steel types considered. The real-world chemical compositions of all steel types are crucial for ensuring both the applicability of the analysis and the accuracy of the model. To achieve this, we randomly select 10 different steel types from the standards defined in Bringas (2004). For each selected type, we generate 9 additional variants by randomly sampling the percentage of each chemical element within its allowed range defined by the standard. These variants, together with the original steel types, result in a total of 100 different steel types and chemical compositions for testing our model.

Overall, the benchmark instances should be regarded as semi-synthetic rather than purely historical datasets. The chemical composition ranges of the steel grades are derived from established industrial standards, ensuring that all generated steel types are realistic and practically feasible. The remaining instance characteristics, including the scrapyard configurations and their similarity and dispersion levels, are generated synthetically to enable a systematic analysis of different operational scenarios while remaining representative of realistic steelmaking conditions.

The primary focus of our analysis is on the scrapyard composition and its impact on the solution. We define three indicators to characterize the scrapyard: 1) the number $|S|$ of different scrap types allowed in the scrapyard, 2) the similarity (*Sim*) between the scrapyard and a target chemical composition, and 3) the internal chemical dispersion ($Disp$) of the scrapyard. We consider 3 values for each parameter, resulting in 27 unique scrapyard combinations. To mitigate sampling effects, we generate 10 scrapyards for each combination, producing a total of 270 scrapyards.

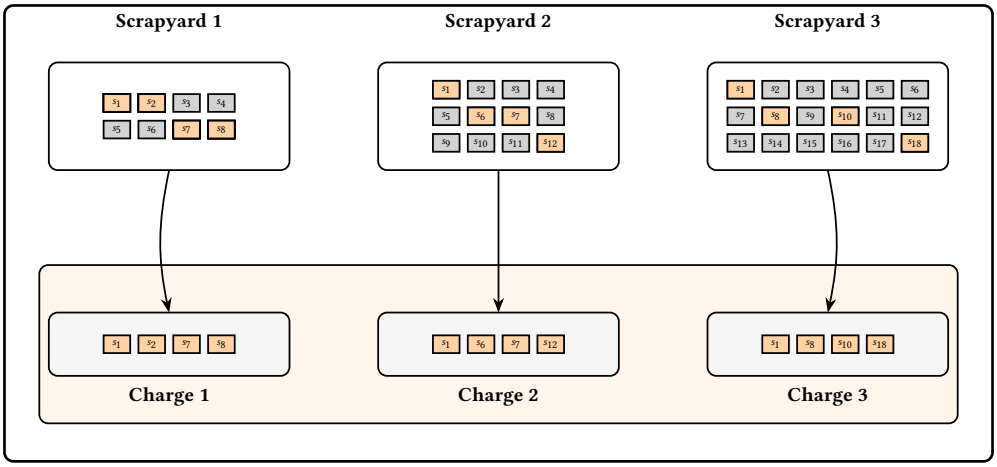
For scrapyard size, we consider values of 8, 12, and 18 scrap types. This allows us to assess how variations in scrapyard size influence production, which is particularly relevant in plants where handling large quantities of different scrap types is logistically challenging. Additionally, it provides insights into how raw material shortages affect production and how adjustments can mitigate these effects.

The parameter *Sim* measures how closely a scrapyard matches a target chemical composition. The target, defined by the decision maker, can vary depending on the context. In our study, the mean composition for steel types was used. Three classes are defined based on cosine similarity, which computes the cosine of the angle between two vectors (the target chemical composition and the scrapyard composition) as a metric to quantify their difference: low (L), medium (M), and high (H). These classes differ by 5%.

The parameter $Disp$ quantifies the chemical dispersion within the scrapyard. It is computed as the mean of the variance of each chemical element, normalized by its allowed chemical range. Three classes are identified: low (L), medium (M), and high (H).

Figure 2.1 presents a simple graphical example illustrating the behavior and practical application of the proposed model. Even under identical values of *Sim* and $Disp$, different scrap selections can be obtained, potentially leading to distinct trade-offs between steel quality and production cost.

Figure 2.1: Scrapyards charge example.



2.6.2 Trade-off results when f_P worsening is not allowed

In this set of experiments, we analyze the average cost and quality of a single casting across all combinations of scrapyards, parameters, and steel types under the constraint that the primary objective cannot deteriorate with respect to its optimal value.

		$\|S\| = 8$				$\|S\| = 12$				$\|S\| = 18$			
		Cost Components %			Total Cost	Cost Components %			Total Cost	Cost Components %			Total Cost
Sim	*Disp*	Fa	Sc	En	Avg [StDev]	Fa	Sc	En	Avg [StDev]	Fa	Sc	En	Avg [StDev]
L	L	17.9	73.2	9.3	637.34 [320.91]	3.4	86.1	10.5	491.95 [117.05]	4.0	85.1	11.0	471.43 [131.01]
L	M	29.2	63.2	8.8	779.07 [430.57]	9.2	81.1	9.8	550.29 [214.46]	5.3	84.3	10.4	519.50 [221.44]
L	H	22.5	69.4	11.3	722.01 [429.64]	12.1	78.6	9.5	576.66 [267.22]	7.6	81.4	11.1	493.37 [173.76]
	Avg:	23.2	68.6	9.8	**712.81 [393.71]**	8.2	82.0	10.0	**539.63 [199.58]**	5.7	83.6	10.8	**494.77 [175.40]**
M	L	10.2	79.6	10.4	521.60 [182.81]	5.8	83.7	10.6	502.78 [225.51]	5.0	84.0	11.1	459.66 [133.65]
M	M	15.3	74.9	11.8	565.55 [234.46]	5.6	83.9	11.5	488.76 [144.98]	2.9	85.6	11.7	438.99 [105.08]
M	H	19.0	72.3	18.8	642.22 [314.55]	6.3	82.8	12.1	484.82 [178.28]	2.1	87.0	11.3	458.53 [87.21]
	Avg:	14.9	75.6	13.7	**576.46 [243.94]**	5.9	83.5	11.4	**492.12 [182.92]**	3.3	85.5	11.4	**452.39 [108.65]**
H	L	6.5	83.0	10.7	481.64 [137.70]	4.6	83.2	12.7	443.1 [130.56]	3.0	84.7	12.4	427.41 [112.00]
H	M	4.4	84.5	12.8	474.49 [121.03]	5.1	83.5	13.0	452.59 [127.45]	4.3	83.4	12.4	423.69 [128.99]
H	H	13.2	76.7	12.9	525.39 [231.35]	7.9	80.8	12.9	472.85 [159.30]	4.8	82.9	12.4	424.32 [117.58]
	Avg:	8.0	81.4	12.1	**493.84 [163.36]**	5.9	82.5	12.9	**456.17 [139.10]**	4.1	83.7	12.4	**425.14 [119.52]**
Overall Avg:		15.4	75.2	11.9	**594.37 [267.00]**	6.7	82.6	11.4	**495.97 [173.87]**	4.3	84.3	11.5	**457.43 [134.52]**

Table 2.6: Costs distribution

Table 2.6 presents the results obtained from solving $(f_2|f_1, 0)$, where the total cost is divided into the contribution given by the ferroalloy cost (Fa), the scrap cost (Sc), and the energy cost (En). Several insights emerge from this cost assessment. When both scrapyard size and similarity are at their lowest, a high proportion of ferroalloys is used, leading to increased costs. This suggests that while smaller scrapyards may be logistically efficient, they can drive up overall production costs. Notably, the average amount of scrap used across all combinations remains consistent, with about 4 scrap types used per casting. Thus, costs are strongly influenced by the scrapyard composition. Regarding size, we can observe a cost decrease of more than 16% when increasing scrapyard size from 8 to 12, with an additional reduction of 7% from 12 to 18. These results highlight the significant impact of scrapyard size on production costs. However, determining the optimal size can be challenging due to

exogenous factors such as raw material availability, which can affect both the daily scrapyard size and costs. In terms of results stability when these exogenous events occur, the chemical dispersion of a scrapyard is more easily managed than its similarity, as the latter depends heavily on market and customer demands. The impact of the chemical dispersion *Disp* is less straightforward and appears to be correlated with the parameter *Sim*. The data suggest that mid-level dispersion minimizes costs, whereas low or high dispersion tends to increase them. Therefore, decision makers should aim to create scrapyards with balanced but not excessively sparse chemical distributions to reduce costs. Notably, energy consumption remains largely unaffected by scrapyard size, similarity, or chemical dispersion. This implies that energy costs related to scrap selection are relatively stable and do not exhibit strong dependencies on the defined scrapyard parameters.

Sim	*Disp*	$\|S\| = 8$	$\|S\| = 12$	$\|S\| = 18$
L	L	2.26	1.02	0.46
L	M	3.69	1.57	0.55
L	H	2.70	1.58	0.77
	Avg:	2.88	1.39	0.60
M	L	3.75	1.11	0.79
M	M	2.96	2.26	0.92
M	H	4.48	2.25	0.64
	Avg:	3.73	1.87	0.79
H	L	3.67	2.01	0.96
H	M	3.71	2.98	1.83
H	H	4.80	2.99	1.66
	Avg:	4.06	2.66	1.48
Overall Avg:		3.56	1.97	0.95

Table 2.7: Square deviation from $\hat{q}_e$

Table 2.7 presents the results of $(f_1|f_2, 0)$, i.e., the optimization of the final chemical quality as the primary objective. The results confirm previous trends, highlighting that an increase in scrapyard size positively impacts the final chemical quality across all parameter combinations. Specifically, increasing the size from 8 to 12 improves the chemical quality by approximately 44%, while further increasing it from 12 to 18 leads to an additional 51% improvement. A notable difference compared to cost minimization is that increasing *Sim* negatively affects the chemical quality. Moving from low to medium similarity leads to an average worsening of quality across all sizes of approximately 32%, while moving from medium to high results in a further reduction of about 46%. This highlights the importance of accurately estimating future production requirements to maintain fine control over chemical quality. *Disp* also plays a key role. Low dispersion enhances chemical quality, as chemically similar scraps make it easier to achieve target compositions. Conversely, high dispersion introduces greater variability, making it more challenging to meet exact chemical specifications. In summary, chemical quality is more sensitive to future production requirements (captured by *Sim*) but can be improved by maintaining low chemical dispersion (*Disp*) and larger scrapyard sizes.

2.6.3 Trade-off results when f_P worsening is allowed

Since our model includes two objective functions, understanding the interplay between them is crucial. Figures 2.2 and 2.3 illustrate how the objectives relate to each other under different percentage worsening of the prioritized objective function (reported on the horizontal axis). Figure 2.2 presents the results of $(f_1|f_2, w)$ for different values of w (0, 5, 10, 25, 50, 75, and

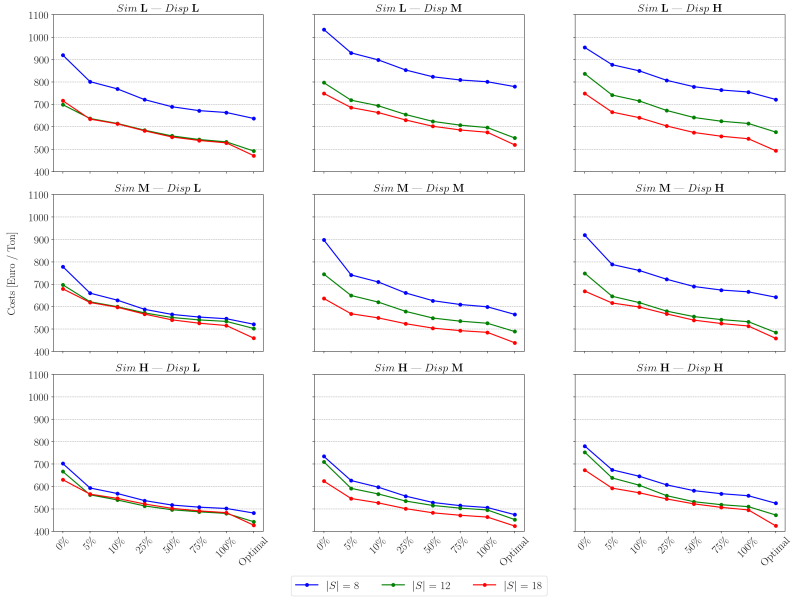


Figure 2.2: Costs evolution using $(f_1|f_2, w)$

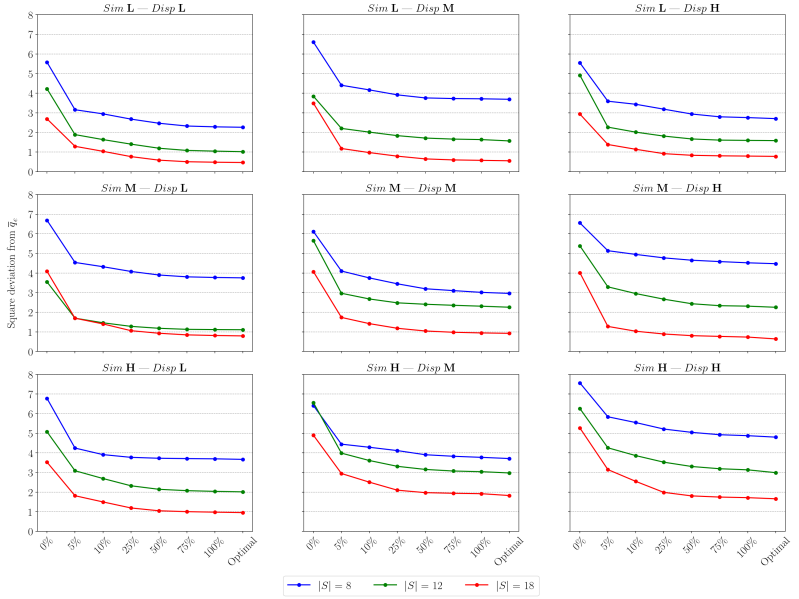


Figure 2.3: Square deviation evolution using $(f_2|f_1, w)$

100). The case $(f_2|f_1, 0)$ is also added with the label "Optimal". Figure 2.3 shows the results for $(f_2|f_1, w)$, with the same possible values of w, and as "Optimal" the case in which $(f_1|f_2, 0)$. Both figures display results for all 27 parameter combinations in a 3×3 grid. The columns represent increasing *Disp* (from left to right), while the rows represent increasing *Sim* (from

top to bottom). Within each plot, the three different sizes are analyzed. This approach allows us to evaluate how the worsening of one objective function impacts the other, helping decision makers identify strategies and promising trade-offs.

The analysis in Figure 2.2 shows that maintaining tight control over chemical quality comes at a significant cost, as expected. Specifically, even a small 5% deterioration in quality can reduce costs by approximately 100 €/ton. Increasing the scrapyard size lowers costs, with a linear decrease across all sizes. However, for certain parameter combinations, the cost difference between sizes 12 and 18 becomes negligible, a result that did not emerge while analyzing just one objective function at a time. This indicates that returns diminish as the scrapyard size increases, suggesting that the maximal efficiency could be achieved at a certain size beyond which further expansion is not cost-effective. It is important to note that these figures do not account for the logistical costs of managing larger scrapyards. A deeper analysis incorporating these factors could identify the true optimal size. It is also relevant to note that as *Sim* increases, size-related cost differences are reduced. Moreover, a small scrapyard consistently results in the highest costs across all combinations, while the cost difference between sizes 12 and 18 depends more on *Sim* and *Disp*.

In Figure 2.3, we can see that a modest 5% increase in costs leads to a significant quality improvement. However, beyond this point, improvement reaches a plateau or decreases slightly. As seen in the chemical quality analysis of Section 2.6.2, larger scrapyards consistently yield better results. Unlike the cost trends, there is always a measurable quality improvement when scrapyard size increases. Interestingly, quality improvements are not directly linked to the *Sim* or *Disp* parameters in isolation, but rather to their interaction.

2.7 Conclusions

In this chapter, we present a bi-objective optimization approach to the Cost-Quality Scrap Selection Optimization Problem in steel production. The problem minimizes costs associated with raw materials and energy consumption while ensuring a precise chemical quality of the steel through effective scrap selection. The developed mathematical model incorporates real-world constraints and parameters, offering a realistic and practical representation of the problem. This research serves as a valuable tool for decision makers in the steel industry, promoting more sustainable and cost-effective production processes.

The experimental analysis highlights the significant impact of scrapyard features, namely size, similarity, and chemical dispersion, on both cost and quality. Larger scrapyards consistently lead to cost reductions and improved chemical quality. However, results suggest the existence of a threshold beyond which further increases in scrapyard size yield diminishing returns, outweighed by the added complexity and maintenance requirements. High chemical dispersion should be avoided to reduce costs, but no clear recommendation emerges for chemical quality based solely on dispersion. While similarity helps reduce costs, it negatively affects chemical quality when optimized in isolation. We note that, in practice, managing high-quality scrapyards proves challenging, especially due to the difficulty of controlling their similarity over time.

Future research could further extend the proposed framework by relaxing some of the assumptions adopted to isolate the effect of scrapyard characteristics on cost and steel quality. In particular, the model considers a static decision setting in which scrap properties, energy consumption, and material availability are assumed to be known in advance. While this assumption allows a clear analysis of the trade-offs between production cost, chemical quality, and scrapyard composition, real steelmaking operations are affected by uncertainty

in scrap quality, fluctuating electricity prices, and changes in material availability over time. Incorporating these sources of uncertainty through stochastic or robust optimization techniques, would further improve the practical applicability of the proposed methodology. Additional research could also investigate the impact of supply chain disruptions and exploit the proposed framework to derive guidelines for designing scrapyards better tailored to the operational requirements of individual steelmaking plants.

Chapter 3

A Combinatorial Approach to Steelmaking Cost Reduction: Liquid Heel and Energy Considerations in Casting Sequences

Contents

3.1 Introduction . . . 23
3.2 Literature Review: Multi-Period Blending . . . 25
3.3 Problem description . . . 27
3.3.1 Modeling Assumptions . . . 29
3.4 Mathematical model . . . 30
3.4.1 Managing subtour in the Hamiltonian path . . . 34
3.4.2 Valid inequalities tightening the bounds on q_i^e . . . 34
3.5 Two-Step Constructive heuristic . . . 35
3.5.1 Ranking methods . . . 36
3.5.2 Aggregation methods . . . 39
3.5.3 Heuristic sorting . . . 39
3.5.4 Computing feasibility . . . 40
3.6 An ALNS Algorithm for the MHSSP-LH . . . 41
3.6.1 Introduction to ALNS . . . 41
3.6.2 The proposed ALNS Algorithm . . . 42
3.7 Experimental Results . . . 46
3.7.1 Instances description . . . 46
3.7.2 Numerical Tolerances and Precision Considerations . . . 48
3.7.3 Company Solution Definition . . . 49
3.7.4 Constructive Heuristic analysis . . . 50

3.7.5 Solving the complete model 59
3.7.6 ALNS Evaluation 63
3.8 Conclusions 73

This chapter introduces a novel optimization model for sequencing steel heats in an Electric Arc Furnace (EAF) environment. The main contribution of the proposed approach is the explicit modeling of the liquid heel, defined as the residual molten steel remaining in the furnace after tapping a heat, and its influence on subsequent heats. Although operationally significant, this aspect has not been thoroughly addressed in the existing literature. Its inclusion creates interdependencies between consecutive casting operations, affecting both feasibility conditions and production costs.

The model integrates key operational constraints representative of real industrial settings, including scrap availability, furnace capacity limits, and production requirements. In addition, energy consumption is explicitly incorporated into the cost structure, allowing the model to capture the interaction between scrap mix decisions, residual heel composition, and melting energy requirements.

The problem is formulated as the Multi-Heat Scrap Selection and Sequencing Problem with Liquid Heel (MHSSP-LH), whose objective is to minimize total production costs, including raw material and energy costs, over a sequence of heats.

A Mixed-Integer Linear Programming (MILP) formulation is developed to solve the proposed problem. The practical applicability of the model is assessed through a comprehensive computational study based on real industrial data. The optimized solutions are compared with the production plans currently implemented by the company, demonstrating the economic and operational improvements enabled by incorporating liquid heel dynamics into the decision-making process.

3.1 Introduction

Scrap-based steelmaking in the electric arc furnace (EAF) is not operated as a sequence of isolated heats, but rather as a tightly interconnected production process evolving over time. While single-heat scrap selection captures the static blending decision for one casting, real industrial environments require planning across multiple consecutive heats, often within a daily or weekly production horizon. In this setting, decisions taken at one heat directly influence the feasible and economic outcomes of subsequent heats. This temporal interdependence fundamentally transforms the structure of the optimization problem.

A key source of this coupling, rarely modeled explicitly in existing optimization formulations, is the *liquid heel*, i.e., the residual molten steel left in the furnace after tapping. Unlike classical blending models that assume a clean furnace at the start of each heat, the presence of a liquid heel introduces chemical and thermal memory into the system. The chemical composition of the heel becomes part of the initial state of the next heat, affecting both the feasibility and the amount of corrective additions required. Consequently, scrap selection decisions propagate forward in time: an economically attractive choice for one heat may generate additional costs or infeasibilities in later heats due to the heel effect. Beyond chemical propagation, multi-period planning must also account for dynamic resource constraints. Scrap availability evolves as inventory is consumed, procurement decisions may be bounded over the planning horizon, and operational restrictions such as furnace capacity and discrete loading constraints remain binding at each period.

From a modeling perspective, the transition from single-heat to multi-heat introduces several additional layers of complexity. First, the problem becomes inherently state-dependent, as the liquid heel acts as a state variable linking consecutive heats and transmitting chemical characteristics across time. Second, scrap inventory dynamics must be explicitly modeled, since available stocks evolve as materials are consumed, thereby restricting feasible selections in subsequent periods. Third, the cost structure becomes cumulative: total production cost must be evaluated over the entire planning horizon rather than heat-by-heat. Fourth, feasibility itself propagates across periods, as chemical deviations that are not properly controlled in one heat may affect the feasibility and cost-efficiency of future heats.

While single-heat blending has been extensively studied, significantly less attention has been devoted to models that jointly decide (i) the casting sequence and (ii) the scrap/ferroalloy mix of each heat while explicitly capturing the state dependency induced by the liquid heel. Existing approaches typically treat heats independently or incorporate inter-heat effects only by managing inventory. However, ignoring such dependencies can lead to policies that appear cost-efficient locally but may generate practically infeasible solutions.

Within this context, this chapter extends the single-heat framework to a multi-heat one and introduces the Multiple Heat Scrap Selection Problem with Liquid Heel management (MHSSP-LH). The problem is formulated over a finite number of chemically different heats. The objective is to minimize total production cost while ensuring that chemical specifications are satisfied at each heat and that the evolution of the liquid heel is explicitly tracked.

The methodological contributions of this chapter can be summarized as follows:

(i) A new MILP formulation that explicitly models the liquid heel as a state variable linking consecutive heats.

(ii) Integration of scrap inventory evolution over multiple heats.

(iii) Explicit modeling of energy consumption as a function of input material melting cost.

(iv) Four different constructive heuristics' method, that focus on feasibility.

(v) An ALNS implementation for solving hard instances.

(vi) Computational analysis quantifying the economic value of multi-heat optimization in the company setting.

(vii) Computational analysis quantifying the effectiveness of both constructive heuristics, the ALNS and the MILP performance solved with Gurobi.

By modelling liquid heel dynamics within a combinatorial multi-heat optimization structure, the proposed formulation moves beyond static blending and provides a mathematical model capable of capturing the true operational interdependencies of EAF steelmaking. The model offers strategic insights for production planning, scrap procurement, and furnace operation policies.

Extensive computational experiments conducted on company based instances demonstrate that explicitly modeling liquid heel propagation can generate significant cost improvements compared to other solutions.

The remainder of this chapter is organized as follows. Section 3.2 reviews the relevant literature on multi-heat blending. Section 3.3 formally defines the multi-heat problem and its assumptions. Section 3.4 presents the mixed-integer formulation of the MHSSP-LH. Section 3.5, defines how we developed four different constructive heuristics in order to obtain

a feasible solution for our problem. Section 3.6 explains the ALNS method with its destroy and repair operators, and how it has been applied to our problem. Section 3.7 reports computational experiments and managerial insights. Finally, Section 3.8 concludes and outlines directions for future research.

3.2 Literature Review: Multi-Period Blending

While Chapter 2 reviewed the *single-heat* blending problem—where each melt is optimized independently—industrial steelmaking operations are inherently sequential. In EAF environments, heats are produced in series, often sharing raw material inventories, logistical resources, and process constraints. Consequently, decisions taken in one heat influence the feasibility and cost structure of subsequent heats. This interdependence motivates the study of *multi-heat* blending models, in which the optimization horizon spans several consecutive melts.

It is important, however, to distinguish between models that operate in multi-heat production environments and those that are structurally multi-heat in their mathematical formulation. In many industrial applications, optimization models are applied repeatedly over a sequence of heats within a day. Yet, the underlying mathematical problem may still optimize one heat at a time, without explicitly incorporating inter-heat coupling inside a unified formulation. In contrast, true multi-heat optimization simultaneously determines decisions for several heats and explicitly captures dependency between them.

Early modeling of multi-heat scrap charge planning can be traced to medium-term planning models in steel production. In contrast to single-heat models that minimize the cost of a single charge subject to chemical constraints, multi-heat models allocate limited scrap inventories across multiple heats to minimize total cost over a planning horizon. A representative discussion of this distinction is provided by Rong and Lahdelma (2008), where single-heat, multi-heat, and long-term recipe models are clearly differentiated. The multi-heat formulation addresses the risk that minimizing cost in the current heat may exhaust high-quality or low-cost scrap, thereby increasing future costs. This inventory-coupled structure introduces inter-temporal trade-offs that are ignored in single-heat models.

Large-scale industrial implementations of multi-period planning in steelmaking further demonstrate the relevance of horizon-wide optimization. The work of Kim and Lewis (1987) highlighted the economic impact of systematic least-cost charging when scaled across many melts. Although embedded in daily production involving multiple heats, the underlying linear programming formulation determines the optimal charge for one melt at a time, with inventory updates occurring between runs. Consequently, while the system operates in a multi-heat environment, it does not constitute a structurally multi-heat optimization model in which several heats are jointly optimized within a single mathematical program. This distinction is essential, as repeated single-heat optimization should not be confused with integrated multi-heat decision-making.

A similar observation applies to blending modeling studies in process industries such as Ashayeri et al. (1994), where emphasis is placed on data management, implementation, and organizational embedding of blending systems. Although these studies acknowledge operational continuity, they do not introduce explicit mathematical coupling between consecutive heats. A more explicit multi-heat perspective is found in scrap combination models for steel production such as Bernatzki et al. (1998). In that study, the authors formulate a mixed-integer programming (MIP) model to determine minimum-cost scrap combinations for consecutive converter processes, subject to metallurgical and transportation constraints.

Scrap is transported in discrete units (railway cars), and constraints on track capacity and sequencing directly couple consecutive heats. The formulation illustrates two key features of structurally multi-heat blending: (i) discrete and logistical constraints become binding when multiple charges are considered jointly; and (ii) the feasible region depends not only on chemistry but also on dynamic material availability across time.

The integration of scrap charge decisions with production scheduling has been further explored in Yang et al. (2018), where robust optimization is used to address uncertainty in metal element concentrations while simultaneously considering production scheduling under time-of-use electricity tariffs. The model links scrap charge composition, furnace assignment, and temporal electricity pricing within a unified framework. From a multi-heat standpoint, this contribution is significant because scrap allocation and heat scheduling decisions are optimized jointly, introducing explicit temporal interdependence within the model.

Scheduling-oriented multi-period blending formulations have also been developed in adjacent metallurgical contexts. For example, Kim et al. (2008) propose a mixed-integer programming model for multiperiod blending in hydrometallurgical zinc production, where minimizing setup changes in blending ratios over time is critical for process stability. Although the application differs from steel scrap charging, the structural analogy is clear: maintaining consistent blending proportions across periods introduces temporal coupling beyond isolated batch optimization.

The operational integration of blending and information systems has long been recognized as a prerequisite for effective multi-heat optimization. The online mixture calculation system described in Sutton and Coates (1981) demonstrates how real-time data on stock levels, material arrivals, and production requirements can be embedded in a least-throughput-cost mix system. While such systems manage multiple heats operationally, the optimization core typically remains heat-by-heat unless explicit inter-heat state variables are introduced.

Despite these advances, most existing multi-heat models share certain structural characteristics. First, heats are generally coupled through external resources such as inventory levels, logistics, or scheduling constraints. Second, the internal metallurgical state of the furnace is typically assumed to reset between heats. That is, while resource consumption links heats economically, the chemical and thermodynamic state of the furnace is not explicitly modeled as evolving across consecutive melts. From the perspective of this dissertation, this omission is structurally significant. In practical EAF operations, a residual quantity of molten steel, referred to as the *liquid heel*, remains in the furnace after tapping. The composition and thermal state of this liquid heel directly influence the next heat. Consequently, heat $t + 1$ does not start from a neutral initial state but from a state determined by the outcome of heat t. True multi-heat optimization must therefore account not only for shared inventories or scheduling interactions, but also for state dependency between consecutive heats. The presence of a liquid heel introduces deterministic carry-over effects that create path-dependency in scrap selection and feasibility conditions. Under such circumstances, optimizing each heat independently, even within a daily production context, cannot capture the full economic and feasibility implications of sequential production.

In summary, the literature on multi-heat blending extends deterministic single-heat models primarily through: (i) inventory allocation across multiple heats, (ii) integration with discrete logistics and transportation constraints, and (iii) coupling with production scheduling and time-dependent energy costs. However, the majority of these contributions treat heats as chemically independent units linked only through external resource constraints. The explicit modeling of liquid heel dynamics introduces a fundamentally different form of inter-heat dependency based on physical state continuity inside the furnace.

Table 3.1: Comparison of optimization-based multi-heat scrap charging models.

Work	Structurally Multi-Heat	Inventory Management	Scheduling Integration	Uncertainty Handling	Liquid Heel Modeling
Kim and Lewis (1987)	○	◐	○	○	○
Ashayeri et al. (1994)	○	◐	○	○	○
Bernatzki et al. (1998)	●	●	◐	○	○
Rong and Lahdelma (2008)	●	●	○	●	○
Kim et al. (2008)	●	●	●	○	○
Yang et al. (2018)	●	●	●	●	○
This work	●	●	○	○	●

Notation: ● = fully addressed; ◐ = partially addressed; ○ = not addressed.

Table 3.1 summarizes the distinguishing characteristics of multi-heat blending models. While several contributions jointly optimize multiple heats within a unified mathematical formulation, none explicitly incorporates liquid heel dynamics into the optimization model. This observation highlights a clear gap in the existing literature and motivates the framework developed in this chapter.

Building on this gap, the proposed model introduces a multi-heat optimization framework in which the liquid heel is represented as a dynamic state variable linking consecutive heats. Unlike previous formulations, where inter-heat dependencies arise from shared resources such as inventories or production schedules, the proposed approach captures the physical continuity of the steelmaking process. Consequently, the optimization accounts for how the metallurgical and thermal state of one heat influences the feasible operating conditions, scrap selection, alloy additions, and energy requirements of subsequent heats. To the best of the author's knowledge, this is the first multi-heat scrap blending model to explicitly represent liquid heel dynamics within the optimization framework.

The introduction of state-dependent coupling substantially increases the complexity of the optimization problem. To efficiently solve realistically sized industrial instances, the chapter therefore complements the mathematical formulation with tailored constructive heuristics and an Adaptive Large Neighborhood Search (ALNS) algorithm, providing high-quality solutions while preserving the operational characteristics of EAF steelmaking.

3.3 Problem description

Let H denote the set of heats to be scheduled. The sequencing decision is represented by a directed graph $G = (V, A)$ with node set $V = H \cup \{b, t\}$, where b and t are dummy nodes. The source node b provides the initial furnace state, i.e., the initial liquid heel before the first scheduled heat, whereas the sink node t is introduced to store the final heel state after the last scheduled heat. Accordingly, arcs are allowed only out of b and only into t, while all transitions between distinct heats are acceptable; formally:

$$A = \{(b, i) : i \in H\} \cup \{(i, j) : i, j \in H,\ i \neq j\} \cup \{(i, t) : i \in H\}.$$

The liquid-heel weight is assumed constant and equal to ϕ for all heats. In contrast, the chemical composition of the heel is known only at the starting node b (initial furnace state) and is otherwise endogenously determined by the sequence: the heel transferred to the next heat is chemically equivalent to the molten steel produced in the preceding heat.

Regarding the input materials, let S denote the set of scrap types and K the set of ferroalloys. Let E denote the set of chemical elements. For each heat $i \in H$ and element $e \in E$, the target steel grade imposes lower and upper bounds on the final composition, denoted by $\underline{\beta}_i^e$

and $\overline{\beta}_i^e$ (expressed as mass fractions or percentages). Each scrap type $s \in S$ is characterized by its chemical composition $\alpha_{s,e}^S$ for all $e \in E$, its unit cost ζ_s^S, its yield coefficient μ_s^S, and its specific heat-of-fusion parameter ν_s^S. Analogously, each ferroalloy $k \in K$ is characterized by its parameters $\alpha_{k,e}^K$, ζ_k^K, μ_k^K, and ν_k^K.

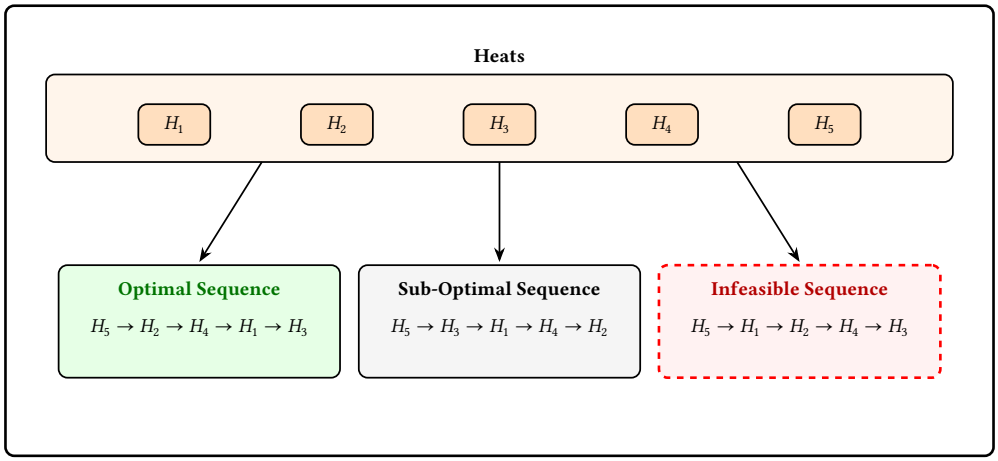
Together with the energy price γ, these parameters determine the cost components associated with charging and melting the input materials. For each heat $i \in H$, the target liquid-steel output is given and denoted by ψ_i. Hence, the charged quantities must be adjusted by the corresponding yield coefficients to ensure that the required amount of liquid steel is produced. For instance, the effective liquid contribution of scrap $s \in S$ is μ_s^S times the charged mass. Since all materials are charged into an EAF, the total input mass in each heat is constrained by the furnace maximum capacity λ, which must not be exceeded to ensure operational feasibility and operators' safety.

To ensure practical implementability, the problem accounts not only for chemical specifications but also for additional operational constraints. First, material availability is limited by plant inventories, which bound the total amount of each input material that can be used across the scheduled heats. We indicate as ω_s^S and ω_k^K, the available supply (stock) of each scrap type $s \in S$ and each ferroalloy $k \in K$, respectively. Second, charging decisions must reflect handling limitations: scrap quantities are typically moved and loaded in discrete batches, whereas ferroalloys are dosed with higher precision using dedicated equipment. Accordingly, we introduce a discretization (handling) parameter Δ that represents the minimum manageable scrap quantity, while no such discretization is imposed on ferroalloys.

The problem objective is to minimize the total production cost over all heats, defined as the sum of the purchasing costs of the charged scrap and ferroalloys and the corresponding energy costs required to melt these materials in the EAF.

In Tables 3.2–3.3 are summarized sets and parameters used to define the problem.

Figure 3.1: Problem sequencing example.



A simple illustrative example is presented in Figure 3.1. The figure demonstrates that the feasibility of a production sequence is determined not only by the set of heats to be processed, but primarily by their relative ordering. Different permutations of the same heats

may therefore result in feasible, sub-optimal, or infeasible schedules due to the influence of the liquid heel composition on consecutive heats.

The optimal sequence, $H_5 \rightarrow H_2 \rightarrow H_4 \rightarrow H_1 \rightarrow H_3$, satisfies all process compatibility requirements while maintaining a consistent progression of steel grades and liquid heel compositions throughout the sequence. Conversely, the infeasible sequence, $H_5 \rightarrow H_1 \rightarrow H_2 \rightarrow H_4 \rightarrow H_3$, violates these transition constraints through the subsequence $H_5 \rightarrow H_1$. In this case, the residual liquid heel composition associated with H_5 is incompatible with the metallurgical requirements of H_1, rendering the transition infeasible. Consequently, although both sequences contain the same set of heats, only the former can be executed in practice.

Table 3.2: Sets

H	Set of heats
S	Set of scrap types
K	Set of ferroalloys
E	Set of chemical elements
$V = H \cup \{b, t\}$	Set of all the nodes
A	Set of directed arcs between nodes

Table 3.3: Indices

b	Starting node for the sequence of heats
t	Terminal node for the sequence of heats

Table 3.4: Parameters

ϕ	Liquid foot weight
ψ_i	Target weight of heat i
λ	Maximum EAF capacity
ω_s^S	Total inventory of scrap s
ω_k^K	Total inventory of ferroalloy k
$\underline{\beta_i^e}$, $\overline{\beta_i^e}$	Min/Max concentration of element e in heat i
a_{se}^S, a_{ke}^K	Fraction of element e in scrap s or ferroalloy k
μ_s^S, μ_k^K	Yield of scrap s or ferroalloy k
δ	Minimum manageable material quantity
ζ_s^S, ζ_k^K	Cost per kg of scrap s and ferroalloy k
ν_s^S, ν_k^K	Heat of fusion coefficient for scrap s or ferroalloy k
γ	Energy cost coefficient
M_i^e	Large positive constant

3.3.1 Modeling Assumptions

Given the realistic settings of this problem, the mathematical model presented in this chapter relies on a set of assumptions that balance industrial realism with computational tractability. Although steel production is inherently affected by uncertainty, several operational aspects are represented through deterministic approximations in order to focus on the integration of multi-period planning and liquid heel management. The principal assumptions adopted by the model are summarized below together with their motivation.

Deterministic material properties. Each raw material is assumed to have a fixed and exactly known chemical composition, unit cost, and specific energy consumption throughout the planning horizon. In practice, these quantities may vary due to supplier variability, contamination, measurement uncertainty, and operational conditions. Such variability is neglected because the objective of this chapter is to study the interaction between multi-heat scrap selection and liquid heel management rather than uncertainty in material properties.

Deterministic production plan. The sequence of heats, production requirements, and target steel compositions are assumed to be known before the optimization process and remain unchanged throughout the planning horizon. This assumption reflects the offline tactical production planning stage considered in this work, where production orders are established before the optimization model is executed. No disruption, as in addition or removal of heats, in real time is considered in this case.

Deterministic liquid heel behavior. The quantity and chemical composition of the liquid heel transferred between consecutive heats are assumed to be exactly known. Although industrial operations may experience deviations due to process variability or

measurement inaccuracies, this assumption allows the proposed model to explicitly capture the physical coupling between consecutive heats while maintaining mathematical tractability.

Deterministic energy price. The electricity price is assumed to remain constant over the planning horizon. Although electricity prices may fluctuate over time, this simplification is reasonable in order to avoid the introduction of additional uncertainty that is outside the scope of this work.

These assumptions are consistent with the scope of the proposed model, which focuses on capturing the operational interaction between consecutive heats through liquid heel while maintaining a tractable optimization framework. Although uncertainty is not explicitly modeled, the proposed formulation provides a realistic representation of the production planning process, under the above defined assumptions. Future research may extend this framework by incorporating stochastic or robust optimization techniques to account any of the uncertainty properties.

The proposed framework can be embedded in a dynamic rescheduling scheme, in which the structure is repeatedly updated as soon as new information regarding system disruption are detected.

3.4 Mathematical model

In this section, we present a mixed-integer linear programming (MILP) formulation for the Multi-Heat Scrap Selection and Sequencing Problem with Liquid Heel (MHSSP-LH). To this end, we introduce the following sets of decision variables that determine (i) the casting sequence, (ii) the charge composition in each heat, and (iii) the inter-heat chemical propagation induced by the liquid heel.

Binary variables z_{ij} identify the sequencing decisions and take the value 1 if node $i \in V$ is processed immediately before node $j \in V$, with $(i, j) \in A$, and 0 otherwise.

For each heat $i \in H$, continuous variables y_{ki} denote the charged quantity of ferroalloy $k \in K$. To account for charging discretization of scrap, integer variables x_{si} are introduced that indicate the number of minimum material lots introduced in the furnace for each heat $i \in H$ and each scrap $s \in S$, where δ represents the minimum lot, expressed as a minimum manageable scrap batch size (no discretization is imposed on ferroalloys).

For each heat $i \in H$ and element $e \in E$, variable q_i^e denotes the amount of element e introduced through charged materials, whereas l_i^e denotes the total amount of element e available in the molten steel during heat i, including the contribution of the liquid heel. Finally, for each arc $(i, j) \in A$ and element $e \in E$, variable p_{ij}^e denotes the amount of element e transferred as liquid heel from node i to its immediate successor node j. A summary of the decision variables is reported in Table 3.5.

The resulting MILP formulation is given below.

Table 3.5: Decision variables summary

z_{ij}	Binary variable equal to 1 if node $i \in V$ precedes node $j \in V$, $(i,j) \in A$, 0 otherwise
x_{si}	Integer number of minimum lots of δ quantity of scrap type $s \in S$ charged in heat $i \in H$
y_{ki}	Continuous quantity of ferroalloy $k \in K$ charged in heat $i \in H$
q_i^e	Continuous amount of element $e \in E$ introduced in heat $i \in H$ from charged materials
l_i^e	Continuous total amount of element $e \in E$ available in heat $i \in H$ (including liquid heel)
p_{ij}^e	Continuous amount of element $e \in E$ transferred as liquid heel from heat i to heat j, $(i,j) \in A$
a_s^S	Continuous additional purchased quantity of scrap type $s \in S$
a_k^K	Continuous additional purchased quantity of ferroalloy type $k \in K$

$$\min \quad \sum_{i \in H} \left(\sum_{s \in S} \left(\zeta_s^S + v_s^S \gamma \right) \delta x_{si} + \sum_{k \in K} \left(\zeta_k^K + v_k^K \gamma \right) y_{ki} \right) \tag{3.1}$$

$$\text{s.t.} \quad \sum_{(b,j) \in A} z_{bj} = 1 \tag{3.2}$$

$$\sum_{(j,t) \in A} z_{jt} = 1 \tag{3.3}$$

$$\sum_{(j,i) \in A} z_{ji} = 1 \qquad i \in H \tag{3.4}$$

$$\sum_{(i,j) \in A} z_{ij} = 1 \qquad i \in H \tag{3.5}$$

$$\sum_{s \in S} \delta x_{si} + \sum_{k \in K} y_{ki} \leq \lambda \qquad i \in H \tag{3.6}$$

$$\sum_{s \in S} \mu_s^S \, \delta x_{si} + \sum_{k \in K} \mu_k^K \, y_{ki} = \psi_i \qquad i \in H \tag{3.7}$$

$$q_i^e = \sum_{s \in S} \alpha_{se}^S \, \delta x_{si} + \sum_{k \in K} \alpha_{ke}^K \, y_{ki} \qquad i \in H, \; e \in E \tag{3.8}$$

$$l_i^e = q_i^e + \sum_{(j,i) \in A} p_{ji}^e \qquad i \in H, \; e \in E \tag{3.9}$$

$$\sum_{(i,j) \in A} p_{ij}^e = \theta_i \, l_i^e \qquad i \in H, \; e \in E \tag{3.10}$$

$$p_{bj}^e = p_0^e z_{bj} \qquad j \in H, e \in E \tag{3.11}$$

$$\theta_i \, l_i^e - M_i^e \, (1 - z_{ij}) \leq p_{ij}^e \leq M_i^e \, z_{ij} \qquad (i,j) \in A, \; e \in E \tag{3.12}$$

$$\underline{\beta}_i^e \, (\psi_i + \phi) \;\leq\; l_i^e \;\leq\; \overline{\beta}_i^e \, (\psi_i + \phi) \qquad i \in H, \; e \in E \tag{3.13}$$

$$\sum_{i \in H} \delta x_{si} \;\leq\; \omega_s^S + a_s^S \qquad s \in S \tag{3.14}$$

$$\sum_{i \in H} y_{ki} \;\leq\; \omega_k^K + a_k^K \qquad k \in K \tag{3.15}$$

$$z_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{3.16}$$

$$x_{si} \geq 0, \qquad s \in S,\ i \in H \tag{3.17}$$

$$y_{ki} \geq 0, \qquad k \in K,\ i \in H \tag{3.18}$$

$$q_i^e \geq 0,\ l_i^e \geq 0 \qquad i \in H,\ e \in E \tag{3.19}$$

$$p_{ij}^e \geq 0 \qquad (i,j) \in A,\ e \in E \tag{3.20}$$

$$a_s^S \geq 0 \qquad s \in S \tag{3.21}$$

$$a_k^K \geq 0 \qquad k \in K \tag{3.22}$$

The objective minimizes the total production cost over all heats. For each heat $i \in H$, the term $\sum_{s\in S} \left(\zeta_s^S + v_s^S \gamma\right) \delta x_{si}$ accounts for the cost of charging scrap, where ζ_s^S is the unit purchasing cost of scrap type s and $v_s^S \gamma$ represents the corresponding energy-related melting cost per unit mass (with v_s^S the material-specific melting parameter and γ the energy price). Analogously, $\sum_{k\in K} \left(\zeta_k^K + v_k^K \gamma\right) y_{ki}$ captures the purchasing and melting costs associated with ferroalloys. Constraint (3.2) enforces that exactly one arc leaves the dummy source node b, thereby selecting the first heat in the sequence. Constraint (3.3) enforces that exactly one arc enters the dummy sink node t, thereby selecting the last heat. Constraints (3.4)–(3.5) impose a degree-1 structure on the heat nodes: each heat has exactly one predecessor and exactly one successor in the selected sequence. Together, these constraints define a Hamiltonian path from b to t over the heats. Constraint (3.6) imposes the safety margin, related to the maximum amount of raw material placed in the EAF for a single heat that should not exceed the maximum capacity λ. Constraint (3.7) imposes the liquid-steel output requirement for each heat $i \in H$. The left-hand side represents the effective liquid contribution of the charged materials, obtained by scaling the charged masses δx_{si} and y_{ki} by their yield coefficients μ_s^S and μ_k^K, respectively. The equality enforces that the resulting liquid output matches the prescribed target ψ_i. Constraint (3.8) defines q_i^e as the amount of element $e \in E$ introduced into heat $i \in H$ through charged materials. Specifically, α_{se}^S and α_{ke}^K denote the mass fraction of element e in scrap type s and ferroalloy k, respectively, so that the right-hand side aggregates the elemental contributions of all charged materials. Constraints (3.9) defines the total amount l_i^e of element e available in the molten bath during heat i as the sum of the amount q_i^e introduced through charged materials in heat i and the elemental contribution carried by the liquid heel transferred from the immediate predecessor node (captured through p_{ji}^e on the selected incoming arc). Constraints (3.10) state that the total amount of element e transferred from heat i to its successor (as liquid heel) equals the retained fraction θ_i of the molten-bath content l_i^e. For each heat i and element e, the amount of element e that remains in the furnace as liquid heel after completing heat i is assumed to be proportional to the bath composition of heat i. Thus, we define $\theta_i := \frac{\phi}{\psi_i + \phi}$, representing the fraction of the molten bath that remains in the furnace after tapping heat i. For each arc $(i,j) \in A$ and element $e \in E$, heel-transfer variables are linked to the sequencing decisions by constraints (3.12). The right-hand inequality activates p_{ij}^e only on selected arcs: if $z_{ij} = 0$, it enforces $p_{ij}^e = 0$. The left-hand inequality is a conditional lower bound: if $z_{ij} = 1$, it yields $p_{ij}^e \geq \theta_i l_i^e$, i.e., the selected successor must receive at least the retained fraction of element e from heat i. Combined with the conservation constraint $\sum_{(i,j)\in A} p_{ij}^e = \theta_i l_i^e$, this implies that the entire heel outflow of heat i is assigned to its (unique) successor, thereby preventing fractional splitting across multiple outgoing arcs in the linear relaxation. A tight choice for the constant M_i^e is $M_i^e = \overline{\beta}_i^e \phi$. Constraint (3.11) imposes that the starting liquid heel value for each chemical element is known and defined by the parameter p_0^e. This value is associated only with the

liquid heel going from source b to the first heat; in all the other cases, the transferred liquid heel is zero.

Constraint (3.13) defines the chemical composition requirements for each heat. For every heat $i \in H$ and element $e \in E$, the total amount l_i^e of element e in the molten steel must remain within the admissible lower and upper bounds. These bounds are computed as a fraction $\underline{\beta}_i^e$ and $\overline{\beta}_i^e$ of the total mass $\psi_i + \phi$, which includes both the input material and the liquid heel already present in the furnace. Therefore, this constraint guarantees that the overall liquid steel composition, satisfies the prescribed metallurgical specification.

Constraints (3.14) and (3.15) define the amount of inventory present in the system for each type of raw material, scraps or ferroalloys, support variable a_s^S and a_k^K, defines how much new material is needed, for the complete creation of all the necessary heats.

Figure 3.2: Graph based explanation of the liquid heel.

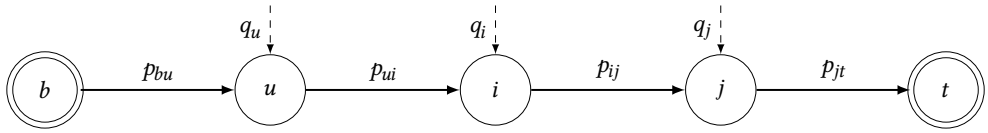
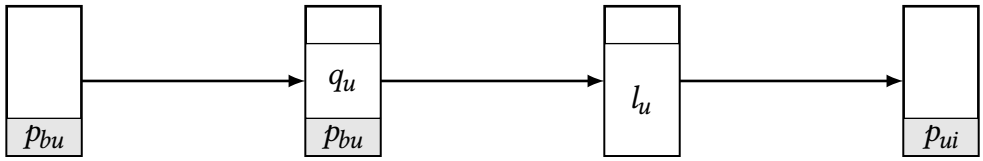


Figure 3.3: Schematics of resource flow from node b to node u.



To better illustrate the behavior of the model, a simplified example of a solution is presented in Figure 3.2 and Figure 3.3. At the beginning of the production sequence, only the composition of the initial liquid heel is known. When a heat corresponding to node u is processed, a new quantity of raw materials q_u is loaded into the furnace. However, the resulting chemical composition l_u of the molten steel must account not only for the newly introduced materials, but also for the contribution of the liquid heel inherited from the predecessor node b, modelled by p_0.

If the resulting composition satisfies the required chemical bounds for the target steel grade, a fixed amount ϕ of molten steel remains in the furnace as the new liquid heel. The chemical content of this heel, denoted p_{ui}, is proportional to the total composition of heat u, scaled according to the ratio θ_u between the known heel weight ϕ and the production quantity ψ_u. This propagated composition then becomes an input for the subsequent heat. The same mechanism is repeated sequentially until the final heat in the schedule is completed.

3.4.1 Managing subtour in the Hamiltonian path

The proposed model relies on the construction, over the graph G, of a Hamiltonian path from b to t. Such a structure cannot be enforced solely through the 1-degree constraints introduced above, as they do not prevent the creation of disconnected cycles. This motivates the introduction of appropriate subtour elimination constraints (SEC). Several families of SEC have been proposed in the literature, differing in compactness, polyhedral strength, and computational impact.

We first consider the classical subtour elimination constraints (SEC). Formally, for any subset $S \subset V$ such that $b, t \notin S$ and $2 \leq |S| \leq |V| - 2$, the following inequality is imposed:

$$\sum_{i \in S} \sum_{j \in S} z_{ij} \leq |S| - 1.$$

These constraints forbid the existence of a cycle entirely contained within S, ensuring that no proper subset of vertices induces a closed tour. The SECs are exponential. Consequently, they are typically enforced through separation procedures embedded in the solution process. This implies that such constraints are dynamically generated and added during the optimization. We introduced them as lazy constraints, checking them only for integer solutions found during the branch-and-bound process.

An equivalent representation of inequalities is given by the connectivity constraints (CC). For any subset $S \subset V$ with $b, t \notin S$, they are expressed as

$$\sum_{i \in S, , j \notin S} z_{ij} \geq 1.$$

These constraints guarantee that each subset of nodes remains connected to the rest of the graph, thereby preventing the isolation of subtours. Similarly to SEC inequalities, CC constraints define an exponential family and are therefore handled via separation. A key advantage of this formulation is that violated inequalities can be identified through a max-flow/min-cut computation, for which efficient polynomial-time algorithms are available. In our case, violated inequalities are detected using the algorithm proposed by Boykov and Kolmogorov (2004). Moreover, this class of inequalities can be separated not only over integer solutions but also over fractional ones. As a consequence, they are introduced both as cutting planes, when they eliminate fractional solutions at intermediate nodes of the search tree, and as lazy constraints when they exclude infeasible integer solutions. These constraints are dynamically added throughout the solution process.

3.4.2 Valid inequalities tightening the bounds on q_i^e

In the proposed model, the elemental content in heat i satisfies the mass-balance relation

$$l_i^e \;=\; q_i^e \;+\; \sum_{(j,i) \in A} p_{ji}^e \qquad \forall i \in H,\ e \in E,$$

where l_i^e is the total amount of element e in heat i, q_i^e is the contribution coming from newly charged materials, and p_{ji}^e denotes the amount of element e carried by the retained heel transferred from predecessor j to i. Moreover, due to quality requirements we have

$$\underline{\beta}_i^e(\psi_i + \phi) \;\leq\; l_i^e \;\leq\; \overline{\beta}_i^e(\psi_i + \phi) \qquad \forall i \in H,\ e \in E.$$

Since the initial state of the furnace is known, the elemental composition of the initial heel is a given constant p_0^e for each element $e \in E$, and it is injected into the first heat through the arc (b, i) (with binary variable z_{bi}).

Combining the mass-balance with the upper/lower bounds on l_i^e and with the implied bounds on heel transfers

$$\underline{\beta}_j^e \phi \, z_{ji} \;\le\; p_{ji}^e \;\le\; \overline{\beta}_j^e \phi \, z_{ji} \qquad \forall (j,i) \in A,\; j \in H,\; e \in E,$$

we obtain the following valid (strengthening) inequalities for q_i^e:

$$q_i^e \;\ge\; \underline{\beta}_i^e(\psi_i + \phi) \;-\; p_0^e \, z_{bi} \;-\; \sum_{j \in H \setminus \{i\}} \overline{\beta}_j^e \, \phi \, z_{ji} \qquad \forall i \in H,\; e \in E, \tag{3.23}$$

$$q_i^e \;\le\; \overline{\beta}_i^e(\psi_i + \phi) \;-\; p_0^e \, z_{bi} \;-\; \sum_{j \in H \setminus \{i\}} \underline{\beta}_j^e \, \phi \, z_{ji} \qquad \forall i \in H,\; e \in E. \tag{3.24}$$

Inequalities (3.23)–(3.24) tighten the LP relaxation by bounding the amount of element e that must (or can) come from newly charged materials in heat i. Intuitively, if a predecessor j is selected (i.e., $z_{ji} = 1$), then the retained heel brings at most $\overline{\beta}_j^e \phi$ and at least $\underline{\beta}_j^e \phi$ units of element e, thereby reducing the feasible range of q_i^e. The same applies to the first heat through the known initial heel contribution p_0^e, activated by z_{bi}.

3.5 Two-Step Constructive heuristic

Given the difficulty of the problem, especially as the instance size increases, we first define some constructive heuristics, that can be used as both full problem solutions and, if necessary, as starting solutions for other metaheuristic methods that need a feasible start or as a warm start for the MILP solver. The main purpose is to obtain, at negligible computational cost, one or more feasible solutions. In practice, providing a feasible initial solution is often decisive to (i) reduce the time needed to reach feasibility, (ii) improve the primal bound early in the search, (iii) stabilize the solver behavior on large-scale instances, and (iv) obtain a starting solution for more complex methods. We propose four variants of a constructive 2-step procedure.

Step 1: construction of candidate Hamiltonian paths. We developed four alternative sorting procedures that rank the heats by exploiting their chemical composition ranges (element by element) and then aggregate these element-wise rankings into a single global ordering. Each global ordering induces a candidate Hamiltonian path from the source node b to the sink node t. The four variants differ in (i) how the element-wise ranks are computed and (ii) how the ranks are aggregated across elements.

Step 2: feasibility assessment through a path-constrained MILP. A purely sorting-based ordering does not guarantee feasibility for the full model. The reason is that feasibility depends not only on the existence of a Hamiltonian path, but also on the additional constraints (e.g., mass balance, chemical constraints, resource limits, and liquid heel influence) that are not explicitly enforced by sorting alone. Therefore, we evaluate each candidate ordering by solving a restricted MILP in which the routing decisions are fixed to follow

the heuristic Hamiltonian path. Concretely, arcs that are not part of the heuristic path are fixed to be zero, and only the arcs consistent with that sequence are active. This restriction transforms the original combinatorial structure into a much simpler problem, enabling a fast feasibility check.

If the restricted model is feasible, the corresponding solution is a feasible solution for the original model (since it satisfies all original constraints while using a fixed path). In that case, we retain it as a candidate feasible solution. Among all feasible restricted solutions obtained from the four heuristic variants, we can also select the one with the best objective value and provide it as the best constructive heuristic.

3.5.1 Ranking methods

The first step of the heuristic requires defining how heats are ranked based on their chemical ranges. For each heat $i \in H$ and chemical element $e \in E$, we assume the admissible chemical content lies within the interval

$$\underline{\beta_i^e} \leq \beta_i^e \leq \overline{\beta_i^e},$$

where $\underline{\beta_i^e}$ and $\overline{\beta_i^e}$ denote the minimum and maximum allowed concentration, respectively. Additionally, each heat is associated with a production weight ψ_i.

The key idea is to sort heats firstly based on their lowest chemical bound ($\underline{\beta_i^e}$) then in case of equal minimum, to use the maximal chemical bound ($\overline{\beta_i^e}$), and then if both values are the same, use the weight (ψ_i). The underlying principle is to order heats from the most restrictive to the least restrictive chemical interval. This was done to reduce the risk of having infeasibility generated by the liquid heel transfer from one heat to the next.

To define the precedence relation between heat i and heat j for element e, we describe the following *dominance* criteria. Heat i *dominates* heat j if $\underline{\beta_i^e} < \underline{\beta_j^e}$.In case of a tie on the lower bound, we compare the upper bounds: heat i *dominates* heat j if $\underline{\beta_i^e} = \underline{\beta_j^e}$ and $\overline{\beta_i^e} < \overline{\beta_j^e}$. Finally, if both bounds are tied, heat i *dominates* heat j if $\underline{\beta_i^e} = \underline{\beta_j^e}$ and $\overline{\beta_i^e} = \overline{\beta_j^e}$ and $\psi_i < \psi_j$.

This dominance criterion aims to preserve, when possible, a range of feasible values for element e from heat i to heat j, where both heats are feasible. The practical reason that suggests this dominance criterion in our context is the following: if a feasible subset of input materials exists to produce heat i, then the chemical content transferred to the subsequent heat, through the liquid heel mechanism, is less likely to violate the bounds of a later heat, if that heat is less constrained, or if a range of chemical values where both elements overlap exists. The dominance rule can also produce a case in which the chemical ranges of i and j do not overlap; in that case, $\overline{\beta_i^e} \leq \underline{\beta_j^e}$, and the feasibility issues will not arise from the liquid heel, since the concentration of the specific element would need to increase drastically from one heat to the next. Therefore, a sequence that preserves such dominance relations tends to reduce the risk of infeasibility in the sequence.

The complication is that the dominance relations defined above are generally element-specific. Heat i may dominate heat j for a subset of elements but not for others; for example, consider two heats with $|E| = 5$: i may dominate j for e_1, e_2, e_4, while j dominates i for e_3, e_5.In this case, the dominance is not obvious. Moreover, even if in industrial settings there exists prioritization among chemical elements, based on their capacity to be reduced during the process by oxidation processes or similar procedures, from a modeling standpoint, this cannot be assumed, since even just one violation among a single chemical element in the

Hamiltonian path makes the model infeasible. An ordering that is perfectly dominance-consistent for all elements simultaneously may not exist. The solution to this problem is not simple. Our solution approach to this problem is to first compute ranks element by element and then aggregate them into an overall ordering: the aggregation acts as a compromise among conflicting element-wise preferences.

To compute the rank for an element e, we sort heats increasingly by the tuple $(\underline{\beta_i^e}, \overline{\beta_i^e}, \psi_i)$, using the above-defined dominance rules. Then we assign a rank to each heat in the sequence. Importantly, we do not assign ranks from 1 to $|H|$ purely by position. Doing so would impose a strict hierarchy even among heats that share the exact same chemical interval. This would be undesirable because when many heats have identical chemical ranges, the difference among them is not chemical but only related to secondary attributes (e.g., weight). Enforcing a fully strict order would incorrectly suggest that each position implies a strong difference in chemical restrictiveness. Therefore, both ranking methods adopt a dense scheme, in the sense that all heats with identical $(\underline{\beta_i^e}, \overline{\beta_i^e})$ are groupedand receive the same rank.

As an illustrative example, consider $|H| = 10$ and suppose the first three heats share one chemical interval while the remaining seven share another interval. A strict positional ranking would assign values from 1 to 10, suggesting large relative differences between heats ranked 4 and 10, even if they differ only marginally (e.g., by less than one ton of production). Under dense ranking, instead, we obtain only two rank groups, reflecting the true chemical equivalence structure. The ψ_i parameter can still be useful for local sorting, since it defines a strong local relation between heats, even if this relative position is not preserved by the rank.

After obtaining dense ranks $r = 1, 2, \ldots$, we assign the value r^2 rather than r. The motivation is to increase the relative influence of elements exhibiting high dispersion in their feasible intervals. As an example consider two elements: for e_1 the heats may cluster into only two dense ranks (low variability), while for e_2 they may split into many dense ranks (high variability). Since incompatibilities are more likely when variability is high, we want the final aggregated ordering to more closely reflect the ranking induced by e_2 than that induced by e_1. Squaring the ranks provides a simple and effective way to amplify differences without introducing additional parameters. Other convex transformations could be considered; we selected the square as a good compromise between impact and computational simplicity.

Up to this point, both ranking algorithms share the same structure. Their difference concerns how heats with an *unconstrained* interval for an element are treated.

We define an element to be unconstrained for a given heat when its admissible interval coincides with the full feasible domain, i.e.,

$$\underline{\beta_i^e} = 0 \quad \text{and} \quad \overline{\beta_i^e} = 100.$$

In the baseline method (Algorithm 3.1), such heats are simply included in the standard sorting. Because they have the lowest minimum and highest maximum, they tend to be placed at the end of the ordering for that element. This corresponds to interpreting the unconstrained heat as the least dominant.

However, in our setting, a more realistic interpretation is that an unconstrained element behaves as a *wildcard*: it can be placed in multiple positions without constraining the sequence, because its effective feasible interval will be shaped by the feasibility requirements imposed by neighboring heats and other constraints.

For example, consider three heats for a single element with intervals $[0, 1]\%$, $[2, 4]\%$, and $[0, 100]\%$. A strict dominance ordering would suggest $[0, 1]\% \rightarrow [2, 4]\% \rightarrow [0, 100]\%$. Yet feasibility for that element can still be achieved with sequences such as $[0, 100]\% \rightarrow [0, 1]\% \rightarrow [2, 4]\%$, because the wildcard heat placed before $[0, 1]\%$ effectively becomes constrained by what follows. A similar argument applies to $[0, 1]\% \rightarrow [0, 100]\% \rightarrow [2, 4]\%$.

To model this behavior in the heuristic, the wildcard-aware method (Algorithm 3.2) assigns rank 0 to unconstrained heats for that element. In this way, such heats do not enforce a strict dominance relation and are allowed to float in the overall aggregation.

Algorithm 3.1 RankDenseSquared(e, H)

$\mathcal{I} \leftarrow \emptyset$
$Rank \leftarrow \emptyset$
for all $h \in H$ **do**
  $min \leftarrow \text{MinComp}(h, e)$; $max \leftarrow \text{MaxComp}(h, e)$; $w \leftarrow \text{Weight}(h)$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{(name(h), min, max, w)\}$
end for ▷ $|\mathcal{I}| = |H|$
Sort $\mathcal{I}$ increasingly by (min, max, w) ▷ Sorting complexity: $\mathcal{O}(|H| \log |H|)$
$i \leftarrow 1$; $r \leftarrow 1$
while $i \leq |\mathcal{I}|$ **do**
  $j \leftarrow i$
  while $j \leq |\mathcal{I}|$ **and** $\mathcal{I}[j].min = \mathcal{I}[i].min$ **and** $\mathcal{I}[j].max = \mathcal{I}[i].max$ **do**
    $j \leftarrow j + 1$
  end while
  for $k \leftarrow i$ **to** $j - 1$ **do**
    $Rank[\mathcal{I}[k].name] \leftarrow r^2$
  end for
  $r \leftarrow r + 1$
  $i \leftarrow j$
end while
return *Rank* ▷ Overall complexity: $\mathcal{O}(|H| \log |H|)$

Algorithm 3.2 RankDenseSquaredWildcards(e, H)

$W_{\min} \leftarrow 0$; $W_{\max} \leftarrow 100$
$\mathcal{I} \leftarrow \emptyset$
$Rank \leftarrow \emptyset$
for all $h \in H$ **do**
  $min \leftarrow \text{MinComp}(h, e)$; $max \leftarrow \text{MaxComp}(h, e)$; $w \leftarrow \text{Weight}(h)$
  if $min = W_{\min}$ **and** $max = W_{\max}$ **then**
    $Rank[name(h)] \leftarrow 0$ ▷ wildcard assignment
  else
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{(name(h), min, max, w)\}$
  end if
end for ▷ $|\mathcal{I}| \leq |H|$
Sort $\mathcal{I}$ increasingly by (min, max, w) ▷ Sorting complexity: $\mathcal{O}(|\mathcal{I}| \log |\mathcal{I}|)$
$i \leftarrow 1$; $r \leftarrow 1$
while $i \leq |\mathcal{I}|$ **do**
  $j \leftarrow i + 1$
  while $j \leq |\mathcal{I}|$ **and** $\mathcal{I}[j].min = \mathcal{I}[i].min$ **and** $\mathcal{I}[j].max = \mathcal{I}[i].max$ **do**
    $j \leftarrow j + 1$
  end while
  for $k \leftarrow i$ **to** $j - 1$ **do**
    $Rank[\mathcal{I}[k].name] \leftarrow r^2$
  end for
  $r \leftarrow r + 1$
  $i \leftarrow j$
end while
return *Rank* ▷ Overall complexity: $\mathcal{O}(|H| \log |H|)$

3.5.2 Aggregation methods

After computing the element-wise ranks, we must aggregate them into one overall ordering that accounts for all elements simultaneously. Let $\mathscr{R}$ denote the collection of rank maps, i.e., for each element e we have a mapping $Rank_e(\cdot)$ from heats to rank values (possibly including zeros for wildcards).

A natural approach is to define for each heat a global score obtained by summing its ranks across elements, and then sort heats by this score. This yields a simple yet effective multi-criteria aggregation.

The first aggregation method (Algorithm 3.3) computes

$$Score(h) = \sum_{e \in E} Rank_e(h),$$

and then sorts heats increasingly by $(Score(h), \psi_h)$. This is computationally attractive, but it can overweight elements whose rank values span a wider numeric range.

It is fundamental to understand that not all element-wise rankings have the same spread. Some elements may produce only a few dense ranks (hence small squared ranks), while others may produce many. If ranks are summed directly, a high-variability element can dominate the aggregated score purely due to scaling, even when such dominance is not desired. To mitigate this, the second method (Algorithm 3.4) normalizes each element-wise rank into $[0, 1]$ before summation. The resulting score lies in $[0, |E|]$. This reduces sensitivity to scale differences while preserving the ordering signal within each element (including the effect of the squared transformation).

Algorithm 3.3 AggregateBySum$(H, \mathscr{R})$

```
Score ← ∅
for all h ∈ H do
    Score[name(h)] ← 0
end for
for all e ∈ E do
    Rank_e ← ℛ[e]
    for all h ∈ H do
        Score[name(h)] ← Score[name(h)] + Rank_e[name(h)]
    end for
end for
Result ← H
Sort Result increasingly by (Score, Weight)
return Result                                   ▷ Overall complexity: 𝒪(|E||H| + |H| log |H|)
```

3.5.3 Heuristic sorting

By combining the two ranking methods (Algorithm 3.1 and Algorithm 3.2) with the two aggregation methods (Algorithm 3.3 and Algorithm 3.4), we obtain four heuristic variants. These are implemented by Algorithms 3.5 to 3.8. Each heuristic returns a global ordering of the heats, which is interpreted as a candidate Hamiltonian path.

These heuristics are intentionally designed to be computationally cheap. Each method requires $|E|$ independent sorts of size $|H|$ (one per element), followed by a linear aggregation and a final sort. Consequently, each heuristic has complexity

$$\mathcal{O}(|E||H| \log |H| + |E||H| + |H| \log |H|),$$

which is negligible compared to the time typically required to solve the full MILP, or even a restricted one, especially in large instances. This makes it practical to compute all four orderings and test them systematically.

Algorithm 3.4 AggregateByNormalizedSum($H, \mathscr{R}$)

$Score \leftarrow \emptyset$
for all $h \in H$ **do**
  $Score[name(h)] \leftarrow 0$
end for
for all $e \in E$ **do**
  $Rank_e \leftarrow \mathscr{R}[e]$
  $maxRank \leftarrow \max_{h \in H} Rank_e[name(h)]$
  for all $h \in H$ **do**
    $r \leftarrow Rank_e[name(h)]$
    if $maxRank \leq 1$ **then**
      $nr \leftarrow 0$
    else
      $nr \leftarrow \dfrac{r-1}{maxRank-1}$
    end if
    $Score[name(h)] \leftarrow Score[name(h)] + nr$
  end for
end for
$Result \leftarrow H$
Sort *Result* increasingly by ($Score$, $Weight$)
return *Result*
▷ Overall complexity: $\mathcal{O}(|E||H| + |H| \log |H|)$

Algorithm 3.5 HeurSorting 1 (H)

$\mathscr{R} \leftarrow \emptyset$
for all $e \in E$ **do**
  $\mathscr{R}[e] \leftarrow$ RankDenseSquared(e, H)
end for
$Sorted \leftarrow$ AggregateBySum$(H, \mathscr{R})$
return *Sorted*

Algorithm 3.6 HeurSorting 2 (H)

$\mathscr{R}^W \leftarrow \emptyset$
for all $e \in E$ **do**
  $\mathscr{R}^W[e] \leftarrow$ RankDenseSquaredWildcards(e, H)
end for
$Sorted \leftarrow$ AggregateBySum$(H, \mathscr{R}^W)$
return *Sorted*

Algorithm 3.7 HeurSorting 3 (H)

$\mathscr{R} \leftarrow \emptyset$
for all $e \in E$ **do**
  $\mathscr{R}[e] \leftarrow$ RankDenseSquared(e, H)
end for
$Sorted \leftarrow$ AggregateByNormalizedSum$(H, \mathscr{R})$
return *Sorted*

Algorithm 3.8 HeurSorting 4 (H)

$\mathscr{R}^W \leftarrow \emptyset$
for all $e \in E$ **do**
  $\mathscr{R}^W[e] \leftarrow$ RankDenseSquaredWildcards(e, H)
end for
$Sorted \leftarrow$ AggregateByNormalizedSum$(H, \mathscr{R}^W)$
return *Sorted*

3.5.4 Computing feasibility

After computing a heuristic ordering, no explicit guarantee is available regarding feasibility for the original model: since this only proposes a Hamiltonian path, whereas feasibility depends on the full set of constraints. Therefore, we validate each heuristic path by solving the original MILP with the additional restriction that the path is fixed to the heuristic ordering.

Operationally, for a given heuristic ordering $Path_k$, we activate an arc only if it belongs to that ordering, and all other arcs are disabled. This enforces that the solver follows exactly the heuristic Hamiltonian path. The resulting problem is substantially easier, because the solver no longer needs to explore the combinatorial space of possible paths. As a result, it can quickly determine whether the remaining constraints admit a feasible assignment, and, if so, compute an objective value, or if the heuristic path is infeasible.

The general procedure, that allows for obtaining the best feasible solution is described in Algorithm 3.9. The algorithm computes each heuristic path, solves the corresponding restricted model, evaluates the heuristic solution to the local best heuristic and if the newly computed is better update the best known solution. This produces three possible scenarios:

1. All / Some heuristics are feasible: some if not all restricted model yields a feasible solution. In this case multiple candidates exist, and we select the one with the best objective value.

2. No heuristic feasible: none of the restricted models is feasible.

In case (1), the best heuristic solution provides a feasible primal point that can reduce the computational effort required to improve bounds and, when possible, help the solver in proving optimality. In case (2), the solver must rely entirely on its internal mechanisms to find a first feasible solution.

Algorithm 3.9 BESTHEURISTIC($\mathscr{P}, H$)

```
bestSol ← ∅; bestObj ← +∞
for all r ∈ {1, 2, 3, 4} do
    Path_r ← HEURSORTINGr(H)
    (Sol_r, Obj_r, status_r) ← SOLVEPATHCONSTRAINEDMODEL(P, Path_r)
    if status_r = feasible then
        if Obj_r < bestObj then
            bestObj ← Obj_r
            bestSol ← Sol_r
        end if
    end if
end for
return (bestSol, bestObj)
```

Regarding the heuristic performance, and related effectiveness to help the model, in proving the optimality, all the information is reported in Section 3.7.

3.6 An ALNS Algorithm for the MHSSP-LH

In this section, we first provide a general introduction to the Adaptive Large Neighborhood Search (ALNS) metaheuristic, then we introduce the proposed ALNS to solve the MHSSP-LH.

3.6.1 Introduction to ALNS

First introduced in Ropke and Pisinger (2006) and Pisinger and Sigurd (2007), the Adaptive Large Neighborhood Search (ALNS) metaheuristic has become one of the most commonly used metaheuristic algorithms to solve complex combinatorial problems, thanks to its adaptability and ease of implementation. An ALNS algorithm starts from an initial feasible solution, then it iteratively modifies the current solution by applying, destroy and repair operators. Each destroy operator modifies the current solution by rendering it either infeasible or of the worst quality. On the other hand, a repair operator either restores the feasibility or tries to improve the value of the current solution. An ALNS algorithm makes use of multiple destroy and repair operators. At each iteration, random destroy and repair operators are selected, usually through a roulette wheel selection. Given Ω^- and Ω^+, the sets of destroy and repair operators, the wheel selection method creates $|\Omega^-| \times |\Omega^+|$ local search algorithms, each one with its own distinct neighborhood of the current solution. The different operators contribute to expanding the effective size of the neighborhood, without the need to design and implement complex local search heuristics.

In Algorithm 3.10 we present the general outline of the method. First, an initial solution is constructed, usually with a constructor operator, and it is saved as the current incumbent solution x^*. In the version proposed by Ropke and Pisinger (2006), the ALNS is controlled

Algorithm 3.10 Adaptive Large Neighborhood Search

1: Construct a feasible solution x
2: $x^* \leftarrow x$
3: $t \leftarrow$ initial temperature
4: **while** $t \geq t_{min}$ **do**
5: $d \leftarrow$ random destroy operator from Ω^- using scores π_i^-
6: $r \leftarrow$ random repair operator from Ω^+ using scores π_i^+
7: Generate new solution $x' \leftarrow r(d(x))$ applying destroy and repair operators
8: **if** Accept x' as current solution ? **then**
9: $x \leftarrow x'$
10: **end if**
11: **if** $f(x') < f(x^*)$ **then** ▷ Assume a minimization problem.
12: $x^* \leftarrow x'$
13: **end if**
14: Update score $\pi_{(r)}^+$ and $\pi_{(d)}^-$ of the selected operators
15: $t \leftarrow cr * t$
16: **end while**
17: **return** x^*

with the simulated annealing method. The temperature is used to determine the termination criteria, as shown in Line 4, and as an acceptance criterion for worsening solutions. However, it is possible to embed the ALNS inside other metaheuristics like a Tabu-Search. During each iteration, a random destroy and a random construct operator are selected. These operators are applied to the current solution x to generate a new solution x'. Then the algorithm decides to accept or reject the new solution. A common acceptance criteria is to always accept an improving solution, and to accept a worsening solution with a probability dependent on the current temperature. At this point the algorithm checks if the new solution is better than the incumbent, and in that case updates it. After checking the solution value the algorithm updates the scores for the selected operators according to the obtained improvements. At the end of each iteration the algorithm proceeds with the temperature schedule.

ALNS effectively explores large solution spaces, balancing intensification and diversification to address the trade-offs in MHSSP-LH. This type of search could be promising given the structure of the problem.

3.6.2 The proposed ALNS Algorithm

The ALNS framework iteratively applies the destroy and repair operators to refine solutions. Destroy operators remove arcs from the underlying graph, while repair operators reassign them more effectively. An adaptive scoring mechanism prioritizes effective operators. This process is embedded in an Iterated Simulated Annealing (ISA) framework, which regulates solution acceptance via a temperature-based criterion, aiding in escaping local optima while balancing exploration and exploitation. The stopping condition is the maximum computation time.

In the following, we introduce the concept of heat cost to denote the cost of creating a single heat in the sequence. As stated before, this cost is sequence dependent, in the sense that it can change based on its relative position with respect to all the other heats. In this sense we can assume that the high cost of some heats can be reduced if their position is

changed. This principle guides the development of our destroy operators, which aim to minimize the creation of high-cost heat in the path.

One important difference to the classical implementation of the ALNS is that in our case, just one repair operator is used based on the resolution of the MILP formulation, where some decision variables have been fixed. This approach allows us to compute with ease the chemical boundaries needed for the feasibility of the solution, and with that, know directly if the repaired solution is feasible or not.

Algorithm 3.11 outlines our ALNS within an ISA framework, initializing a baseline starting solution and assigning an initial score of 1 to each operator. The outer loop runs until the time limit is reached, initializing operator scores before executing the simulated annealing temperature schedule. In each iteration of the inner loop, a destroy operator is selected via roulette wheel selection (Line 6). The selected destroy operator removes a predetermined number of arcs from the current Hamiltonian path (Line 7). The single repair operator then tries to find a new minimal feasible cost solution, in which the removed arcs are the only arcs not with a fixed value (Line 8). In Lines 10–23, the algorithm decides whether to accept the new solution. Lower-cost solutions are always accepted, while worse solutions are accepted probabilistically based on the current temperature. If the new solution improves the incumbent, the latter is updated, and a reheating policy is applied. Given the high possible values of the solution, the increase in temperature is capped at a maximal value of three times the initial temperature T_0. The new solution is compared to the current solution based on cost difference, and a temperature-dependent criterion is used to determine if the new solution should be accepted. The temperature is systematically reduced by the so-called cooling rate factor *cr* (Line 25). Finally, in Lines 26–29, operator scores are reset to their initial value after each *Limit* iterations. During iterations, operator scores are updated based on solution quality (Lines 15–21):

- New best solution: Operators are rewarded by η_{best} when finding a new globally improved solution.
- New incumbent solution: Operators are increased by $\eta_{\text{incumbent}}$ when finding a solution better than the current one but not the best.
- Accepted solution: Operators are rewarded by η_{accepted} even if they do not provide any solution improvement.

In the remainder of this section, we describe the destroy operators tailored to the MHSSP-LH, highlighting their role in addressing the problem's specific characteristics.

Destroy Operators

Each destroy operator removes some arcs from the current feasible Hamiltonian path. Some classical random arc removal procedures are defined. However, other operators that are problem-based are also introduced, all of which use the information related to the single node cost.

(`RA_`n) Random Arcs Elimination. This procedure randomly removes n arcs from the current Hamiltonian path. The removal is equivalent to fixing at zero the select arc in the MILP model.

(`HCN_`n) Highest Cost Nodes Elimination. This heuristic targets the n highest cost nodes along the path, and then removes the corresponding entering and leaving arcs for each identified node. This method is deterministic. Thus, to avoid cycling, the following procedure is implemented:

Algorithm 3.11 ALNS within ISA Framework

Initialize solution sol
while stopping criteria not met **do**
    operator scores $\omega_d \leftarrow 1$, temperature $temp \leftarrow T_0$
    iterations $\leftarrow 0$
    while $temp > t_{\min}$ **do**
        Get destroy operators d via roulette wheel method on ω_d
        $sol' \leftarrow$ apply d to remove n arcs from sol ▷ Destroy phase
        $sol' \leftarrow$ apply r to reinsert removed parcels into sol' ▷ Repair phase
        Compute cost difference $\delta = \text{cost}(sol') - \text{cost}(sol)$
        if $\delta < 0$ or $U(0,1) < \exp(-\frac{\delta}{temp})$ **then**
                ▷ Acceptance rule, with $U(0,1)$ Uniform random in $[0, 1]$.
            $sol \leftarrow sol'$
            Update scores:
            if S' is a new best solution **then**
                $\omega_d \leftarrow \omega_d + \eta_{\text{best}}$ ▷ Reward for new best solution
                Reheat: $temp \leftarrow min(temp + \Delta t, 3 * T_0)$ ▷ $\Delta t = (\bar{z} - z^*) \exp\left(-\frac{temp}{T_0}\right)$
            else if S' is better than current solution **then**
                $\omega_d \leftarrow \omega_d + \eta_{\text{incumbent}}$
                ▷ Reward for new incumbent
            else
                $\omega_d \leftarrow \omega_d + \eta_{\text{accepted}}$
                ▷ Reward for accepted solution
            end if
        end if
        Cool temperature: $temp \leftarrow cr * temp$ $(\alpha \in (0, 1))$
        if iterations = Limit **then**
                ▷ Reset operator scores after a certain number of iterations
            operator scores $\omega_d \leftarrow 1$,
            iterations $\leftarrow 0$
        else
            iterations $\leftarrow$ iterations + 1
        end if
    end while
end while
return best solution found

(a) Sort Nodes: Sort the nodes (heats), based on their cost.

(b) Check Cost Solution: Verify if the current solution is the same as the one obtained from the previous call. If it is the same, remove the nodes stored in the disabled-nodes list. Then discard the current highest-cost node and add it to the list.

(c) Node Selection: Select the n nodes with the highest cost among those not present in the disabled-nodes list.

(d) Arc Removal: Remove the entering arc and leaving arc from each selected node.

(`HCS_n`) Highest Cost SubSequence Elimination. This heuristic targets the highest subsequence

of n nodes along the path, and then removes the corresponding entering and leaving arcs for each identified node in the sequence. This method is deterministic. Thus, to avoid cycling, the following procedure is implemented:

(a) Sort Nodes: Sort all possible subpath of dimension n (n-heat-path), based on their cost.

(b) Check Cost Solution: Verify if the current solution is the same as the one obtained from the previous call. If it is the same, remove all the subpaths stored in the disabled-subpath list. Then discard the current highest-cost subpath and add it to the list.

(c) Subpath Selection: Select the highest subpath among the one not present in the disabled-subpath list.

(d) Arc Removal: Remove the entering arc and leaving arc from each selected node, inside the subpath.

In standard ALNS implementations, a destroy operator may receive the same incumbent solution multiple times, without creating any problematic situation, because different repair operators can generate different new incumbent solutions even when starting from the same destroyed solution. In our approach, however, multiple repair operators are not used. As a result, the algorithm may more easily enter a cycle in which identical solutions are repeatedly generated and evaluated. This situation becomes particularly problematic when the temperature parameter decreases, since the probability of accepting worsening solutions decreases, increasing the risk of stagnation.

To address this issue, our approach introduces a controlled perturbation to the destroy operators that is activated only when this specific cycle is detected. When such a condition occurs, the method removes the highest node or sequence, as described above, thereby forcing the generation of a different destroyed solution. This increases the likelihood of producing a new incumbent solution capable of altering the current solution and allowing the search process to continue exploring new regions of the solution space, with a reduced risk of cycling.

Repair Operator

The repair operator aims to construct a feasible solution for the MHSSP-LH. This is achieved by solving a restricted MILP formulation of the problem in which part of the decision variables have been fixed according to the outcome of the destroy operator. Specifically, all arcs selected by the destroy operator are fixed to zero, while all arcs belonging to the portion of the solution that is not destroyed are fixed to one. The remaining arcs are left unconstrained. This reduction of the search space significantly simplifies the problem and allows the original MILP formulation to be directly used as a repair operator. After tackling the restricted problem, the solver may return either an optimal solution or determine that the restricted problem is infeasible. This behavior does not affect the general ALNS framework, given that if the restricted problem is infeasible, the iteration simply does not produce a candidate solution and the algorithm proceeds to the next iteration, effectively increasing the iteration count without updating the current solution. The constructive heuristic defined in section 3.5 is used to provide a starting solution for ALNS.

3.7 Experimental Results

This section presents the experiments conducted. First, we in depth describe how we defined the test instances that are based on the company data. Then, using as reference the optimized company solution, we evaluate all the constructive heuristics, and the ALNS, under both cost and execution time. Furthermore, extensive computational analysis of the model solutions through Gurobi solver are presented. In particular, Section 3.7.1 discusses how the instances are defined.

The mathematical model was implemented in Java 21. All experiments were conducted on a Manjaro 23.1 system equipped with a Ryzen 5 5600U 12-core processor and 42 GB of RAM, using Gurobi 10.0.1 as MILP solver. All available threads were used during the experiments.

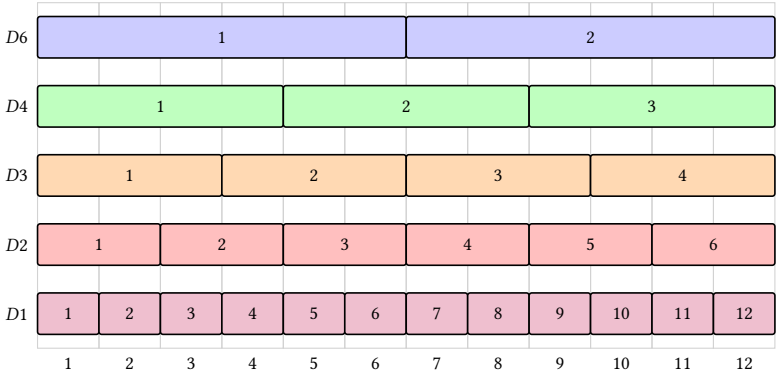
3.7.1 Instances description

The instances considered in this study are derived from real production data collected from the company over a time horizon of 30 consecutive days. From this dataset, we generate several sets of benchmark instances corresponding to different planning horizons. Specifically, the instances are grouped into five sets, denoted by $D1$, $D2$, $D3$, $D4$, and $D6$, representing optimization problems spanning one, two, three, four, and six consecutive days, respectively. The selection of these planning horizons reflects the company's actual production planning practices. The one-day ($D1$) and two-day ($D2$) horizons support purely operational decision-making and are primarily used to react to unforeseen production issues or to validate short-term heat sequences before execution. The three-day ($D3$) and four-day ($D4$) horizons correspond to tactical planning activities, where production schedules are coordinated and validated across multiple departments while balancing operational efficiency and resource availability. Finally, the six-day ($D6$) horizon is employed for strategic production assessments, supporting inventory management, logistics coordination, and higher-level inter-department planning. A five-day planning horizon is intentionally not considered, as it is not adopted in the company's planning process and therefore does not represent a relevant industrial use case. By aligning the benchmark instances with these real-world planning horizons, the experimental evaluation reflects the different decision-making levels encountered in practice, ranging from operational scheduling to strategic production planning.

Within each set, the instances are constructed so that they do not overlap. For example, in the set $D2$, the first instance includes days 1 and 2, the second instance includes days 3 and 4, and so on, until the end of the 30-day time horizon is reached. All instance sets are derived from the same 30-day time frame. Consequently, longer-horizon instances naturally contain several shorter-horizon segments. As the illustrative example shown in Figure 3.4 shows.

This experimental design enables the analysis of the model behavior as the number of heats to be optimized increases with the planning horizon. However, each instance in the set is considered independent. For example, in the set $D1$ instances 1 and 2 and their overall cost cannot be compared to the single resolution of the first $D2$ instance. When optimizing single day instances, we suppose to have no prior liquid heel, which is not the case for instance in D2. This means that there is no connection between consecutive days in D1. The same logic

Figure 3.4: Instances overlapping ranges.



is applied over all the defined instances. This implies that no rolling horizon like analysis can be done between different time intervals since they are not comparable.

Table 3.6 provides an overview of the instances used in computational experiments, including summary statistics describing the number of heats contained in each instance, as well as the average increase in the number of heats in consecutive instance classes. Some

Table 3.6: Instances summary description.

# Days	# Instances	# Heats			Size % Increase
		Min	Max	*Avg.*	
1	30	10	18	14.76	-
2	15	24	36	29.53	100.00
3	10	34	49	44.30	50.00
4	7	52	63	59.28	33.83
6	5	82	96	88.60	49.45

characteristics of the instances are common across all defined sets. Starting from λ, which is defined as 65000 kg, then each possible scrap and ferroalloy starting inventory (ω_s^S, ω_k^K) is supposed to be equal to 50000 kg. The γ is equal to €0.2872 for each KWh, that was the average cost in Europe during the first half of 2025. The liquid heel weight is based on company estimation and is equal to 4000 kg. The minimum lot δ is defined equal to 100 kg. All the scrap and ferroalloy cost with their respective yield, and heat fusion coefficients, are based on company data. The model evaluates the chemical composition of the steel by considering 15 chemical elements, namely C, S, P, Mn, Cr, Ni, Mo, Cu, Si, V, Sn, As, Co, W, and Sb. These elements define the set E.

Based on the chemical analyses of the steel grades considered in the dataset, Table 3.7 reports the distribution of each chemical element across all the different types of steel included in the instances.

Overall, the benchmark instances are reconstructed from historical company data rather than being entirely synthetic. The production orders, steel grades, and planning horizons are taken directly from industrial operations, while a limited number of parameters that are not available in the historical records, such as some inventory levels and planning assumptions,

Table 3.7: Chemical composition related to the steel.

β	Type	C	S	P	Mn	Cr	Ni	Mo	Cu	Si	V	Sn	As	Co	W	Sb
Max	min (%)	0.121	0.010	0.005	0.240	0.050	0.072	0.010	0.045	0.280	0.003	0.004	0.005	0.010	0.003	0.001
	max (%)	4.500	0.075	0.045	1.500	15.300	4.450	0.500	1.800	0.700	2.850	0.050	0.100	1.350	0.800	0.100
Min	min (%)															
	max (%)	0.320				15.100	4.200	0.459	1.550						0.750	

are defined according to company estimates or fixed values. Consequently, the resulting benchmark instances should be regarded as representative industrial case studies.

3.7.2 Numerical Tolerances and Precision Considerations

Before presenting the computational results, it is necessary to discuss the numerical tolerances adopted in the optimization process. The model relies entirely on real industrial data and aims to produce solutions with a level of accuracy that is consistent with real production environments. For this reason, all parameters used by the model correspond to actual measurements rather than artificially generated values. While this choice improves the realism and applicability of the solutions obtained, it also introduces numerical challenges. Real-world datasets typically exhibit greater variability and scale differences than synthetically generated data, which may lead to increased numerical instability during the optimization process.

In particular, the model requires high precision to represent the chemical composition of scrap materials, ferroalloys, and steel requirements. All chemical percentages are expressed with a precision of 10^{-3}. Subsequently, these percentages are multiplied by the quantities of materials used in the model to compute the absolute chemical contributions. The quantities are expressed in kilograms, which is appropriate given our defined δ in the 100 kg range, moreover a typical production heat ranges from approximately 30 to 55 tons of steel. As a consequence, the chemical coefficients—initially defined with precision 10^{-3} are transformed into decimal values with effective precision up to 10^{-5} when combined with mass quantities. These values are then multiplied by decision variables whose magnitudes can reach 10^5, and the associated constraint coefficients may also reach values of the same order of magnitude. This results in a constraint matrix characterized by a relatively wide numerical range. Such a scaling can introduce non-negligible numerical instability during the optimization process, particularly when the optimality gap is forced to be very small or when the search space remains weakly restricted, as in the case where many heats need to be optimized.

Preliminary computational experiments revealed that enforcing the standard Gurobi optimality tolerance of 10^{-4} significantly increases computation times without yielding a meaningful improvement in the objective value. As an illustrative example, large industrial instances containing up to 96 heats may reach objective values on the order of €2.2 million. Under an optimality tolerance of 10^{-4}, the solver attempts to identify solutions whose deviation from optimality is below approximately €220. From both a theoretical and an industrial perspective, such a level of precision is unnecessarily strict for the application considered. Based on these considerations, the optimality tolerance was relaxed to 10^{-3}. In the previous example, this corresponds to allowing an optimality gap of approximately €2200, which represents a more reasonable trade-off between computational effort and practical relevance in a real production environment. A similar rationale was applied to the feasibility tolerance of the constraints. The default Gurobi feasibility tolerance of 10^{-5} was increased to 10^{-4}. This level of tolerance remains sufficiently strict for the problem at hand. Indeed, a violation

of this magnitude would correspond to discrepancies measured in milligrams rather than kilograms when translated into material quantities. Such precision remains well beyond what is required in the industrial context while providing improved numerical stability from a computational standpoint. Given these considerations, throughout this work all solution methods classify a solution as optimal whenever the optimality gap is below 0.1%.

3.7.3 Company Solution Definition

The company solution is adopted as a reference baseline throughout this analysis. However, this baseline does not correspond exactly to the full solution implemented in practice. Instead, it is constructed by fixing the historical company heat sequence and subsequently optimizing it under the proposed model formulation. In real operations, the company determines not only the sequence of heats, but also the composition for each heat, including the quantities of scrap and additional ferroalloys. Therefore, the true operational solution jointly involves sequencing and material selection decisions. The baseline considered in this study should not be interpreted as a replication of the real industrial solution. Rather, it represents an estimate of the performance that the company could achieve if the same sequences were optimized using the proposed modeling framework. This evaluation choice is motivated by structural differences between the current industrial practice and the proposed model. In particular, the company does not explicitly account for the impact of the liquid heel across consecutive heats. Instead, the residual liquid heel from one heat is treated as a fixed input to the next, without any cross-heat optimization. As a consequence, the planning process is effectively decomposed into a sequence of independent single-heat optimization problems, even when addressing multi-heat production scenarios.

To further contextualize the baseline, we computed the average cost reduction obtained by re-optimizing the fixed company sequences and compared it with historically accurate raw material selection. The results presented for the set $D6$ are reported in Table 3.8 indicate an average cost reduction of approximately 18%. This significant improvement suggests that even without modifying the sequencing decisions, a more accurate and integrated modeling of the process can lead to substantial cost savings.

Table 3.8: Cost comparison: real historical cost vs optimized company solution.

Instance	Real Cost	Optimized Company Solution	Δ %
1	2361907.604	1935600.927	22.0
2	2891977.438	2434443.213	18.8
3	2883902.307	2406679.079	19.8
4	2692103.812	2366046.901	13.8
5	2735792.922	2366052.277	15.6
Avg	2713136.8171	2301764.479	18.0

Finally, it is worth noting that the proposed approach does not aim to replicate the full historical company solution. Instead, it leverages historically accurate sequences to reconstruct what the company's optimization outcomes could have been if the proposed model had been available. This evaluation framework ensures consistency and fairness when comparing the proposed solution methods, including constructive heuristics, Adaptive Large Neighborhood Search (ALNS), and the full MILP formulation.

3.7.4 Constructive Heuristic analysis

In this section, we evaluate the performance of the proposed constructive heuristics in comparison with the optimized solution of the heats sequence currently adopted by the company. We denoted our constructive methods as Heuristic 1–4.

Given the relevance of each instance and the corresponding heuristic outcomes, both costs and computational times are reported. In particular, detailed cost results are presented in Tables 3.9 to 3.13. Each table includes one column for the company solution and one for each heuristic approach. The best-performing heuristic (i.e., the one achieving the minimum cost) is highlighted in bold. The columns *Abs Diff* and *% Diff* report, respectively, the absolute and percentage differences between the company solution and the best heuristic value. The corresponding detailed computational times are reported in Tables 3.16 to 3.20. In this case as well, results are provided for both the company solution and the heuristic approaches, including computational time and the final optimality gap.

As the summary overall Figure 3.5, shows it is evident that the absolute cost reduction increases with the length of the planning horizon (upper panel of the figure). In particular, the gap between the company solution and the best heuristic grows approximately linearly, ranging from about 9k€ for the one-day horizon ($D1$) up to around 50k€ for the six-day horizon ($D6$). At the same time, the percentage improvement remains remarkably stable across all horizons, with an average value of approximately 2.1%. This indicates that the increase in absolute savings is primarily driven by the larger scale of the problem, rather than by a structural improvement in solution quality. In other words, extending the time horizon increases the total achievable savings, while the relative gain per heat remains essentially constant. This suggests that the cost reduction is largely independent of the number of heats considered, pointing to a structural and consistent improvement provided by the constructive heuristics.

The presence of several outliers in both panels highlights that, for specific instances, the heuristics can exploit particularly favorable configurations, achieving substantially higher improvements than average. Conversely, there are also cases in which the constructive heuristics perform worse than the company solution. However, these situations are not representative of the general trend, and they further emphasize the inherent variability and complexity of the problem, as well as the impact of adopting a modeling-based optimization approach.

Figure 3.5: Cost reduction of best constructive heuristics compared to the company solution.

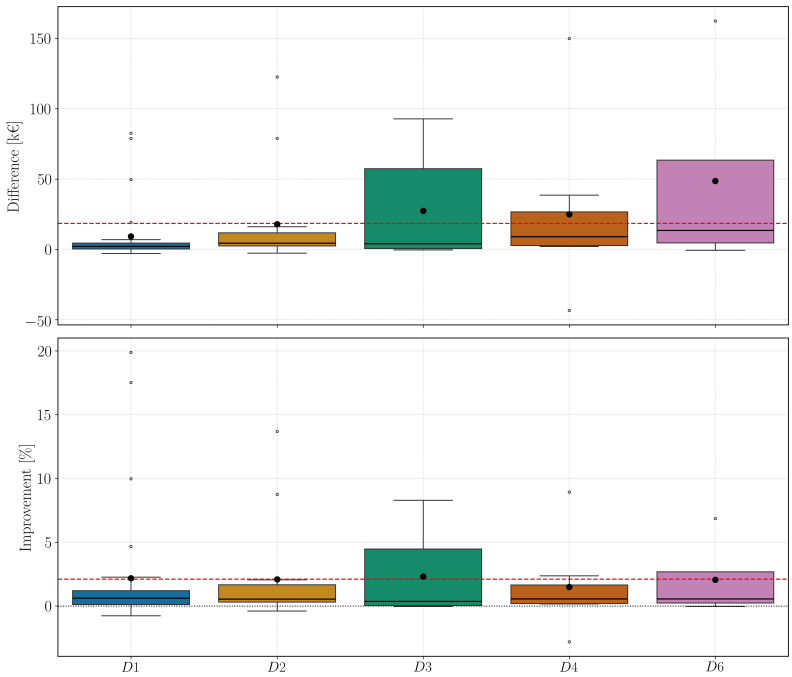


Table 3.9: *D*1 Company and Constructive heuristic solution.

Instance	Company	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Abs Diff	% Diff
1	238950.611	**236413.589**	**236413.589**	238756.021	239187.210	2537.022	1.062
2	308706.699	**306769.196**	307302.358	308576.631	310987.594	1937.503	0.628
3	236722.938	**236817.149**	**236817.149**	237704.203	**236817.149**	-94.211	-0.040
4	431513.961	433052.767	**433038.452**	433863.530		-1524.491	-0.353
5	357277.439	**356651.245**	356664.989	357280.440	358378.160	626.194	0.175
6	360251.820	361424.595	360825.445	361424.595	**358086.378**	2165.442	0.601
7	408408.909	406681.520	407659.278	**405850.321**		2558.588	0.626
8	479995.351			**476896.279**		3099.072	0.646
9	438354.110		**434195.427**	435694.726	437090.153	4158.683	0.949
10	432817.113	434868.894	435012.267	**432474.097**	479431.911	343.016	0.079
11	388668.281	391613.414	**391598.629**	395773.047	396942.820	-2930.348	-0.754
12	284064.143	281690.985	**281671.215**	283538.129	283555.893	2392.928	0.842
13	401109.694	406482.573	407184.408	405088.101	**401837.522**	-727.828	-0.181
14	396445.864	**394591.501**	**394591.501**	397425.069	396613.397	1854.363	0.468
15	379075.908				**374431.560**	4644.348	1.225
16	353218.489				**348653.304**	4565.185	1.292
17	447174.700	446479.925	446858.649	**445695.256**	446023.004	1479.444	0.331
18	428582.054	429204.612	429439.284	**428513.591**	429347.778	68.463	0.016
19	308685.820	302187.901	**301668.190**		306776.327	7017.630	2.273
20	379400.192	379397.892	376408.582	**374907.727**	377800.484	4492.465	1.184
21	397269.184	321422.933	**318285.777**			78983.407	19.882
22	498119.924		455327.143		**448377.804**	49742.120	9.986
23	411718.860	392759.062	395660.333		**392507.514**	19211.346	4.666
24	370274.881	370859.159	371153.460	**368255.942**	420349.027	2018.939	0.545
25	439902.837	439998.090	440727.943	**439056.738**	441053.863	846.099	0.192
26	289885.097	**289534.238**	289862.033	290251.770	289534.730	350.859	0.121
27	429920.870	430968.421	431361.769	**429406.550**	430954.903	514.320	0.120
28	471131.921	**388525.149**	390328.782	430329.205		82606.772	17.534
29	437602.796	**431660.404**	434825.931		438606.692	5942.392	1.358
30	293572.906	**293455.940**	**293455.940**	295198.259	294774.143	116.966	0.040
Avg	383294.112	**366540.446**	372531.056	377041.749	373524.773	9299.890	2.184

Table 3.10: D2 Company and Constructive heuristic solution.

Instance	Company	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Abs Diff	% Diff
1	548617.875	**543943.329**	546733.451	548580.194		4674.546	0.852
2	668941.120	669498.889	**668712.199**	669665.143	673721.093	228.921	0.034
3	718097.950	715392.619	715493.123	**714266.552**	715100.053	3831.398	0.534
4	888598.867	**881058.978**	925229.618	883764.615		7539.889	0.849
5	870828.018		**866394.127**	869699.233	990824.363	4433.891	0.509
6	673495.082	677075.179	677118.563	**676102.718**	678271.623	-2607.636	-0.387
7	797903.005	**794711.695**	795594.513	797346.281	797775.846	3191.310	0.400
8	732391.498			747685.191	**719716.136**	12675.362	1.731
9	875907.845	877567.138	876743.355	**874033.451**	875464.441	1874.394	0.214
10	688434.732	677537.380	**677349.323**	680146.781	682964.913	11085.409	1.610
11	895624.234	777506.172	778505.862		**773010.585**	122613.649	13.690
12	782123.493				**765992.840**	16130.653	2.062
13	729975.137	729819.590	**729539.387**	731629.085	732281.391	435.750	0.060
14	901896.778	824320.297	**822941.443**	922375.237		78955.335	8.754
15	733037.568	**729779.485**			731531.804	3258.083	0.444
Avg	767058.213	**741517.563**	756696.247	759607.873	761387.924	17888.064	2.090

Table 3.11: D3 Company and Constructive heuristic solution.

Instance	Company	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Abs Diff	% Diff
1	785548.659	**783764.778**	783842.850	785443.873		1783.881	0.227
2	1149173.772	1151108.202	1150581.367	**1148841.537**	1151575.692	332.235	0.029
3	1327187.806			**1319922.717**	1373603.506	7265.089	0.547
4	1108183.719	1107061.289	**1106944.920**	1108445.744	1184680.604	1238.799	0.112
5	1177066.307			1181919.613	**1177155.288**	-88.981	-0.008
6	1229269.583				**1223097.785**	6171.798	0.502
7	1085702.642	1002394.597	999420.697	**995707.651**	1001386.866	89994.991	8.289
8	1280008.481	**1205870.893**	1213951.515	1215409.221	1260660.479	74137.588	5.792
9	1160341.697	**1160634.841**	1162094.871	1161285.132	1228515.311	-293.144	-0.025
10	1204733.401	**1111912.652**	1114615.445			92820.749	7.705
Avg	1150721.607	**1074678.179**	1075921.666	1114621.936	1200084.441	27336.300	2.317

Table 3.12: *D*4 Company and Constructive heuristic solution.

Instance	Company	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Abs Diff	% Diff
1	1217595.550	1216942.423	**1215567.529**	1220188.196		2028.021	0.167
2	1606489.431			**1597539.724**		8949.707	0.557
3	1545670.473			**1589049.701**	1622247.338	-43379.228	-2.806
4	1530446.729			**1526914.433**	1545708.773	3532.296	0.231
5	1564644.207	1550164.662	**1549773.910**	1555776.751	1622102.817	14870.297	0.950
6	1677770.991	**1527826.567**	1530259.666	1532859.417	1533679.072	149944.424	8.937
7	1632209.630	1667839.749	**1593563.655**	1617338.441		38645.975	2.368
Avg	1539261.002	1490693.350	**1472291.190**	1519952.380	1580934.500	24941.642	1.486

Table 3.13: *D*6 Company and Constructive heuristic solution.

Instance	Company	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Abs Diff	% Diff
1	1935600.927	**1930971.791**	1931878.162	1934453.542		4629.136	0.239
2	2434443.213			2458732.115	**2421006.232**	13436.981	0.552
3	2406679.079			**2407334.954**	2418641.295	-655.875	-0.027
4	2366046.901	2207900.739			**2203629.789**	162417.112	6.864
5	2366052.277	2306506.252	**2302515.655**	2337928.493	2338081.153	63536.622	2.685
Avg	2301764.479	2148459.594	**2117196.908**	2284612.276	2345339.617	48672.795	2.063

A key aspect of the proposed constructive heuristics is their ability to generate feasible solutions across a wide range of instances. As shown in Table 3.14, none of the heuristics is able to consistently provide feasible solutions for all instances. Each heuristic covers only a subset of the problem instances, with performance that varies depending on the planning horizon. However, when all heuristics are considered together, the coverage significantly improves. In particular, the combined use of all heuristics allows us to obtain feasible solutions for all instances, demonstrating that the heuristics are complementary rather than redundant. This highlights the importance of employing the full set of constructive strategies, as no single heuristic is sufficient to robustly address all cases. This behavior is further confirmed in Table 3.15, where the distribution of best-performing solutions shows that each heuristic contributes to the overall performance. No single method consistently dominates the others; instead, different heuristics outperform the company solution on different subsets of instances.

These results emphasize that the strength of the proposed approach lies not in a single heuristic, but in their combined application. From a practical perspective, this suggests that our strategy where all heuristics are executed and then the best feasible solution is selected represents the most effective way to ensure both feasibility and good solution quality.

Finally, it is important to recall that the primary objective of these constructive heuristics was not to outperform the company's solution, but to reliably generate feasible solutions. In this respect, the results are particularly significant, given that not only is feasibility achieved for all of the instances, but the resulting solutions also consistently improve upon the company's baseline. This further underlines the effectiveness of a model-based approach, even when the focus is on feasibility rather than optimality.

Table 3.14: Feasibility results of company solution and heuristics.

Days	# Instances	Company	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	*min* Heuristic
D1	30	30 / 30	25 / 30	27 / 30	23 / 30	25 / 30	30 / 30
D2	15	15 / 15	12 / 15	12 / 15	12 / 15	12 / 15	15 / 15
D3	10	10 / 10	7 / 10	7 / 10	8 / 10	8 / 10	10 / 10
D4	7	7 / 7	4 / 7	4 / 7	7 / 7	4 / 7	7 / 7
D6	5	5 / 5	3 / 5	2 / 5	4 / 5	4 / 5	5 / 5
Tot.	67	67 / 67	51 / 67	52 / 67	54 / 67	53 / 67	67 / 67

Table 3.15: Best solution occurrences of heuristics and number of instances where heuristics outperform the company solution.

Days	# Instances	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	*min* Heuristic
D1	30	9 / 30	10 / 30	9 / 30	7 / 30	26 / 30
D2	15	4 / 15	5 / 15	3 / 15	3 / 15	14 / 15
D3	10	4 / 10	1 / 10	3 / 10	2 / 10	8 / 10
D4	7	1 / 7	3 / 7	3 / 7	0 / 7	6 / 7
D6	5	1 / 5	1 / 5	1 / 5	2 / 5	4 / 5
Tot.	67	19 / 67	20 / 67	19 / 67	14 / 67	58 / 67

Regarding the computation time, all the Tables 3.16 to 3.20, report the whole set of instances in detail. A summary representation of execution time and gap is presented in Fig-

ure 3.6 and Figure 3.7.

Regarding the execution time, all the tests were executed with a time limit of 600s. As the data, and related figures show, all of our constructive heuristics have a maximal execution time of less than 120s; this means that combining the four methods independently of the size, the maximal computation time is at most 8 minutes. The average case, however, suggests that the value can be much lower. Another aspect that can be easily seen by the data is that, independently of the heuristic solution, there is a consistent increase in computation time every time we increase the time period under consideration. A computational fundamental aspect that the plots cannot explain but is much clearer in the tables is that the execution time necessary for proving the infeasibility of a solution is less than half a second.

As already introduced, the solution is considered optimal for every problem if the corresponding gap is less than 0.1%, which can be seen in Figure 3.7. However, even imposing such an optimality gap, we notice that the average mean is around 0.07%. This implies that if needed, at almost zero computational cost, the optimality precision can be increased to that value. This result also suggests that imposing a much higher precision than the one initially defined by Gurobi (1e-4) can be challenging. The execution time is relevant for the real problem, and correctly trading between a good quality solution and a restricted time is mandatory.

Figure 3.6: Execution time for company solution and constructive heuristics.

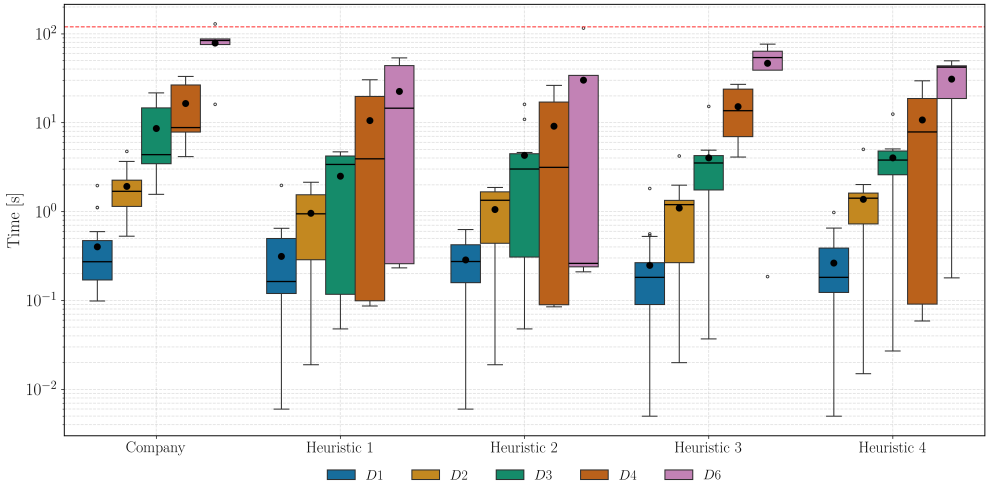


Figure 3.7: Gap percentage for company solution and constructive heuristics.

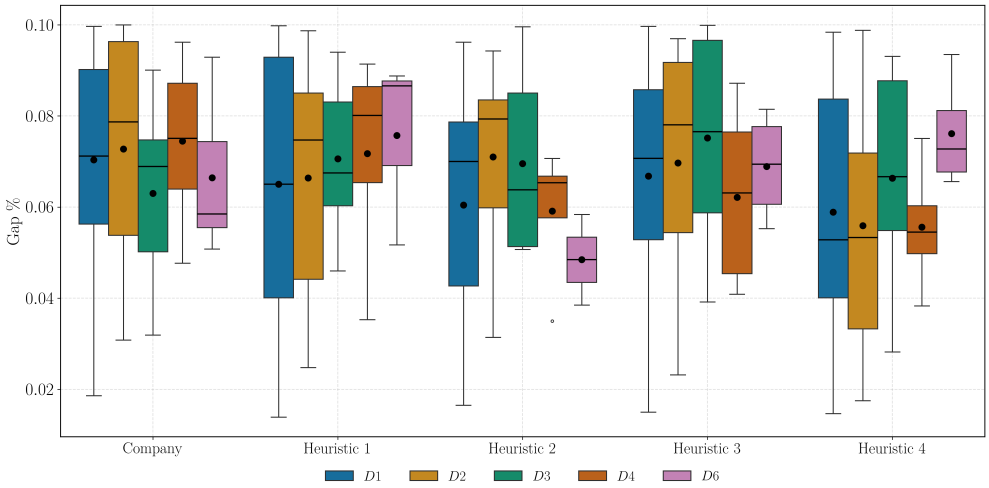


Table 3.16: $D1$ final gap and execution time for heuristics and company solution

Instance	Company		Heuristic 1		Heuristic 2		Heuristic 3		Heuristic 4	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
1	0.060	0.313	0.017	0.224	0.017	0.225	0.015	0.201	0.053	0.145
2	0.086	0.136	0.100	0.105	0.086	0.131	0.063	0.100	0.030	0.139
3	0.055	0.108	0.040	0.129	0.040	0.127	0.059	0.102	0.040	0.128
4	0.091	0.191	0.089	0.499	0.084	0.277	0.081	0.303		0.010
5	0.070	0.183	0.014	0.153	0.017	0.178	0.018	0.176	0.015	0.174
6	0.036	0.290	0.080	0.162	0.054	0.170	0.080	0.162	0.036	0.366
7	0.086	0.256	0.027	0.517	0.096	0.180	0.090	0.179		0.008
8	0.100	1.110		0.008		0.009	0.071	0.199		0.008
9	0.085	0.251		0.008	0.089	0.411	0.054	0.188	0.062	0.200
10	0.100	0.209	0.096	0.200	0.092	0.430	0.063	0.205	0.047	0.356
11	0.055	0.340	0.100	0.186	0.071	0.209	0.052	0.431	0.027	0.491
12	0.088	0.121	0.051	0.114	0.042	0.128	0.071	0.141	0.072	0.129
13	0.059	0.439	0.082	0.159	0.025	0.440	0.036	0.443	0.090	0.157
14	0.019	0.418	0.075	0.435	0.075	0.437	0.098	0.269	0.051	0.535
15	0.072	1.960		0.007		0.007		0.006	0.059	0.396
16	0.059	0.166		0.006		0.006		0.006	0.046	0.329
17	0.042	0.543	0.098	0.651	0.075	0.540	0.096	0.527	0.019	0.573
18	0.095	0.208	0.034	0.496	0.076	0.379	0.077	0.217	0.049	0.522
19	0.092	0.159	0.053	0.147	0.023	0.301		0.005	0.089	0.127
20	0.054	0.482	0.093	0.450	0.043	0.481	0.058	0.539	0.068	0.268
21	0.100	0.099	0.096	0.117	0.045	0.272		0.005		0.005
22	0.072	0.204		0.008	0.070	0.554		0.008	0.098	0.978
23	0.066	0.597	0.064	0.519	0.081	0.361		0.007	0.034	0.467
24	0.056	0.402	0.085	0.164	0.047	0.395	0.045	0.184	0.084	0.118
25	0.023	0.420	0.029	0.533	0.074	0.245	0.032	0.563	0.082	0.226
26	0.092	0.156	0.034	0.134	0.089	0.085	0.100	0.087	0.091	0.089
27	0.066	0.549	0.065	0.579	0.040	0.629	0.098	0.258	0.096	0.190
28	0.092	0.497	0.046	0.561	0.075	0.288	0.099	1.818		0.006
29	0.058	1.114	0.100	1.979	0.047	0.528		0.008	0.052	0.653
30	0.083	0.142	0.057	0.155	0.057	0.155	0.081	0.131	0.084	0.122
Avg	0.070	0.402	0.065	0.314	0.060	0.286	0.067	0.249	0.059	0.264

Table 3.17: *D*2 final gap and execution time for heuristics and company solution

Instance	Company		Heuristic 1		Heuristic 2		Heuristic 3		Heuristic 4	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
1	0.100	0.530	0.079	0.256	0.087	0.964	0.034	0.279		0.015
2	0.054	1.029	0.087	0.319	0.077	0.326	0.077	1.302	0.099	0.449
3	0.100	0.957	0.025	0.942	0.033	0.976	0.067	1.377	0.023	1.003
4	0.096	3.671	0.079	2.145	0.079	1.601	0.095	1.298		0.026
5	0.100	2.338		0.027	0.094	1.732	0.097	0.523	0.086	1.428
6	0.061	1.097	0.045	1.482	0.060	1.340	0.023	1.198	0.018	1.241
7	0.054	1.559	0.041	1.363	0.060	1.432	0.093	1.658	0.035	1.971
8	0.079	4.768		0.019		0.019	0.061	0.967	0.056	1.355
9	0.051	1.923	0.035	1.721	0.031	1.871	0.034	1.986	0.029	2.021
10	0.097	1.685	0.099	0.445	0.080	1.387	0.079	1.289	0.044	1.409
11	0.079	2.187	0.089	1.610	0.087	1.731		0.020	0.067	1.502
12	0.080	2.057		0.021		0.021		0.021	0.051	1.658
13	0.031	1.197	0.063	0.619	0.081	0.554	0.084	0.253	0.067	1.574
14	0.071	2.627	0.085	1.997	0.082	1.870	0.091	4.225		0.023
15	0.040	1.231	0.071	1.404		0.020		0.020	0.098	5.007
Avg	0.073	1.924	0.066	0.958	0.071	1.056	0.070	1.094	0.056	1.379

Table 3.18: *D*3 final gap and execution time for heuristics and company solution

Instance	Company		Heuristic 1		Heuristic 2		Heuristic 3		Heuristic 4	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
1	0.071	1.568	0.086	0.318	0.052	1.058	0.039	1.259		0.027
2	0.032	3.792	0.080	4.696	0.064	4.622	0.100	3.236	0.038	5.002
3	0.067	14.702		0.048		0.059	0.076	4.913	0.071	3.980
4	0.090	4.047	0.054	4.254	0.072	3.706	0.054	3.667	0.028	3.470
5	0.042	14.526		0.048		0.048	0.060	4.295	0.091	12.466
6	0.048	4.687		0.051		0.051		0.051	0.060	3.586
7	0.073	2.995	0.068	3.037	0.051	2.319	0.096	3.353	0.087	2.301
8	0.076	21.655	0.066	4.120	0.098	16.118	0.099	15.255	0.093	5.062
9	0.075	3.357	0.046	3.722	0.051	4.027	0.077	4.212	0.062	4.252
10	0.056	14.962	0.094	4.691	0.100	10.937		0.037		0.038
Avg	0.063	8.629	0.071	2.498	0.070	4.294	0.075	4.028	0.066	4.018

Table 3.19: *D*4 final gap and execution time for heuristics and company solution

Instance	Company		Heuristic 1		Heuristic 2		Heuristic 3		Heuristic 4	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
1	0.075	4.167	0.075	3.910	0.071	3.132	0.063	5.763		0.059
2	0.063	24.320		0.093		0.092	0.087	13.654		0.096
3	0.065	8.790		0.087		0.087	0.073	8.172	0.054	7.846
4	0.088	8.170		0.106		0.085	0.080	4.101	0.055	29.701
5	0.086	7.572	0.035	14.604	0.065	9.070	0.047	27.066	0.038	9.771
6	0.096	33.178	0.091	24.912	0.066	25.090	0.043	23.654	0.075	27.602
7	0.048	28.737	0.085	30.396	0.035	26.360	0.041	24.019		0.086
Avg	0.074	16.419	0.072	10.587	0.059	9.131	0.062	15.204	0.056	10.737

Table 3.20: *D6* final gap and execution time for heuristics and company solution

Instance	Company		Heuristic 1		Heuristic 2		Heuristic 3		Heuristic 4	
	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
1	0.093	16.134	0.052	14.527	0.058	34.003	0.076	39.030		0.180
2	0.051	83.931		0.233		0.239	0.055	53.800	0.068	18.760
3	0.056	129.231		0.260		0.261	0.062	63.758	0.093	43.591
4	0.074	87.672	0.087	43.871		0.210		0.186	0.077	42.100
5	0.059	75.579	0.089	53.549	0.038	115.771	0.082	76.404	0.066	49.630
Avg	0.066	78.509	0.076	22.488	0.048	30.097	0.069	46.636	0.076	30.852

3.7.5 Solving the complete model

This section evaluates the performance of the complete MILP formulation in terms of solution quality (optimality gap) and computational effort (execution time). The analysis is conducted across all the planning horizons to assess the scalability of the model.

Detailed results for each instance are reported in Tables 3.22 to 3.26, whereas the aggregated summary results are provided in Table 3.21.

For the one-day instances shown in Table 3.22, the model demonstrates strong performance. The average optimality gap is approximately 0.11%, indicating that near-optimal solutions are obtained, but the optimality is not always proven, even if several instances are solved to optimality. In some instances, even if part of the easiest set reaches the time limit of 1 hour, with an associated gap below 0.3%, indicating that even for small-scale problems, proving the optimality can be challenging. The average time-to-best suggests that high-quality solutions are typically found early in the search process; this also suggests the difficulty of proving that such solutions are optimal.

A different behavior emerges from the two-day instances as shown by Table 3.23. In this case, all instances reach the time limit, and the average optimality gap increases to 0.46%. This indicates that the problem complexity grows rapidly with the increase of the planning horizon. Despite this, the gap values remain small. The time-to-best values indicate that good incumbent solutions are still identified before termination.

The degradation in performance becomes more pronounced for the three-day instances as shown in Table 3.24. The average optimality gap rises to 2.51%, with some instances exhibiting substantially larger gaps (e.g., exceeding 12%). Additionally, several instances fail to produce feasible solutions within the time limit, highlighting the increasing difficulty of the problem instances.

For the four-day instances shown in Table 3.25, the trend continues, with an increase of the average gap of 3.23% and missing feasible solutions for almost half of the instances. However, the feasible instances yield a bound that is less than 10%.

The final and more complex instances of six day time frame shown in Table 3.26 clearly demonstrate the scalability limitations of the complete model. Only a very limited number of instances produce feasible solutions, and the corresponding optimality gaps are extremely large, with an average exceeding 47%. This indicates that the solver is unable to find a feasible solution when the dimensionality increases and even when a feasible solution is obtained, converging to the optimal solution remains challenging.

Overall, the results reveal that while the complete MILP formulation can be highly effective for short planning horizons up to 1 day, its performance deteriorates rapidly as the problem size increases. This suggests that, for larger instances, alternative approaches such as heuristic methods are necessary to obtain high-quality solutions within reasonable computational times.

Table 3.21: Average results across all instance sets.

Days	Cost	Termination		Best Solution		Difference	
		Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
$D1$	372003.807	1040.186	0.112	801.367	0.153	238.820	0.041
$D2$	744555.608	3600.000	0.460	2404.000	0.493	1196.000	0.033
$D3$	1110226.215	3600.000	2.514	1827.000	2.967	1773.000	0.454
$D4$	1496199.657	3600.000	3.232	1800.000	4.975	1800.000	1.743
$D6$	7148906.300	3600.000	47.661	1330.000	56.750	2270.000	9.089

Table 3.22: Results for the complete model over $D1$ instances.

Instance	Cost	Termination		Best Solution		Difference	
		Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
1	235937.254	45.949	0.097	40.000	0.170	5.949	0.073
2	306516.771	89.221	0.087	89.000	0.090	0.221	0.003
3	236633.407	34.188	0.098	27.000	0.170	7.188	0.072
4	430204.975	3600.000	0.189	3573.000	0.190	27.000	0.001
5	355174.535	381.436	0.098	206.000	0.170	175.436	0.073
6	357975.168	1023.396	0.098	709.000	0.120	314.396	0.022
7	404469.972	1187.677	0.099	1087.000	0.110	100.677	0.011
8	469810.744	923.529	0.100	923.000	0.100	0.529	0.001
9	432463.128	257.692	0.097	185.000	0.130	72.692	0.033
10	430097.667	515.620	0.100	66.000	0.270	449.620	0.170
11	388447.794	1186.900	0.098	615.000	0.140	571.900	0.042
12	281587.474	59.143	0.098	50.000	0.120	9.143	0.022
13	396930.203	428.584	0.100	354.000	0.110	74.584	0.010
14	393806.317	3600.000	0.110	3368.000	0.120	232.000	0.010
15	374079.198	233.680	0.098	145.000	0.150	88.680	0.052
16	346163.134	45.805	0.096	32.000	0.150	13.805	0.054
17	443012.705	3600.000	0.227	3107.000	0.250	493.000	0.023
18	428231.717	3600.000	0.123	1488.000	0.140	2112.000	0.017
19	301635.480	41.864	0.094	40.000	0.110	1.864	0.016
20	369426.985	880.805	0.097	496.000	0.150	384.805	0.053
21	314691.736	35.807	0.085	29.000	0.170	6.807	0.085
22	446582.471	79.585	0.094	72.000	0.130	7.585	0.036
23	390111.754	603.250	0.099	532.000	0.120	71.250	0.021
24	368138.940	1033.926	0.100	899.000	0.130	134.926	0.030
25	439003.117	3600.000	0.114	3067.000	0.120	533.000	0.006
26	289314.699	64.456	0.099	58.000	0.140	6.456	0.041
27	426037.731	3600.000	0.289	2565.000	0.320	1035.000	0.031
28	385056.340	40.349	0.093	36.000	0.180	4.349	0.087
29	425601.411	98.258	0.097	93.000	0.110	5.258	0.013
30	292971.398	314.466	0.100	90.000	0.220	224.466	0.120
Avg	372003.807	1040.186	0.112	801.367	0.153	238.820	0.041

Table 3.23: Results for the complete model over *D2* instances.

Instance	Cost	Termination		Best Solution		Difference	
		Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
1	543234.286	3600.000	0.305	2680.000	0.340	920.000	0.035
2	668474.634	3600.000	0.590	2858.000	0.600	742.000	0.010
3	712918.455	3600.000	0.347	1958.000	0.390	1642.000	0.043
4	875536.505	3600.000	0.637	2632.000	0.640	968.000	0.003
5	863174.158	3600.000	0.430	2237.000	0.460	1363.000	0.030
6	670803.862	3600.000	0.389	2026.000	0.500	1574.000	0.111
7	792347.015	3600.000	0.694	2330.000	0.760	1270.000	0.066
8	718121.033	3600.000	0.347	3502.000	0.350	98.000	0.003
9	871525.698	3600.000	0.458	1134.000	0.460	2466.000	0.002
10	670721.681	3600.000	0.377	3415.000	0.390	185.000	0.013
11	761740.124	3600.000	0.357	2961.000	0.360	639.000	0.003
12	758873.428	3600.000	0.527	1501.000	0.640	2099.000	0.113
13	728740.581	3600.000	0.269	2515.000	0.290	1085.000	0.021
14	812552.560	3600.000	0.718	2367.000	0.760	1233.000	0.042
15	719570.102	3600.000	0.459	1944.000	0.460	1656.000	0.001
Avg	744555.608	3600.000	0.460	2404.000	0.493	1196.000	0.033

Table 3.24: Results for the complete model over *D3* instances.

Instance	Cost	Termination		Best Solution		Difference	
		Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
1	781197.420	3600.000	0.721	1066.000	0.730	2534.000	0.009
2	1153901.095	3600.000	1.442	1620.000	1.520	1980.000	0.078
3		3600.000					
4	1102402.339	3600.000	0.696	2933.000	0.700	667.000	0.004
5		3600.000					
6	1241867.738	3600.000	2.440	3210.000	2.500	390.000	0.060
7	986140.517	3600.000	0.594	1365.000	3.750	2235.000	3.156
8	1209831.160	3600.000	1.125	2464.000	1.130	1136.000	0.005
9	1157959.918	3600.000	0.873	1863.000	0.910	1737.000	0.037
10	1248509.532	3600.000	12.220	95.000	12.500	3505.000	0.280
Avg	1110226.215	3600.000	2.514	1827.000	2.967	1773.000	0.454

Table 3.25: Results for the complete model over $D4$ instances.

Instance	Cost	Termination		Best Solution		Difference	
		Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
1	1225961.746	3600.000	1.909	2825.000	2.220	775.000	0.311
2		3600.000					
3	1549258.630	3600.000	1.730	1221.000	3.370	2379.000	1.640
4		3600.000					
5	1639078.450	3600.000	6.496	1377.000	8.720	2223.000	2.224
6		3600.000					
7	1570499.800	3600.000	2.794	1777.000	5.590	1823.000	2.796
Avg	1496199.657	3600.000	3.232	1800.000	4.975	1800.000	1.743

Table 3.26: Results for the complete model over $D6$ instances.

Instance	Cost	Termination		Best Solution		Difference	
		Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
1		3600.000					
2		3600.000					
3	2740593.033	3600.000	14.044	1370.000	32.200	2230.000	18.156
4	11557219.567	3600.000	81.278	1290.000	81.300	2310.000	0.022
5		3600.000					
Avg	7148906.300	3600.000	47.661	1330.000	56.750	2270.000	9.089

3.7.6 ALNS Evaluation

The main parameters of the ALNS implementation are set as follows: a time limit of 600 seconds, an initial temperature $T_0 = 1000$, and a cooling rate $cr = 0.9$. The scoring parameters are set to $\eta_{\text{best}} = 10$, $\eta_{\text{incumbent}} = 5$, and $\eta_{\text{accepted}} = 2$. Given the inherent randomness of the method, three independent runs were performed for each instance, and the results were averaged to ensure a reliable assessment. The results are presented in Figures 3.8 to 3.12,while detailed results for each instance are reported Tables 3.27 to 3.31. The reported percentage differences are obtained by averaging the ALNS results over the three independent runs performed for each instance. Each figure reports the percentage deviation of the ALNS solution with respect to three reference solutions: the optimized company solution, the constructive heuristic used as the initial feasible solution, and the complete MILP model.

In addition, optimality boundaries are set at ±0.1%. Whenever the percentage difference lies within this interval, the corresponding solutions are considered equivalent, even if their objective values are not identical, due to the tolerance adopted in the analysis.

Starting from the simplest case, namely the one-day planning horizon, the effectiveness of the ALNS approach is already evident. On average, the method improves the optimized company solution by approximately 2.91%, while the improvement over the initial constructive heuristic is more limited (0.70%), suggesting that the starting solution is already of good quality. When compared with the MILP solution, the performance gap is considerably smaller. In some instances, the MILP still provides better solutions. However, even when the ±0.1% tolerance is taken into account, ALNS reaches optimality in most cases. Only a limited number of instances fall outside this optimality range, confirming that ALNS is able to produce near-optimal solutions consistently for short planning horizons. Proving the effectiveness of the method.

As the time horizon increases to two days, the overall behavior remains consistent. The improvement over the company solution is still significant, while the gain over the constructive heuristic remains moderate. With respect to the MILP, the gap increases slightly in several instances, suggesting that obtaining equivalent solutions becomes more challenging for the heuristic as the problem grows in size. Nevertheless, the average difference with respect to the MILP remains close to the optimality range.

For the three-day instances, the same general trend is observed. ALNS maintains a clear and stable advantage over the company solution and continues to improve upon the constructive heuristic. At the same time, the comparison with the MILP shows, for the first time, an average improvement also with respect to the complete model, with only one instance in which ALNS does not reach optimality. Overall, most of the results remain very close to the optimal solutions, highlighting the capability of the method as the planning horizon increases.

In the four-day setting, the higher problem complexity leads to larger deviations in some cases, especially when compared with the MILP. Although ALNS still provides great improvements over both the company solution and the constructive heuristic, the assessment with respect to the MILP progressively shifts from a purely cost-based comparison to a feasibility-based one. Indeed, for this set of instances, the MILP starts to fail in finding feasible solutions, making a direct comparison of objective values less informative.

Finally, for the six-day instances, the complexity of the problem further amplifies these effects. ALNS continues to outperform the company solution and to improve upon the initial heuristic. In this case, the comparison with the MILP indicates that the heuristic becomes the only practical approach for obtaining high-quality solutions, as it clearly outperforms the exact model on the hardest instances.

Overall, the results show that ALNS is a highly effective method for this problem. It consistently provides significant improvements over the company solutions, achieves near-optimal performance in many cases, and remains capable of producing high-quality solutions for the most difficult instances, while still being competitive on shorter planning horizons.

Figure 3.8: *D*1 ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic.

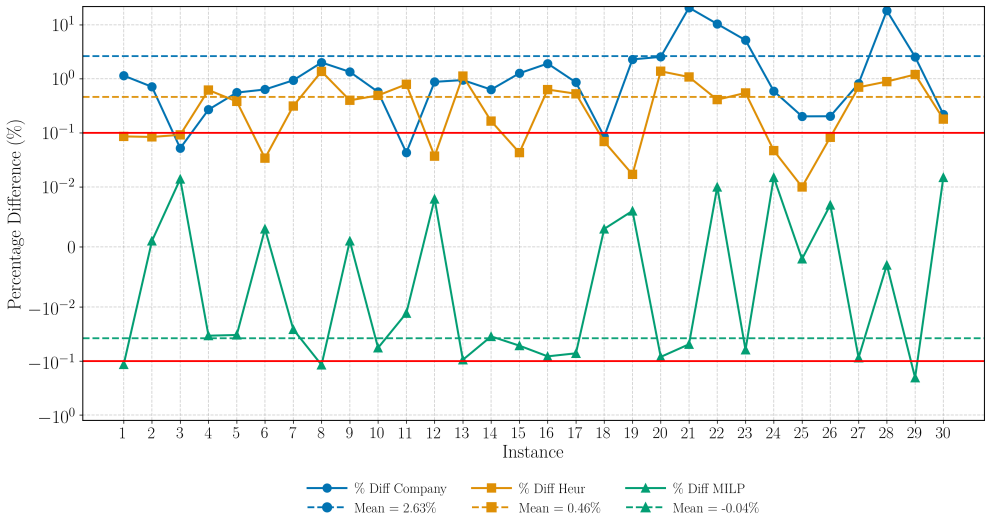


Figure 3.9: *D*2 ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic.

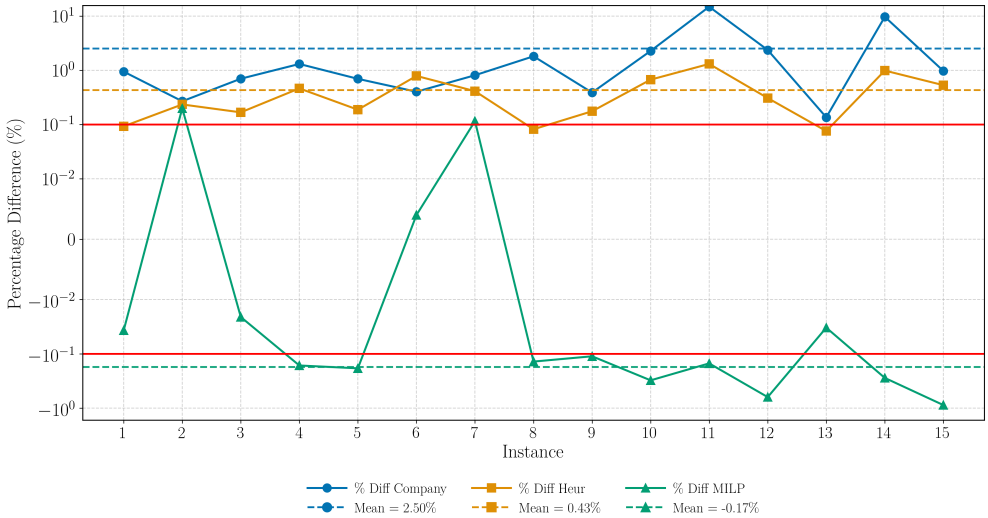


Figure 3.10: *D*3 ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic.

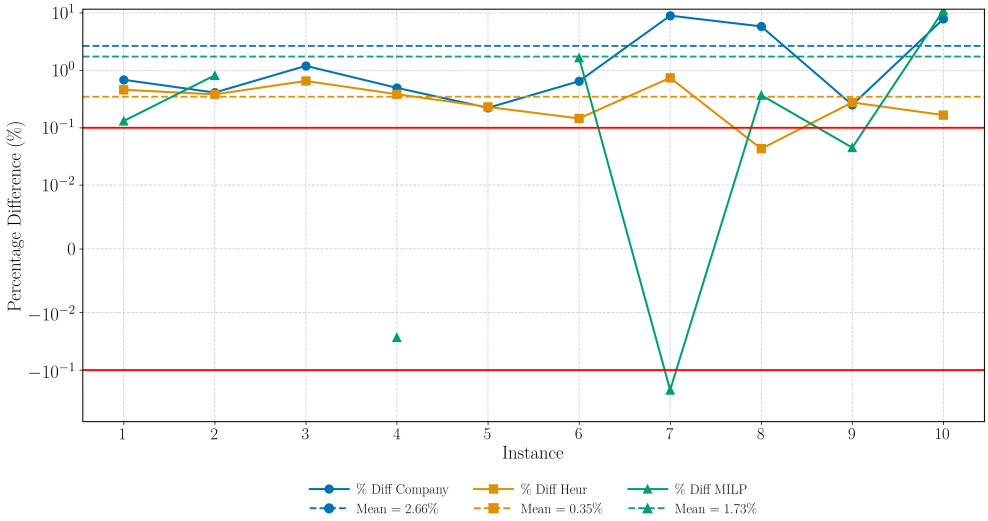


Figure 3.11: *D*4 ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic.

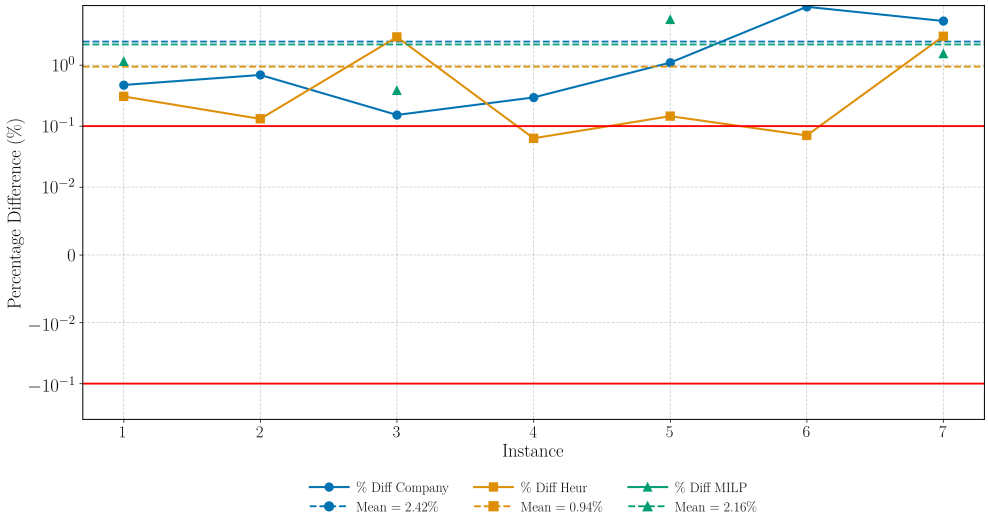
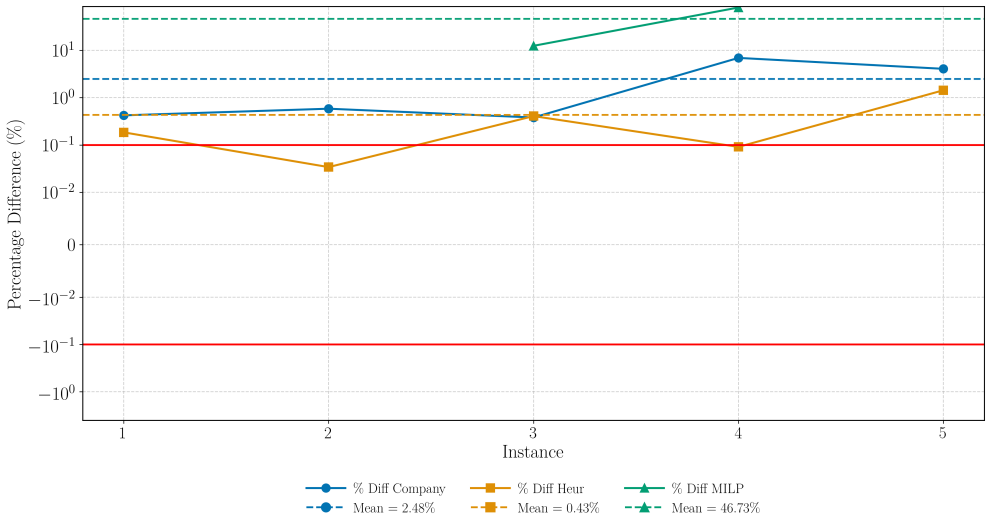


Figure 3.12: *D6* ALNS results: percentage objective differences with respect to the company solution, the MILP model, and the initial constructive heuristic.



Another important aspect, regarding the resolution process of the ALNS, concerns how the destroy and repair operators influence the search performance. Some relevant statistics are reported in Figure 3.13, regarding the repair operator.

A key consideration for the repair phase is the need to solve a restricted MILP through a solver, which inherently requires a proper configuration. To obtain feasible solutions within a limited time. Some specific hyperparameters have been defined. In particular, the maximum allowed time for a single repair operation has been fixed to 20 seconds. This choice is fundamental, since without a time limit, a single repair step could consume the entire computational budget without effectively progressing the search and the method.

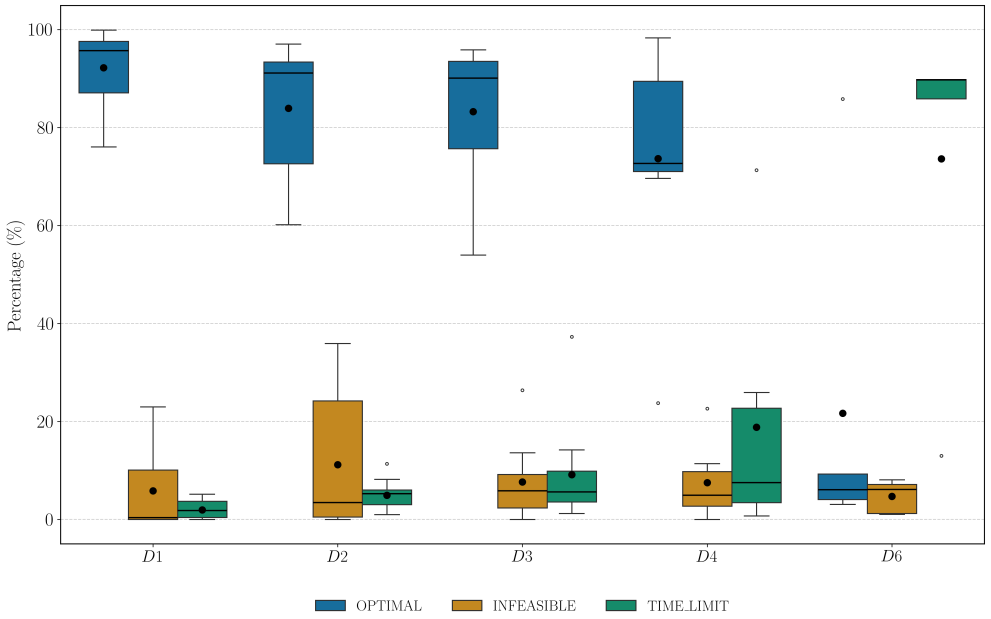
The decision to adopt such a short execution time is primarily motivated by the results obtained in the constructive heuristic phase, where infeasibility is typically detected within a short time. Moreover, the repair phase does not require the same level of optimality as the constructive phase. Even suboptimal solutions can still be valuable, as they contribute to the diversification of the search and support the convergence of the method by enabling the exploration of different regions of the solution space.

The Figure 3.13 reports the percentage of repair operations that terminate with an optimal solution, an infeasible outcome, or by reaching the imposed time limit, for each instance set. From the results, it can be observed that for instances $D1$ to $D3$, more than 80% of the repair operations end with an optimal solution. In the same range, infeasibility does not show a consistent trend, although a noticeable increase is observed for the $D2$ instances. The proportion of iterations reaching the time limit remains relatively low, generally below 15%.

For the more complex instances in $D4$, due to the increased number of heats and the corresponding rise in problem complexity, a higher percentage of repair operations reaches the time limit, with values close to 20%. The percentage of infeasible solutions remains comparable to the previous cases, resulting in a reduction of optimal outcomes to below 80%.

Finally, for the most challenging instances in $D6$, the same behavior is further emphasized. Many repair operations reach the time limit, while only a subset achieves optimality within the allowed time. However, this does not negatively impact the overall performance of the method. As previously discussed, the ALNS framework is able to effectively exploit these repair solutions, which are sufficient to guide the search and maintain good convergence properties.

Figure 3.13: ALNS repair operator performance.



Another relevant aspect in the analysis of the ALNS performance concerns the impact of the destroy operators on the quality of the search. In particular, Figure 3.14 reports, for each destroy operator and instance set, the percentage of times the application of the operator leads to an improvement of the current best objective value within the ALNS framework. Therefore, the figure provides a direct measure of how effective each destroy strategy is in generating a new best objective.

Starting from the random destroy operator, `RA_4` appears to provide the lowest improvement rates among all the tested strategies. This behavior was expected, since the random elimination of arcs introduces diversification without exploiting any structural information from the current solution. As a consequence, although this operator can help to escape local minima, it is less effective in consistently producing improvements.

A different behavior can be observed for the highest-cost-node operators. Both `HCN_2` and especially `HCN_3` achieve significantly higher improvement percentages across all instance groups. This confirms that focusing the destruction phase on the most expensive nodes is an effective strategy, as it directly targets the portions of the solution that contribute the most to the objective value. Between the two, `HCN_3` generally provides the highest improvement rates, especially for medium-sized instances, indicating that removing a limited number of critical nodes allows the repair phase to reconstruct better solutions.

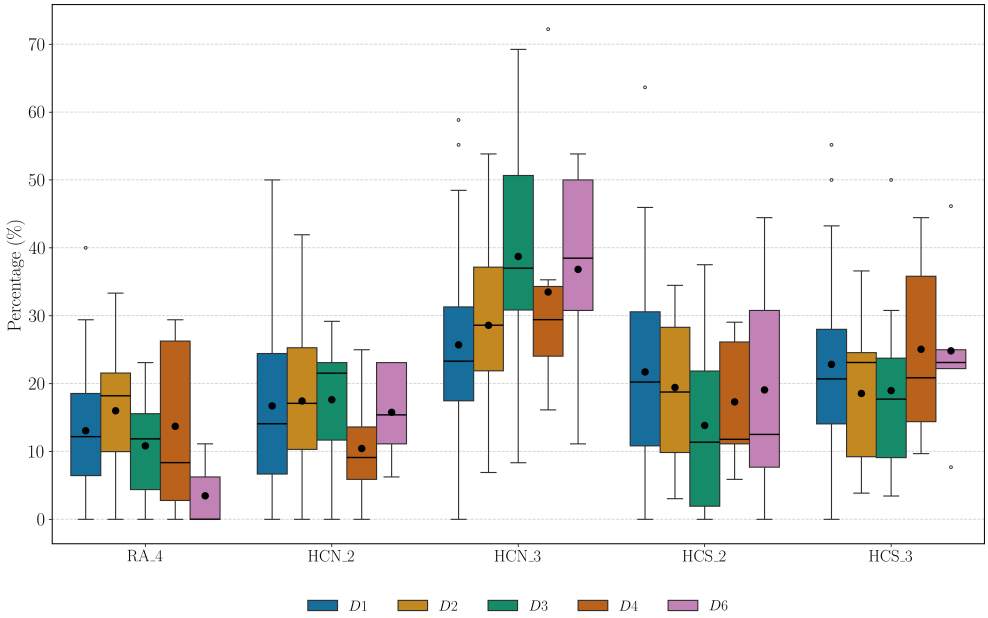
The highest-cost-subsequence operators, `HCS_2` and `HCS_3`, exhibit an intermediate behavior. Their improvement percentages are higher than those of the random operator, but generally lower than those obtained by `HCN_3`. This suggests that targeting costly subsequences is beneficial, as it preserves part of the solution structure while modifying critical segments. However, compared to node-based removal, it appears less effective in enabling large improvements, likely due to its more localized impact.

With respect to the instance sets, the highest improvement rates are observed for `HCN_3`,

particularly in $D3$, where improvements are significantly greater than for the other operators. For more complex instances such as $D4$ and $D6$, the differences among operators become less pronounced, although cost-based strategies still outperform the random one. This behavior reflects the increased difficulty of improving solutions in larger instances, where the effectiveness of the repair phase becomes more critical.

Overall, the results confirm that informed destroy operators that exploit cost information are clearly preferable to purely random strategies, as they more consistently lead to improvements in the best objective value during the search. Moreover, an increase in the number of nodes to be removed seems to improve the effectiveness of cost-based methods.

Figure 3.14: ALNS destroy operator performance.



The following tables summarize the framework's performance in terms of solution cost, reporting the best (minimum), worst (maximum), and average values obtained. In addition, they provide the average time required to reach the best solution, along with the corresponding percentage variation.

Table 3.27: ALNS cost and time summary for benchmark set $D1$

Instance	Gurobi	ALNS			Time (s)	
		Min	Max	Avg	To Best	Δ to Best (%)
1	235937.254	236198.723	236229.851	236210.943	198.179	53.505
2	306516.771	306508.461	306517.326	306512.459	252.606	44.954
3	236633.407	236597.116	236601.244	236599.565	321.581	80.439
4	430204.975	430239.201	430571.598	430351.920	536.564	4.792
5	355174.535	355233.211	355375.899	355290.869	174.271	74.968
6	357975.168	357955.426	357969.989	357964.211	335.265	81.894
7	404469.972	404426.019	404799.995	404575.841	316.476	41.717
8	469810.744	470152.110	470632.551	470364.122	486.518	18.954
9	432463.128	432420.865	432485.297	432458.993	265.439	52.287
10	430097.667	430085.846	430611.002	430341.203	376.181	32.169
11	388447.794	388496.798	388505.740	388499.807	414.294	10.179
12	281587.474	281562.798	281567.890	281565.819	190.930	85.090
13	396930.203	397077.094	397744.042	397308.223	304.227	20.015
14	393806.317	393771.396	394103.338	393942.213	505.780	8.018
15	374079.198	374239.209	374330.131	374272.303	74.948	82.120
16	346163.134	346291.719	346752.501	346447.054	113.143	25.090
17	443012.705	442914.780	443544.131	443333.627	188.014	61.163
18	428231.717	428214.190	428219.721	428217.247	374.776	52.644
19	301635.480	301613.199	301623.102	301617.511	318.953	45.132
20	369426.985	369579.735	369955.052	369737.985	545.707	6.541
21	314691.736	314784.463	314898.820	314846.205	300.130	64.557
22	446582.471	446535.168	446537.216	446536.533	252.584	99.101
23	390111.754	390184.870	390640.007	390354.545	381.445	41.022
24	368138.940	368045.535	368104.811	368083.079	168.610	43.567
25	439003.117	439010.468	439014.561	439012.238	2.056	57.797
26	289314.699	289291.074	289302.277	289295.184	499.351	18.125
27	426037.731	426030.250	426939.362	426409.175	374.599	73.369
28	385056.340	385045.622	385087.607	385067.615	225.150	25.024
29	425601.411	425852.887	427595.069	426467.981	435.117	38.098
30	292971.398	292917.309	292935.218	292927.677	329.222	53.925

Table 3.28: ALNS cost and time summary for benchmark set *D*2

Instance	Gurobi	ALNS			Time (s)	
		Min	Max	Avg	To Best	Δ to Best (%)
1	543234.286	543339.707	543495.924	543434.886	541.324	13.245
2	668474.634	666877.310	667465.063	667138.504	571.560	2.690
3	712918.455	713036.036	713110.336	713068.938	343.761	79.690
4	875536.505	876900.334	877121.166	876974.458	377.564	9.859
5	863174.158	863971.849	865835.114	864767.206	219.858	106.825
6	670803.862	670474.131	671339.953	670778.060	404.412	35.052
7	792347.015	791311.410	791527.196	791431.845	495.116	10.560
8	718121.033	718771.793	719716.136	719122.677	301.669	88.017
9	871525.698	872023.252	872927.368	872496.773	484.300	23.202
10	670721.681	672197.997	673090.897	672788.641	361.911	54.161
11	761740.124	761435.672	764765.755	762881.869	511.127	14.396
12	758873.428	763191.655	763920.968	763630.314	559.183	4.532
13	728740.581	728960.952	729016.011	728982.673	289.875	63.355
14	812552.560	814072.733	816197.897	814810.489	532.030	10.522
15	719570.102	725865.317	725943.617	725897.956	525.845	15.891

Table 3.29: ALNS cost and time summary for benchmark set *D*3

Instance	Gurobi	ALNS			Time (s)	
		Min	Max	Avg	To Best	Δ to Best (%)
1	781197.420	780034.235	780366.472	780176.561	206.688	130.542
2	1153901.095	1144034.607	1145132.072	1144459.271	491.672	27.764
3		1310743.466	1312215.172	1311305.536	580.628	2.896
4	1102402.339	1102451.302	1103090.644	1102701.922	465.381	25.642
5		1173316.746	1175173.149	1174449.546	504.036	9.581
6	1241867.738	1220780.436	1221703.742	1221329.660	540.338	2.822
7	986140.517	987950.406	988803.888	988343.243	474.397	20.135
8	1209831.160	1205324.335	1205412.371	1205353.680	31.504	149.867
9	1157959.918	1156331.643	1158637.549	1157436.978	459.506	26.307
10	1248509.532	1110029.881	1110103.420	1110070.626	581.476	2.865

Table 3.30: ALNS cost and time summary for benchmark set *D*4

Instance	Gurobi	ALNS			Time (s)	
		Min	Max	Avg	To Best	Δ to Best (%)
1	1225961.746	1211176.599	1212892.755	1211860.076	327.485	88.105
2		1594626.135	1596305.750	1595446.525	481.744	18.152
3	1549258.630	1542168.303	1544405.671	1543327.842	479.677	28.027
4		1524797.123	1526914.434	1525949.752	226.613	107.563
5	1639078.450	1545916.576	1549202.975	1547530.826	375.514	74.077
6		1526760.906	1526768.795	1526763.536	226.181	49.028
7	1570499.800	1544497.161	1549414.148	1546323.849	460.896	3.382

Table 3.31: ALNS cost and time summary for benchmark set *D*6

Instance	Gurobi	ALNS			Time (s)	
		Min	Max	Avg	To Best	Δ to Best (%)
1		1926530.074	1927852.334	1927404.435	354.262	61.612
2		2420081.518	2420298.121	2420194.811	192.786	61.114
3	2740593.033	2397276.382	2397838.010	2397509.374	392.123	20.669
4	11557219.567	2199628.586	2202633.596	2201626.175	255.321	98.612
5		2269401.126	2269464.958	2269440.226	244.762	47.899

3.8 Conclusions

This chapter addressed for the first time the Multi-Heat Scrap Selection and Sequencing Problem with Liquid Heel (MHSSP-LH), a complex optimization problem arising in steelmaking operations, in which heat sequencing decisions account for both production raw material selection (scraps, ferroalloys and their energy consumption) and liquid heel management. By explicitly modeling these aspects, the problem provides a more realistic representation of industrial practice, innovating the current literature with a new formulation.

A mixed-integer linear programming (MILP) formulation was proposed to model the problem. However, due to the intrinsic computational complexity of the problem and the scale of real-world instances, exact approaches alone were shown to be insufficient for practical deployment, over multiple day timeframe. To overcome this limitation, four constructive heuristics, based on different sorting methods were developed to quickly generate feasible solutions, providing both effective initialization for both standalone approaches and complex metaheuristic through which large-scale instances can be solved.

Building on these foundations, an Adaptive Large Neighborhood Search (ALNS) metaheuristic was designed, combining problem-specific destroy and repair operators with adaptive selection mechanisms. The proposed ALNS demonstrated strong performance in navigating the solution space, effectively balancing intensification and diversification.

Extensive computational experiments, based on real industrial data, highlighted the effectiveness of the proposed approaches. In particular, the ALNS consistently outperformed both the baseline company solutions and the constructive heuristics, achieving significant cost reductions while maintaining computational efficiency. Moreover, the results showed that incorporating liquid heel considerations into the sequencing decisions leads to operational improvements, confirming the practical relevance of the proposed model.

Overall, this work contributes from both a modeling and algorithmic perspective. It introduces a novel and realistic optimization problem, proposes a comprehensive solution framework, and demonstrates its applicability in an industrial context. Nevertheless, the proposed framework relies on deterministic assumptions regarding input data and considers a static planning horizon, making it primarily suitable for offline tactical planning rather than highly dynamic production environments. In practice, uncertainties in scrap composition, material availability, and unexpected operational disruptions may affect production decisions and require frequent schedule adaptations. Future research should therefore investigate the integration of stochastic or robust optimization techniques to explicitly model uncertainty, as well as the development of rolling-horizon and real-time decision-support frameworks capable of dynamically updating production plans as new information becomes available. These extensions would further enhance the practical applicability of the proposed methodology while preserving its modeling flexibility.

Chapter 4

A Comprehensive Hybrid Flow Shop Scheduling Model for Steelmaking and Ingot Casting

Contents

4.1 Introduction . . . 75
4.1.1 Steelmaking Process Based on the Electric Arc Furnace . . . 76
4.1.2 From physical process to mathematical models . . . 79
4.2 Literature Review . . . 80
4.3 Problem description . . . 84
4.3.1 Modeling Assumptions . . . 87
4.4 Mathematical model . . . 88
4.4.1 The Hybrid Flow Shop Formulation . . . 88
4.4.2 Ingot Management . . . 91
4.4.3 Inter-machine eligibility . . . 93
4.4.4 Valid Inequalities . . . 93
4.4.5 Symmetry Breaking . . . 94
4.4.6 Lower bounds . . . 95
4.4.7 Upper bounds . . . 96
4.4.8 Basic and Advanced models . . . 97
4.5 Experimental Results . . . 98
4.5.1 Instances generation . . . 99
4.5.2 Analysis of the strengthening mechanisms . . . 101
4.5.3 Upper bounds . . . 103
4.5.4 Lower bounds . . . 106
4.5.5 Models comparison . . . 108
4.6 Conclusions . . . 113

In steelmaking, production scheduling plays a critical role in ensuring process continuity and efficient resource utilization, particularly due to the absence of intermediate buffering

and the tight coupling between stages. While extensive research has been conducted on steelmaking through continuous casting, the ingot casting process remains comparatively underexplored despite its industrial relevance.

This chapter introduces a novel Hybrid Flow Shop for Steelmaking and Ingot Casting (HFS-SIC). The proposed model captures key real-world features that are often overlooked or only partially addressed in the literature, including no-wait constraints, stage skipping, machine eligibility and inter-stage compatibility, transportation times, and the allocation of shared auxiliary resources such as ladles and ingots. In addition, the model explicitly incorporates complex spatial and capacity constraints arising from solidification pits and plant layout dependencies.

The problem is formulated as a mixed-integer linear programming (MILP) model with the objective of minimizing the makespan. The primary contribution lies in the comprehensive modeling of the production system, extending classical hybrid flow shop formulations to better capture the operational constraints of ingot casting environments. Furthermore, we introduce three lower bounds and two upper bounds for the defined problem. The model is also strengthened through the incorporation of additional valid inequalities and symmetry-breaking constraints.

The effectiveness of the proposed model is evaluated through computational experiments based on real industrial data. Results highlight the impact of the enriched modeling framework on schedule feasibility and performance, and provide insights into the gap between theoretical formulations and practical production settings.

4.1 Introduction

Steel production is a complex industrial process consisting of many stages that are highly connected. The process itself can be split into different sub-problems, some of which have already been analyzed, in Chapter 2 and Chapter 3.

This chapter addresses the scheduling and planning aspects of the overall process, building upon the foundations established in Chapter 2 and Chapter 3. These chapters represent the initial stages of the complete decision-making framework. In particular, Chapter 3 focuses on the operational problem of scrap selection for a sequence of different steel castings, determining the optimal composition of available materials to satisfy process and quality constraints. In contrast, Chapter 2 adopts a higher-level perspective, addressing strategic decisions related to scrap management, including which materials should be maintained in inventory to support future production requirements.

The scheduling aspect, due to its underlying complexity, constitutes the primary production bottleneck. This complexity arise from the need to coordinate and allocate all available resources, in particular, industrial machinery across the plant, while ensuring their synchronization under strict temporal constraints and complying with steelmaking-specific operational requirements. As highlighted in the survey by (Lee et al., 2024), these significantly complicate the scheduling task, making it critical for the overall production system. In order to understand why it is so complicated, we first need to describe the process.

Steel production can be broadly categorized into two main approaches based on the melting technology employed and the associated input materials: the Basic Oxygen Furnace (BOF) process and the Electric Arc Furnace (EAF) process. The BOF process primarily relies on iron produced from iron ore in a blast furnace, supplemented by a smaller proportion of steel scrap. In contrast, the EAF process mainly uses steel scrap as its primary input material. This work, in line with previous studies, focuses on the EAF process, also referred to as

secondary steelmaking, examining the machinery involved and the operational constraints that enable the production of new steel from recycled scrap.

4.1.1 Steelmaking Process Based on the Electric Arc Furnace

In the electric arc furnace (EAF) process, steel is produced by melting solid scrap into liquid steel, which is then cast into solid semi-finished products. The main difference compared to primary steelmaking lies in the use of scrap as the input material and electricity as the primary energy source for melting. Although this process accounts for only about 30% of global production as explained by World Steel Association (2026), it offers significant potential for process optimization.

To improve efficiency and reduce heat losses, as discussed in Chapter 3, part of the material is kept in a molten state across multiple stages, thereby lowering the energy required for subsequent melting phases. As a result, the process forms a highly interconnected system in which thermal constraints, strict metallurgical requirements, and strong interdependencies between stages must be carefully managed. In addition, auxiliary resources and material-handling operations play a critical role. Despite the fact that each unit operation is well understood in isolation, the overall production flow behaves as an integrated system, where local delays or deviations can quickly propagate throughout the entire plant.

The steelmaking process, as described in detail in Chapter 3, begins in the scrapyard with the crucial step of scrap selection. Depending on the type and quality of steel to be produced, different categories of scrap are carefully chosen. These selected scrap materials are then loaded into baskets, which are subsequently transported to the first stage of the steelmaking process. This initial step is essential, as the composition of the scrap directly influences the properties and quality of the final steel product.

The production process, visually described in Figure 4.1, starts in the EAF with the charging and melting of scrap. Electrical energy, along with supplementary chemical energy from oxygen and methane injectors, is used to fully melt the scrap and reach the target tapping temperature. Throughout the melting phase, operators and control systems monitor the process using temperature measurements and chemical analyses. Once the desired conditions are met, the molten steel is tapped into a ladle. The ladle is a vessel designed to contain and transport molten steel between the different stages of the production process. It enables the material to be processed sequentially by all downstream units, as, except for the EAF, each machine operates on the steel while it is contained within the ladle. In practice, all machines can be interpreted as specialized industrial covers, for the ladles, equipped with dedicated supporting structures that interface with it. The specific configuration and function of these covers distinguish one processing unit from another. This ladle-based handling system is essential to minimize unnecessary material and thermal losses, which will occur if at every stage, we need to transfer the molten steel. For this reason, ladles represent a critical resource within the process and must be managed and scheduled appropriately. In industrial practice, the furnace is not completely emptied during tapping: a residual liquid heel is typically retained to enhance thermal continuity, reduce the energy required for subsequent heats, and lower overall production costs if correctly managed. This creates a natural coupling between consecutive heats, as the state of the furnace at the end of one heat directly affects the efficiency and timing of the next. The role of the liquid heel, as well as its impact on production scheduling and overall costs, has been described in detail in Chapter 3.

After tapping, the ladle becomes the main carrier of the molten steel. It is transported by cranes to the secondary metallurgy stage, typically carried out in one of the available ladle

furnaces (LFs). In the LF, alloying additions and refining operations are performed to achieve the required chemical composition and to adjust the temperature for subsequent processing.

Since different steel grades require specific alloy compositions, treatment intensities, and temperature targets, both the duration and the operational profile of the LF treatment can vary significantly from one heat to another. During this stage, repeated sampling and temperature measurements are often needed to ensure that the metallurgical specifications are met before the steel is released to the next phase.

Following secondary refining, the steel may undergo vacuum treatment, such as vacuum degassing (VD) or vacuum oxygen decarburization (VOD), depending on the requirements of the steel grade. This stage aims to remove dissolved gases and impurities and, in some cases, to support decarburization and improve steel cleanliness.

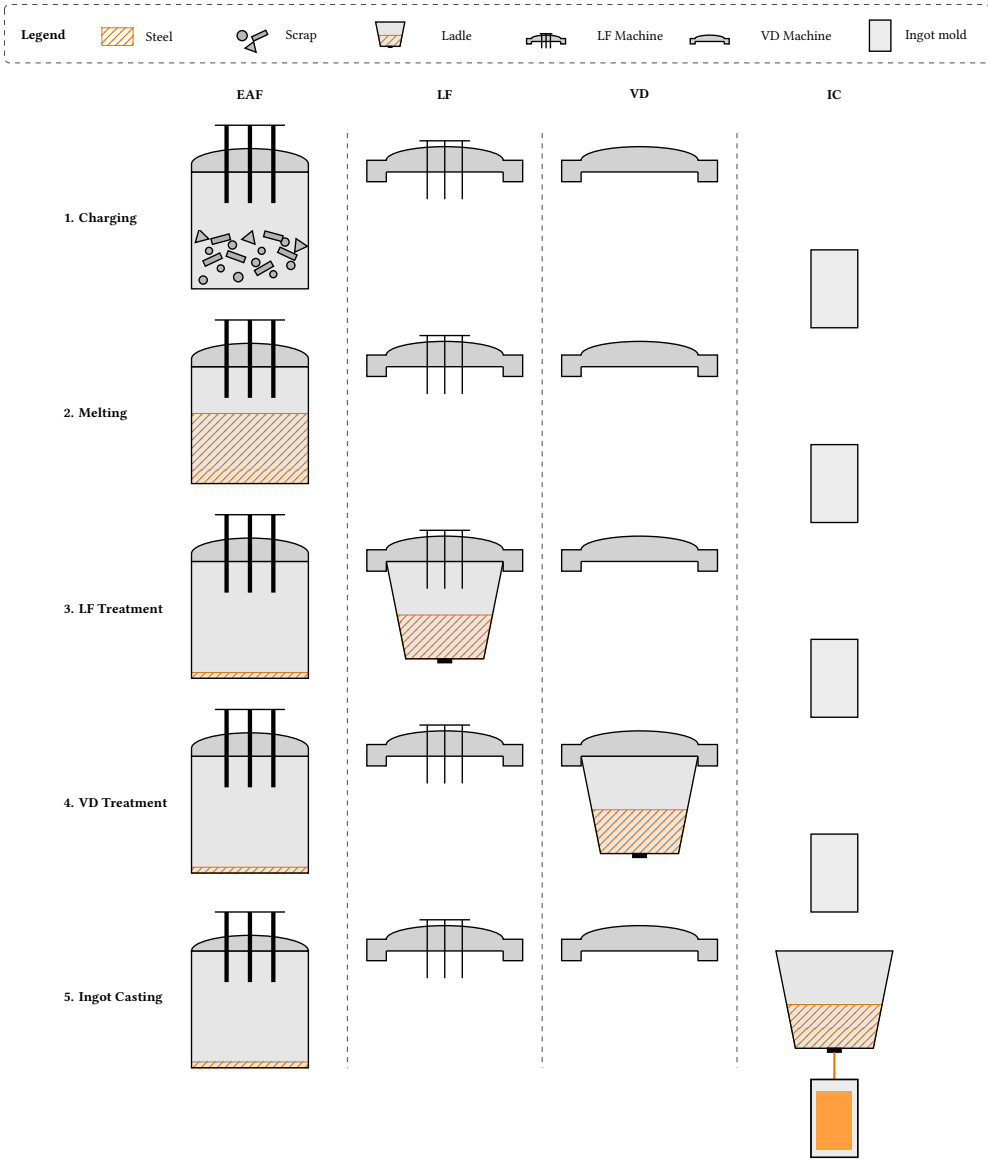
From a process perspective, vacuum treatment increases sensitivity to thermal losses, as maintaining an adequate temperature window becomes critical. Excessive cooling can negatively affect subsequent stages and, in extreme cases, require corrective actions. For high-quality steel grades, additional processing steps may be necessary: vacuum treatment can be followed by reheating in the LF, and further vacuum cycles may be required. As a result, the EAF route does not always follow a single linear sequence. Instead, alternative process paths may arise depending on the product type, operational practices, and thermo-metallurgical conditions.

The next stage of the process is casting, a crucial phase that can be carried out using two main approaches: (i) Continuous Casting and (ii) Ingot Casting.

Continuous casting (CC) is the dominant technique in modern steelmaking. In this process, molten steel is continuously poured from a ladle into a tundish and then into a water-cooled mold, where it begins to solidify. The partially solidified strand is continuously withdrawn from the mold and further cooled and shaped as it moves through the casting machine, eventually being cut into semifinished products such as slabs, billets, or blooms. Compared to traditional methods, continuous casting offers higher productivity, improved yield, better surface quality, and more uniform internal structure, making it the preferred choice for most steel grades.

In contrast, ingot casting (IC) involves pouring molten steel from the ladle into a series of individual molds. The pouring time depends on the steel grade and the selected operational practice. Unlike continuous casting, the process does not end once pouring is complete. The ingots must remain in the molds for a cooling period, which varies depending on the steel composition and mold characteristics, before they can be removed. The solidification stage represents the final step of the process. As a result, ingot molds and solidification pits remain occupied for extended periods, reducing their availability for subsequent heats and lowering overall process efficiency.

Figure 4.1: Steelmaking process.



As a result, casting operations and solidification are strongly coupled stages: decisions made during casting directly influence the availability and utilization of solidification pits, and, conversely, constraints in the solidification stage affect casting choices. Consequently, the effective casting capacity is determined not only by the pouring time, but also by the solidification duration and the associated handling constraints.

In parallel with the main metallurgical flow, auxiliary resource systems introduce additional coupling effects. Ladles circulate throughout the plant and must meet operational readiness requirements, such as preheating, to prevent excessive heat losses during tapping and transfer. Material handling relies on shared cranes and transfer routes, which can be-

come bottlenecks when multiple operations compete for movement or when spatial interference occurs, particularly near casting areas. Furthermore, ingot casting imposes strong spatial constraints due to the limited availability of pit positions and compatible molds. Not all molds can be placed in every pit, adding a layer of complexity to resource allocation. As a result, the plant operates as a system with extended resource holding times, where the availability of space and equipment becomes an integral part of the overall metallurgical process.

Overall, the EAF steelmaking route is characterized by a combination of (i) liquid-state material continuity, (ii) grade-dependent processing routes and treatment durations, (iii) temperature sensitivity that limits idle times between stages, and (iv) auxiliary resource and spatial constraints related to ladles, handling equipment, molds, and casting pits. These features make the process highly coupled and interconnected: operational decisions at one stage are rarely local, as they directly affect the feasibility of downstream operations and the availability of critical supporting resources. This interdependence represents the main source of complexity in steelmaking and casting systems and highlights the need for structured production planning and control frameworks, including the use of mathematical models to support plant managers in effectively managing the overall process.

4.1.2 From physical process to mathematical models

The physical characteristics described in the previous section not only define the metallurgical aspects of production but also directly shape the structural properties of the production system. When abstracted at a sufficiently high level, the steelmaking process exhibits a recurring and well-defined structure. Each heat must pass sequentially through a series of processing stages (EAF, LF, VD/VOD, IC/CC, SOLID). At some of these stages, multiple parallel units may be available, although their configuration and availability depend on plant-specific constraints and the types of steel produced. At the same time, routing across these units may vary depending on the steel grade, due to mandatory or optional treatments and specifications.

From an operations research perspective, this structure naturally corresponds to a multi-stage production system with parallel machines at one or more stages and strong inter-stage coupling. Such systems are commonly modeled as Hybrid Flow Shops (HFS) (Lee et al., 2024). In an HFS environment, jobs must pass through several stages in a predefined order, and at least one stage includes multiple parallel machines. This abstraction effectively captures the essential topology of the EAF steelmaking procedure, where sequential processing is combined with limited parallelism.

However, the steelmaking context introduces additional features that distinguish it from classical flow shop environments. First, the absence of effective intermediate buffers between stages, mainly due to thermal constraints, creates the need for tight temporal synchronization. Second, processing times are highly grade-dependent and may vary significantly across different types of casting. Third, routing flexibility exists but is strictly determined by the steel grade: some products may require optional vacuum treatments or reheating loops, while others follow a more limited set of processing steps. Finally, auxiliary resources such as ladles and material-handling equipment introduce cross-stage coupling constraints that extend beyond individual processing stages.

These characteristics have led researchers to model the steelmaking system as Hybrid Flow shops with additional constraints. Some important modeling techniques mainly related to continuous casting can be found in the survey by (Lee et al., 2024). Although the present

work focuses on the ingot casting route, many of the underlying structural features remain the same, including sequential high-temperature processing, the presence of parallel refining units, and downstream spatial constraints associated with casting and cooling.

The abstraction from the physical process to an HFS model is therefore well established in the literature. In this framework, each heat can be interpreted as a job, each metallurgical treatment as a stage, and each furnace, refining unit, or casting device as a machine, while technological precedence relations define the order of stages. The resulting mathematical formulation makes it possible to represent the complexity of the system through assignment, sequencing, and timing decisions, while preserving the key technological constraints of the plant.

This chapter provides several key contributions to the field, with particular emphasis on the ingot casting stage of the process. We introduce the Hybrid Flow Shop for Steelmaking and Ingot Casting (HFS-SIC). The main contributions can be summarized as follows:

(i) A Hybrid Flow Shop formulation specifically tailored to the EAF-based ingot casting process.

(ii) An explicit consideration and modeling of ladles and inter-stage constraints associated with the casting phase.

(iii) A detailed representation of ingot eligibility and their allocation within solidification pits.

(iv) A systematic analysis of makespan improvements achievable through the proposed modeling approach.

(v) The introduction of strengthening constraints, including valid inequalities and symmetry-breaking constraints, along with the use of tailored upper and lower bounds.

By integrating these components into a unified formulation, the model extends the HFS literature beyond its traditional focus on continuous casting, offering a broader perspective that captures the practical challenges associated with ingot casting. In doing so, it establishes the foundation for a structurally comprehensive modeling framework, supporting high-level production planning and decision-making in industrial environments. Moreover, extensive computational experiments run on real instances prove that our approach can provide high-quality feasible solutions for this complex system.

The remainder of this chapter is structured as follows. Section 4.2 outlines the relevant literature related to the Hybrid Flow Shop problem in the context of steelmaking. Section 4.3 introduces the problem under investigation. Section 4.4 then presents its Hybrid Flow Shop (HFS)-based mathematical formulation. This section also discusses several enhancements to strengthen the model, including the derivation of upper and lower bounds, the introduction of valid inequalities, and the implementation of symmetry-breaking constraints. Finally, the two resulting formulations are summarized. Section 4.5 focuses on our experimental analysis, describing the realistic instances used, the tests performed, and the results obtained. Finally, Section 4.6 discusses conclusions and potential directions for future research.

4.2 Literature Review

The Hybrid Flow Shop (HFS) scheduling problem generalizes the classical flow shop by allowing multiple parallel machines at one or more production stages. Formally, a set of jobs

must be processed through S stages in a predefined order, and at least one stage contains more than one machine. Machines at a stage may be identical, uniform, or unrelated. For comprehensive taxonomies of HFS environments, constraints, and solution approaches, we refer to the surveys of Ruiz and Vázquez-Rodríguez (2010) and Ribas et al. (2010). HFS models naturally capture fundamental aspects of industrial production systems, including multistage precedence constraints, parallel-machine assignment, sequencing decisions at each stage, optional route flexibility (stage skipping), blocking or limited-buffer conditions, no-wait or limited-wait constraints, and additional resource coupling.

Because assignment, sequencing, and timing decisions are interdependent across stages, the HFS problem is strongly NP-hard even under simplified settings (Ruiz and Vázquez-Rodríguez, 2010). Structural variants such as no-wait HFS, blocking/limited-buffer HFS, and eligibility-restricted HFS further increase combinatorial complexity and typically require tailored modeling or solution strategies (Ribas et al., 2010). This makes the HFS abstraction particularly suitable for complex industrial systems, where feasibility is driven by synchronization and capacity interactions rather than by local stage-wise considerations.

Steelmaking scheduling, in particular its steelmaking–continuous casting (SCC) variant, exhibits precisely these structural characteristics. Each heat (charge) must pass through melting, refining, and casting stages, in which each stage contain parallel machines. Continuity requirements at casting or complex machine eligibility constraints impose strong temporal coupling. Consequently, SCC scheduling is widely interpreted as a specialized HFS enriched with industrial constraints, as systematically discussed in the recent critical review of Lee et al. (2024). Although this chapter focuses on steelmaking with ingot casting (SIC), the SCC literature provides the most mature modeling and algorithmic basis in the steel sector and clarifies which constraints are typically addressed and which remain partially treated when one moves toward less studied casting methods.

The review of Lee et al. (2024) classifies steelmaking and continuous casting scheduling approaches into: mathematical programming models, decomposition-based methods, heuristics and matheuristics, simulation-based frameworks, integrated decision systems and uncertainty-aware approaches. Across these, the HFS abstraction is consistently present. The SCC as industrial extensions of the classical HFS strongly differentiate it from the general case. In particular, most SCC models incorporate: (i) cast continuity constraints; (ii) limited buffering induced by molten steel and temperature loss; (iii) route flexibility via stage skipping; (iv) machine eligibility restrictions; and (v) auxiliary resources such as ladles that couple otherwise independent decisions (Lee et al., 2024).

Early work established SCC scheduling as an optimization problem under machine conflicts and strong coupling constraints, that however is not always defined as a strict HFS. Tang et al. (2000) propose a mathematical programming model (originally nonlinear and then linearized) to eliminate machine conflicts while balancing delivery and production continuity. Decomposition is explicitly motivated by industrial scale: Tang et al. (2002) develop a Lagrangian relaxation approach for steelmaking process scheduling, showing how relaxation can separate stages while coordinating them through multipliers. A detailed production scheduling formulation for SCC plants is also proposed by Pacciarelli and Pranzo (2004), emphasizing the modeling of stage interactions and operational constraints. Moreover, this work uses an alternative way of representing the whole system whose foundation relies on an alternative graph formulation, with which the SCC can be modeled through it.

After this initial work, the literature expanded beyond stand-alone models to system-oriented contributions in which models are defined and used in creating a plant wise decision support system or where the monolithic model incorporates more realistic constraints. Miss-

bauer et al. (2009) present a scheduling system validated on industrial data, where modeling choices are shaped by realistic settings and real-time requirements, for instance this work is one of the first that in order to better represent the realistic aspect of the SCC, models the ladle component of the process, however in this case, the fine quantities of these resources are not managed, and supposed unlimited. Fanti et al. (2016) propose an integrated architecture combining scheduling and simulation, in which the simulation and the heuristic approach are the main focus. Notably, the work of Fanti et al. (2016) is one of the few where the alternative casting method of ingot casting is taken into consideration inside the plant operations, although ingot casting is not the primary modeling focus.

At a broader level, not strictly related to steelmaking, methodological insights from process scheduling are relevant to improve the modelling due to similar coupling structures. Floudas and Lin (2004) review continuous-time versus discrete-time modeling and discuss when each paradigm is advantageous. Likewise, decomposition strategies combining mixed-integer programming and constraint programming are surveyed by Harjunkoski and Grossmann (2002), providing methodological foundations applicable to multistage industrial scheduling.

Because realistic SCC instances are large and highly constrained, a substantial literature employs specialized heuristics and hybrid approaches. However, no in depth analysis has been done over to improve the formulations to obtain a consistent evaluation over the effectiveness of heuristic solutions, since each work is focused on company-related improvements. Regarding heuristic solution we start with the work of Pan et al. (2013) which address a real-world steelmaking HFS problem using an artificial bee colony (ABC) algorithm. Pan (2016) extends this line via a co-evolutionary ABC algorithm for SCC scheduling. Hong et al. (2022) propose an iterated greedy matheuristic for SCC scheduling, explicitly handling stage skipping and waiting-time constraints. Armellini et al. (2020) develop a simulated annealing framework for steelmaking and casting scheduling, incorporating practical industrial restrictions including explicit ladle considerations, and the ingot aspect of production. Jiang (2025) developed a soft encoding-based evolutionary algorithm for managing the scheduling under electricity thresholds.

Two notable steel related contributions extend the scheduling aspect beyond the scheduling alone, connecting it to downstream processes or upstream consideration. Mattik et al. (2014) study hierarchical coordination between continuous casters and hot strip mills. Tang et al. (2014) integrate charge batching with casting width selection, illustrating that casting decisions are structurally coupled with upstream planning.

Molten steel processing, as already explained, is highly temperature-sensitive. As a result, many SCC formulations incorporate limited-wait constraints via maximum waiting times or penalties (Hong et al., 2022; Hao et al., 2015; Lee et al., 2024). Strict no-wait propagation, however, remains comparatively rare. From a theoretical standpoint, strict no-wait HFS variants fundamentally alter feasible sequencing structures as explained in a simplified model by (Wang et al., 2015). In SCC environments, strict no-wait is difficult to enforce because it interacts with route flexibility and machine eligibility. However, applying this type of constraints to the steelmaking scheduling, can be a solution strategy to obtain a fine handling of the thermal related aspect of the process, since it imposes a strict sequencing of the different stages of the process.

Stage skipping, another practical aspect acknowledged in literature, arises when certain grades bypass refining stages, or where the factory has a high variability of steel grades and quality that may or may not require additional process, as discussed in (Missbauer et al., 2009; Hong et al., 2022). Nevertheless, route flexibility when implemented is handled through mod-

elling, however a comprehensive analysis over the complexities induce by having multiple routes in the steelmaking context is never fully analyzed

Another common aspect that arises in the problem is related to the machine eligibility restrictions which can arise from caster width constraints, metallurgical compatibility and in the case of ingot casting in particular the strict coupling between solidification pits and the ingots that can be placed in it. While frequently present, eligibility is not always treated as a structural modeling, except in the work of Missbauer et al. (2009), in the other cases as in (Tang et al., 2014; Fanti et al., 2016; Rahal et al., 2020; Hong et al., 2022) are less focused and solved through a heuristic approach.

The largest part of steelmaking scheduling literature focuses on continuous casting, given that the market share of this type of process is around 97% of production as expressed by World Steel Association (2024). Ingot casting, given its lesser relevance, is comparatively underrepresented. While Fanti et al. (2016) explicitly refer to ingot casting in a plant-wide scheduling context, no major contribution formulates ingot casting as the primary structural scheduling environment within an HFS framework. In the ingot casting process, casting continuity constraints are absent, this aspect is one of the main focus in SCC, and not having these types of constraints make the ingot casting process fundamentally different from the close related one with continuous casting. The process in this case is not easier since other specific constraints, especially the one that coordinates the casting phase and the solidification phase may create similar difficulties in finding feasible solution, and by that maintains high complexity. Moreover, ladle logistics and molds management become more prominent. Consequently, the classical model that manages strict continuity, route flexibility, machine eligibility, and explicit ladle constraints, and that also defines the added complexity of the ingot mold handling, defines a largely unexplored type of problem that, to the best of our knowledge, remains unexplored. That is why an ingot-oriented HFS formulation is fundamental to reduce the gap from industry reality and the literature.

Intuitive extension of the process, in order to better manage the reality of the day-to-day production process is the uncertainty, that in steelmaking scheduling has been addressed via robust and stochastic optimization by many works as in (Ye et al., 2014; Rahal et al., 2020). Dynamic rescheduling under caster breakdown is studied in Long et al. (2017). A broader review of uncertainty-aware planning in steel production is provided by Iglesias-Escudero et al. (2019). Scheduling uncertainty, is not the only type of uncertainty that exists. Energy cost can be defined as uncertainly, in this case a more energy-aware extensions, that incorporate time-varying electricity costs and sustainability objectives can be found in the works of (Su et al., 2023, 2025). These works are increasing for many reasons, first expand objective cost in order to incorporate energy management and moreover they preserve the core HFS modelling as backbone, essential defining that the future in the field of steelmaking modelling is strictly related to the HFS formulations.

The literature confirms that steelmaking scheduling can be effectively modeled as a complex and highly constrained hybrid flow shop. However, several important gaps remain:

- Strict no-wait continuity is rarely enforced throughout the production process;
- Stage skipping is considered in some studies, but its implications for production planning are seldom analyzed;
- Steelmaking environments involving ingot casting remain largely underrepresented;
- Shared resources, such as ladles, are often simplified or assumed to be unlimited.

Accordingly, this chapter addresses these limitations by proposing a comprehensive mixed-integer linear programming formulation for steelmaking with ingot casting (SIC). The proposed model explicitly integrates operation precedence, no-wait continuity, route flexibility, machine eligibility, ingot management, and finite ladle availability within a unified scheduling framework. Furthermore, the formulation is strengthened through valid inequalities, symmetry-breaking constraints, and problem-specific bounds, resulting in a model that combines industrial realism with improved computational performance.

Table 4.1: Chronological overview of models for steelmaking scheduling.

Work	Formulation	Stage Skipping	No-Wait	Machine Eligibility	Ladle Modeling	Ingot Casting	Robust/Energy
Tang et al. (2000)	LP	◐	○	○	○	○	○
Tang et al. (2002)	MILP	○	○	○	○	○	○
Pacciarelli and Pranzo (2004)	Alternative Graph	○	●	○	◐	○	○
Missbauer et al. (2009)	MILP	●	○	●	●	○	○
Pan et al. (2013)	HFS	○	◐	○	○	○	○
Tang et al. (2014)	MILP	○	○	◐	○	○	○
Hao et al. (2015)	MILP	○	◐	○	○	○	●
Fanti et al. (2016)	HFS	○	○	◐	◐	●	○
Long et al. (2017)	HFS	○	○	○	○	○	●
Ye et al. (2014)	MILP	○	○	○	○	○	●
Rahal et al. (2020)	HFS	◐	●	◐	○	○	●
Armellini et al. (2020)	HFS	○	◐	○	●	●	○
Hong et al. (2022)	HFS	●	◐	◐	○	○	○
Su et al. (2023)	HFS	○	○	○	○	○	●
Su et al. (2025)	HFS	○	○	○	○	○	●
Jiang (2025)	HFS	○	○	○	○	○	●
This work	HFS	●	●	●	●	●	○

Notation: ● = explicitly addressed; ◐ = partially addressed or discussed; ○ = not modeled.

4.3 Problem description

Let J denote the set of jobs, where each job represents a steel casting to be processed in the plant. Let $S = \{1, 2, \dots, \bar{s}\}$ denote the ordered set of processing stages (operations), corresponding to the main technological steps of the steelmaking process (e.g., EAF, LF, VD/VOD, casting, solidification). The set S is common to all jobs and defines a global processing order. Each job $j \in J$ follows a predefined processing flow given by a subset of stages. Let $S_j \subseteq S$ denote the ordered subset of stages required by job j. Jobs may skip some stages; therefore, different processing flows coexist within the same system. In particular, in our application context, three main processing flows are considered:

(i) *Standard*, made up of :EAF, LF, VD/VOD, IC, solidification,

(ii) *Re-heating*, made up of: EAF, LF, VD/VOD, LFII, IC, solidification,

(iii) *Extended*, made up of: EAF, LF, VD/VOD, LFII, VDII, IC, solidification.

These flows have five, six, and seven stages, respectively. We indicate as i_j^{first} and i_j^{last} the first and last stage of job j, respectively, and when necessary we indicate as $i_j^{\text{first}+1}$ and $i_j^{\text{last}-1}$ the second stage and the second to last stage of job j, respectively, the possible flows and machines parallelism are described in Figure 4.2 and Figure 4.3.

Figure 4.2: Machines and resources available in the HFS model.

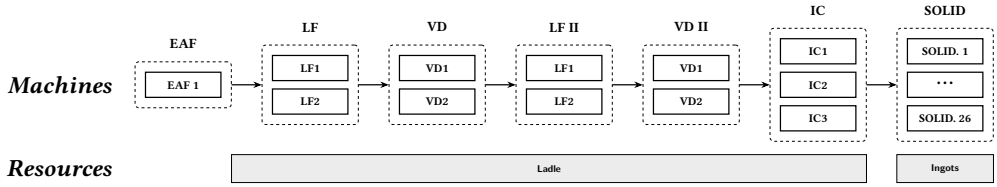
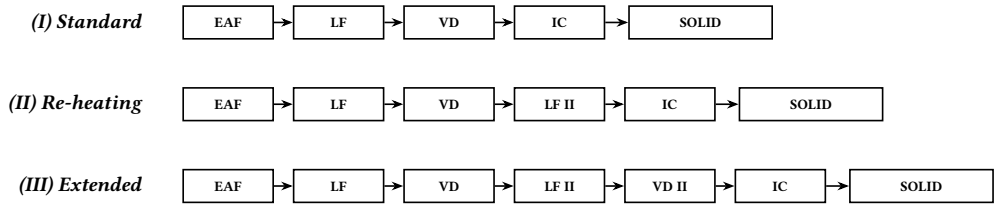


Figure 4.3: Possible Flows.



Each stage $i \in S_j$ must be processed on one machine selected from a set of eligible machines. Let M denote the set of all machines, and let $M_{ij} \subseteq M$ be the subset of machines that can process at stage i job j. Although the machines available at a given stage are assumed to be identical from a processing-time perspective, real-world considerations, such as equipment capabilities or plant-layout constraints, give rise to machine eligibility restrictions. This creates a setting in which not every machine is feasible for processing every stage of every job. Formally, for each job $j \in J$, stage $i \in S_j$, and eligible machine $m \in M_{ij} \subseteq M_i$, the corresponding processing time is denoted by p_{ijm}. Each machine can process at most one job at a time; therefore, stages assigned to the same machine must not overlap. Moreover, distinct stages may require access to the same subset of machines; in such cases, these stages are subject to the same single-machine capacity constraint and must be scheduled without overlap.

Transfer times between consecutive stages are assumed to be non-negligible. For each job $j \in J$ and for each pair of stages $i, u \in S_j$, let t_{iu} denote the corresponding transfer time. These transfer times are determined by the physical layout of the production system and are assumed to be independent of the specific job being processed.

In general, not all pairs (i, u) are feasible. More precisely, transfer times are only defined between consecutive stages, i.e., $u = i + 1$ for all $i \in S_j$.

The no-wait requirement is explicitly enforced by linking consecutive stages of the same job. The start time of stage $i + 1$ is constrained to coincide with the completion time of stage i, adjusted by the corresponding transfer time. This ensures that no intermediate buffering is allowed between stages.

In addition to processing machines, the system includes a limited set of auxiliary resources, namely ladles $k \in K$, which are required to transport and hold molten steel. Since ladles are shared resources, their allocation must be explicitly modeled. Each ladle follows a route starting from a dummy source node α, visiting a subset of jobs, and ending at a dummy sink node β. Each job must be assigned to exactly one ladle.

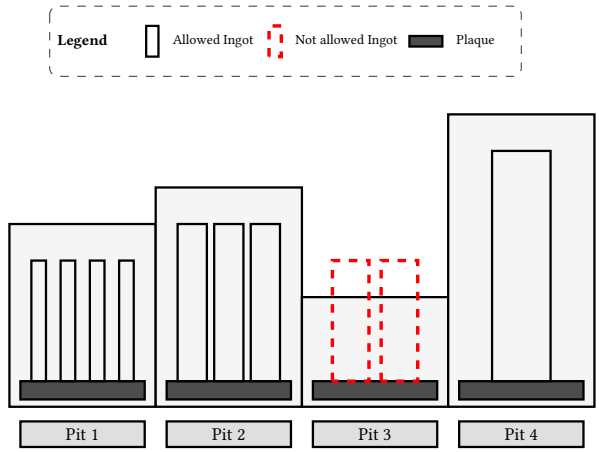
The availability of ladles must be synchronized with the production schedule. A ladle is engaged after the completion of the first stage of a job (i_j^{first}) and is released only after the casting stage, that is always (i_j^{last-1}). Furthermore, after completing a job, each ladle requires

a cleaning and transfer time τ before it can be reused. Therefore, ladles remain occupied across multiple stages and cannot be reassigned until the previous job is fully completed.

A distinctive feature of the system concerns the casting and solidification stage. Each job produces a set of ingots that must be placed into solidification pits. Ingots are defined and differentiated by geometry, capacity, and height, this creates different types of ingots, and each of them requires a supporting plaque. Each plaque can accommodate a limited number of ingots, depending on the type. For each job j, the number of required plaques depends on the number and type of ingots produced. Since pit capacity is defined in terms of plaques rather than ingots, each pit has a maximum plaque capacity. In the standard case, each job must be assigned to exactly one pit. However, a subset of jobs $\bar{J} \subset J$ requires more plaques than a single pit can accommodate. These jobs must be assigned to exactly two adjacent pits, reflecting layout and handling constraints. Consequently, the eligibility set for the solidification stage is highly constrained and depends on both capacity and adjacency conditions.

Figure 4.4 shows a generic arrangement of ingot-mold pits, highlighting their differences and some possible allocations. In this example, each pit accommodates a single plaque; however, depending on the ingot mold, the plaque can hold four molds in pit 1, three in pit 2, or one in pit 4. Pit 3 cannot accommodate the required molds because they exceed the maximum available height in that pit. The same two molds, however, can be placed in any of the other pits, although this requires waiting for the complete solidification of the molds already assigned to their respective solidification pits.

Figure 4.4: Pits and ingots molds example.



Another important aspect arises from layout-induced dependencies between machine assignments across stages. Specifically, selecting a machine at one stage may restrict the feasible set of machines at subsequent stages. For example, casting machines are served by cranes with limited coverage, so only a subset of solidification pits is reachable from each casting machine. Therefore, machine assignment decisions must be consistent across stages.

A key characteristic of the system is the handling of thermally sensitive molten steel. Due to the absence of intermediate buffers, a *no-wait* constraint applies. Once a job completes a stage, it must immediately proceed to the next after the transfer time. Any delay would result in infeasibility due to solidification risks. The problem aims at minimizing the completion time of the last stage among all jobs.

Given these characteristics, the problem can be classified using the notation of Graham et al. (1979) as:

$$HF7 \mid \textit{no-wait},\ \textit{skip},\ M_j \mid C_{\max},$$

where:

- *no-wait*: no intermediate buffering is allowed; jobs must proceed immediately between consecutive stages;
- M_j: machine eligibility constraints are enforced for each job-stage pair;
- *skip*: jobs may skip certain stages within a 7-stage flow shop structure.

To summarize, Tables 4.2 to 4.4, report the sets, indices, and parameters, respectively.

The production system under study closely reflects real industrial settings, characterized by complex flow flexibility, machine eligibility constraints, and shared resources spanning multiple stages. To better illustrate these features, a schematic representation of the plant layout and process flow is provided. The scheme highlights the sequence of production stages, the presence of parallel machines, as well as the role of shared resources such as ladles and ingots. We refer to this problem as to Hybrid Flow Shop for Steelmaking and Ingot Casting (HFS-SIC).

Table 4.2: Model sets

J	Set of Jobs
M	Set of Machines
K	Set of Ladles
S	Ordered set of stages (operations)
$\bar{J} \subset J$	Set of special jobs
$S_j \subseteq S$	Set of stages required by job j
$M_{ij} \subset M$	Set of eligible machines during stage i by job j
$M_i^{jv} = M_{ij} \cap M_{vi}$	Set of machines shared by jobs j and v for the same stage i
$M_{iu}^{jv} = M_{ij} \cap M_{vu}$	Set of machines shared by jobs j and v for different stages i and u
$V = J \cup \{\alpha\} \cup \{\beta\}$	Set of all nodes in the Ladle Graph
$V^+ = J \cup \{\alpha\}$	Set of all job nodes and the starting node
$V^- = J \cup \{\beta\}$	Set of all job nodes and the terminal node

Table 4.3: Indices

j, v	Jobs
i, u	Stages
m, n	Machines
k	Ladle
α	Starting ladle node
β	Terminal ladle node

Table 4.4: Model parameters

p_{ijm}	Processing time of stage $i \in S$ for job $j \in J$ over machine $m \in M$
t_{iu}	Transport time from stage $i \in S$ to $u \in S$
UB	Large constant
τ	Ladle cleaning and transportation time

4.3.1 Modeling Assumptions

The mathematical model presented in this chapter relies on a set of assumptions that balance industrial realism with computational tractability. Although steelmaking operations are inherently affected by uncertainty, several operational aspects are represented through deterministic approximations in order to focus on the scheduling decisions associated with steelmaking involving ingot casting. The principal assumptions adopted by the model are summarized below together with their motivation.

Deterministic processing times. The processing time of each operation is assumed to be known and constant. In practice, processing times may vary due to equipment conditions, operator interventions, or production disturbances. This variability is neglected to focus on the sequencing and resource allocation decisions that characterize the scheduling problem while maintaining a tractable optimization model.

Deterministic production requirements. All production orders, processing routes, and technological requirements are assumed to be known before the optimization process and remain unchanged throughout the scheduling horizon. This assumption reflects the offline production planning setting considered in this work, where scheduling decisions are made after customer orders have been finalized.

Deterministic resource availability. Machines, ladles, and other shared resources are assumed to remain continuously available throughout the planning horizon. Unexpected events such as equipment failures, maintenance activities, or resource breakdowns are not explicitly considered, allowing the model to focus on the interaction between resource constraints and production scheduling decisions.

Deterministic transportation and transfer operations. Transfer times between production stages, together with the movement of ladles and ingots, are assumed to be known and deterministic. Although delays may occur in industrial practice, these operations are represented by fixed parameters to preserve the computational tractability of the scheduling formulation while maintaining a realistic representation of material flow.

These assumptions are consistent with the scope of the proposed model, which focuses on developing an integrated scheduling framework for steelmaking with ingot casting under deterministic operating conditions. While uncertainty associated with processing times, equipment availability, and production disturbances is not explicitly modeled, the proposed formulation captures the principal operational constraints of the production process. Future research may extend this framework through stochastic or robust optimization approaches capable of accounting for uncertain industrial conditions. This framework can also be used as foundation, for a re-scheduling framework whose scope can be the management of the real-time condition that may arise in industrial settings.

4.4 Mathematical model

In this section, we present a mixed-integer linear programming (MILP) formulation for the HFS-SIC. The model is designed to capture the key characteristics of the real production environment described in the previous section, including routing flexibility, machine eligibility constraints, shared resources, and no-wait processing requirements.

4.4.1 The Hybrid Flow Shop Formulation

To correctly model the problem, we need to introduce different sets of decision variables that determine: (i) the assignment of machines at each stage to each job, (ii) the necessary sequencing of jobs on shared machines, to exclude overlapping usage, (iii) the allocation and routing of ladles across jobs, (iv) the enforcement of inter-machine eligibility constraints, and (v) the feasibility of solidification pits, based on the necessary ingot defined by each job.

Binary variables x_{ijm} assign each stage i of job j to exactly one eligible machine $m \in M_{ij}$. The sequencing of stages on shared machines is enforced through binary variables y^{v}_{ijm} and z^{uv}_{ijm}, which define the relative processing order of jobs competing for the same machine, either within the same stage or across different stages executable on that machine. In particular, for each pair of distinct jobs $j, v \in J$, stage $i \in S_j \cap S_v$, and machine $m \in M_{ij} \cap M_{iv}$, the variable y^{v}_{ijm} takes value 1 if job j is processed before job v at stage i on machine m, and 0 otherwise. These variables define the processing order between jobs assigned to the same machine, ensuring that their execution intervals do not overlap. When machines can process jobs belonging to different stages, cross-stage sequencing variables are required: for each pair of jobs $j, v \in J$, stages $i \in S_j$ and $u \in S_v$, and machine $m \in M_{ij} \cap M_{vu}$, we introduce binary variables z^{uv}_{ijm} that takes value 1 if job j at stage i is processed before job v at stage u on machine m, and 0 otherwise. These variables ensure that jobs belonging to different stages but assigned to the same machine do not overlap in time. For each job $j \in J$ and stage $i \in S_j$ and eligible machine $m \in M_{ij}$, continuous variables s_{ijm} and c_{ijm} denote the start and completion times, respectively. In order to manage additional resources as in the ladles, binary variables h^{k}_{jv} indicate whether job j is processed immediately before job v by ladle k. Finally, the makespan variable $C_{\max}$ represents the completion time of the last stage among all jobs and is minimized in the objective function. A summary of all decision variables is reported in Table 4.5.

Table 4.5: Decision Variables

x_{ijm}	Binary variable taking value 1 if stage $i \in S$ of job $j \in J$ is assigned to machine $m \in M$; 0 otherwise.
y^{v}_{ijm}	Binary variable taking value 1 if, for stage $i \in S$, job $j \in J$ is processed before job $v \in J$ on machine $m \in M$; 0 otherwise.
z^{uv}_{ijm}	Binary variable taking value 1 if stage $i \in S$ of job $j \in J$ is processed before stage $u \in S$ of job $v \in J$ on machine $m \in M$; 0 otherwise.
h^{k}_{jv}	Binary variable taking value 1 if job $j \in J$ is processed before job $v \in J$ by ladle $k \in K$; 0 otherwise.
l_k	Binary variable taking value 1 if ladle $k \in K$ is used; 0 otherwise.
a_{jm}	Binary variable used to enforce adjacency in pit assignment for special jobs $j \in \bar{J}$
s_{ijm}	Continuous variable representing the start time at stage $i \in S$ of job $j \in J$ on machine $m \in M$.
c_{ijm}	Continuous variable representing the completion time at stage $i \in S$ of job $j \in J$ on machine $m \in M$.
s_{jk}	Continuous variable representing the start time of job $j \in J$ on ladle $k \in K$.
c_{jk}	Continuous variable representing the completion time of job $j \in J$ on ladle $k \in K$.
$C_{\max}$	Continuous variable representing the makespan of the schedule.
u_{jk}	Integer ordering variable for job $j \in J$ on ladle $k \in K$, used in the MTZ constraints.

The proposed MILP formulation model for the HFS-SIC, can be defined as follows:

$$\min \quad C_{\max} \tag{4.1}$$

$$\text{s.t. } C_{\max} \geq c_{ijm} \qquad j \in J, i = i_j^{last} \tag{4.2}$$

$$\sum_{m \in M_{ij}} x_{ijm} = 1 \qquad j \in J \setminus \bar{J},\ i \in S_j \tag{4.3}$$

$$s_{ijm} \leq x_{ijm} UB \qquad j \in J, i \in S_j, m \in M_{ij} \tag{4.4}$$

$$c_{ijm} \leq x_{ijm} UB \qquad j \in J, i \in S_j, m \in M_{ij} \tag{4.5}$$

$$c_{ijm} = s_{ijm} + p_{ijm} \cdot x_{ijm} \qquad j \in J, i \in S_j, m \in M_{ij} \tag{4.6}$$

$$s_{ivm} \geq c_{ijm} - UB\,(1 - y^{v}_{ijm}) - UB\,(2 - x_{ijm} - x_{ivm}) \qquad j, v \in J, i \in S_j \cap S_v, m \in M_i^{jv} \tag{4.7}$$

$$s_{ijm} \geq c_{ivm} - UB \cdot y^{v}_{ijm} - UB\,(2 - x_{ijm} - x_{ivm}) \qquad j, v \in J, i \in S_j \cap S_v, m \in M_i^{jv} \tag{4.8}$$

$$s_{uvm} \geq c_{ijm} - UB\,(1 - z^{uv}_{ijm}) - UB\,(2 - x_{ijm} - x_{ivm}) \qquad j, v \in J, i \in S_j u \in S_v\ i \neq u, m \in M_{iu}^{jv} \tag{4.9}$$

$$s_{ijm} \geq c_{uvm} - UB \cdot z^{uv}_{ijm} - UB\,(2 - x_{ijm} - x_{ivm}) \qquad j, v \in J, i \in S_j u \in S_v\ i \neq u, m \in M_{iu}^{jv} \tag{4.10}$$

$$s_{i+1,j,n} \geq c_{ijm} + t_{i,i+1} - UB\,(2 - x_{ijm} - x_{i+1,j,n}) \qquad j \in J, i \in S_j \setminus \{i_j^{last}\}, m \in M_{ij}, n \in M_{i+1,j} \tag{4.11}$$

$$s_{i+1,j,n} \leq c_{ijm} + t_{i,i+1} + UB\,(2 - x_{ijm} - x_{i+1,j,n}) \qquad j \in J, i \in S_j \setminus \{i_j^{last}\}, m \in M_{ij}, n \in M_{i+1,j} \tag{4.12}$$

$$\sum_{j \in V^-} h^k_{\alpha j} = 1 \qquad k \in K \tag{4.13}$$

$$\sum_{j \in V^+} h^k_{j\beta} = 1 \qquad k \in K \tag{4.14}$$

$$\sum_{k \in K} \sum_{v \in J \setminus \{j\}} h^k_{vj} = 1 \qquad j \in J \tag{4.15}$$

$$\sum_{k \in K} \sum_{v \in J \setminus \{j\}} h^k_{jv} = 1 \qquad j \in J \tag{4.16}$$

$$\sum_{v \in J \setminus \{j\}} h^k_{vj} - \sum_{v \in J \setminus \{j\}} h^k_{jv} = 0 \qquad j \in J,\ k \in K \tag{4.17}$$

$$u_{vk} - u_{jk} + n \cdot h^k_{vj} \leq n - 1 \qquad j, v \in J,\ v \neq j,\ k \in K \tag{4.18}$$

$$s_{vk} \geq c_{jk} - (1 - h^k_{jv}) UB \qquad k \in K \tag{4.19}$$

$$s_{jk} \leq \sum_{v \in V \setminus \{j\}} (h^k_{jv} + h^k_{vj}) UB \qquad k \in K, j \in J \tag{4.20}$$

$$c_{jk} \leq \sum_{v \in V \setminus \{j\}} (h^k_{jv} + h^k_{vj}) UB \qquad k \in K, j \in J \tag{4.21}$$

$$\sum_{k \in K} s_{jk} = \sum_{m \in M_{ji}} c_{jim} \qquad j \in J, i = i_j^{first} \tag{4.22}$$

$$\sum_{k \in K} c_{jk} = \sum_{m \in M_{ji}} c_{jim} + \tau \qquad j \in J, i = i_j^{last-1} \tag{4.23}$$

$$x_{ijm} \in \{0, 1\} \qquad j \in J, i \in S_j, m \in M_{ij} \tag{4.24}$$

$$y^{v}_{ijm} \in \{0, 1\} \qquad j, v \in J, i \in S_j \cap S_v, m \in M_{ij} \tag{4.25}$$

$$z^{uv}_{ijm} \in \{0, 1\} \qquad j, v \in J, i \in S_j u \in S_v u \neq i, m \in M_{ij} \tag{4.26}$$

$$h^k_{jv} \in \{0, 1\} \qquad j, v \in J, k \in K \tag{4.27}$$

$$s_{ijm} \geq 0, c_{ijm} \geq 0 \qquad j \in J, i \in S_j, m \in M_{ij} \tag{4.28}$$

$$s_{jk} \geq 0, c_{jk} \geq 0 \qquad j \in J, K \in K \tag{4.29}$$

$$C_{\max} \geq 0 \tag{4.30}$$

$$u_{jk} \in \mathbb{Z}_+ \qquad j \in J, K \in K \tag{4.31}$$

The objective function (4.1) minimizes the makespan $C_{\max}$, defined as the maximum completion time among all jobs. This ensures that the overall production schedule is as short as possible. Constraint (4.2) defines the makespan as an upper bound on the completion time of the last stage of each job $j \in J$. Constraint (4.3) imposes that each stage i of job j is assigned to exactly one eligible machine in the set M_{ij}, thereby ensuring a unique processing route for every stage. Constraints (4.4) and (4.5) link the timing variables to the assignment decisions. In particular, they force the start time s_{ijm} and completion time c_{ijm} to be zero whenever stage i for job j is not assigned to machine m (i.e., when $x_{ijm} = 0$), using a suffi-

ciently large constant UB. Constraint (4.6) defines the completion time of each stage as the sum of its start time and processing time. The formulation $c_{ijm} = s_{ijm} + p_{ijm}x_{ijm}$ ensures that the processing time contributes only if the corresponding assignment is active. Constraints (4.7) and (4.8) enforce disjunctive sequencing for stages of the same type processed on the same machine. For any pair of jobs j and v, these constraints ensure that either job j precedes job v or vice versa, thereby preventing overlapping processing on the same resource. The binary variables y^{v}_{ijm} determine the relative order. Constraints (4.9) and (4.10) extend the non-overlapping conditions to stages of different types competing for the same machine. The binary variables z^{uv}_{ijm} determine the sequencing between stages i of job j and u of job v, ensuring that no two stages assigned to the same machine overlap in time. Constraints (4.11) and (4.12) enforce the no-wait condition between consecutive stages of the same job. Together, they impose that the start time of stage $i + 1$ is equal to the completion time of stage i plus the transfer time $t_{i,i+1}$, provided that the corresponding machine assignments are selected. The ladle assignment and routing are modeled as a network flow problem over the set of jobs. Constraint (4.13) ensures that each ladle $k \in K$ departs exactly once from the dummy source node α, while constraint (4.14) enforces that each ladle arrives exactly once at the dummy sink node β. These constraints define the start and end of each ladle route. Constraints (4.15) and (4.16) impose that each job node is visited exactly once across all ladles, ensuring a complete assignment of jobs to ladles. Constraint (4.17) enforces flow conservation for each job and ladle, guaranteeing route continuity. Constraint (4.18) eliminates subtours in the ladle routing by means of a Miller–Tucker–Zemlin (MTZ) formulation. The auxiliary variables u_{jk} impose an ordering over the visited jobs, preventing the formation of disconnected cycles. Constraint (4.19) enforces temporal consistency between consecutive jobs assigned to the same ladle. If job v follows job j on ladle k, then the start time of v must be greater than or equal to the completion time of j. Constraints (4.20) and (4.21) deactivate the timing variables associated with ladle-job pairs when the job is not assigned to the corresponding ladle. Constraint (4.22) synchronizes the beginning of ladle processing with the completion of the first stage of each job, ensuring consistency between production and handling stages. Constraint (4.23) enforces consistency at the end of the process by linking the completion of ladle stages to the completion of the second to last job stage, it also includes the the cleaning and final transportation time τ. Finally, the remaining constraints define the domains of the decision variables: binary variables for assignment, sequencing, and routing decisions, non-negativity for timing variables, and integrality for the auxiliary variables.

4.4.2 Ingot Management

The management of ingots represents a key feature that distinguishes the proposed model from the majority of contributions in the steelmaking literature, which predominantly focus on the continuous casting process. In contrast, the ingot casting process requires an explicit modeling of the assignment of ingots to solidification pits, where they remain until the molten steel has sufficiently cooled to allow further processing or complete solidification, as discussed in Section 4.1.

Each job is associated with a set of ingots that may belong to different families, defined by characteristics such as shape, capacity, and height. Each ingot can be assigned only to a subset of the available solidification pits. Moreover, each ingot requires a supporting plaque, and each plaque can accommodate a limited number of ingots depending on their family. For instance, a plaque may host up to 8 ingots of a given family, while only 1 ingot of another family. As a consequence, pit occupancy is determined by the number of required plaques

rather than by the number of ingots. For each job, the number of required plaques is computed as the ratio between the total number of ingots and the maximum number of ingots that can be placed on a single plaque for each family. This determines the effective capacity requirement for the solidification stage.

Each pit has a limited plaque capacity, and under standard conditions, each job must be assigned to exactly one pit. However, for a subset of jobs $\bar{J} \subset J$, the required number of plaques exceeds the capacity of a single pit. These jobs must therefore be assigned to two pits simultaneously. Due to physical layout and operational constraints, the selected pits must be adjacent to ensure feasible casting stages. Based on these considerations, a feasible set of solidification assignments can be determined for each job. This information can be incorporated through a preprocessing phase that restricts the set of eligible machines (pits). However, this restriction substantially increases the difficulty of the problem, as it reduces the number of feasible solutions.

In real plant operations, the possibility of assigning a job to multiple pits represents an important extension that is not captured in the basic formulation. To model this case, the subset of jobs $\bar{J}$ is explicitly introduced. For these jobs, solidification cannot take place in arbitrary pits, but only in pairs of adjacent pits. This requirement is enforced through additional constraints, which ensure both multiple assignment and adjacency conditions, as defined below:

$$\sum_{m \in M_{ij}} x_{ijm} = 1 \qquad j \in \bar{J},\ i \in S_j \setminus \{i_j^{last}\}, \tag{4.32}$$

$$\sum_{m \in M_{ij}} x_{ijm} = 2 \qquad j \in \bar{J},\ i = i_j^{last} \tag{4.33}$$

$$\sum_{m=1}^{|M_{ij}|-1} a_{jm} = 1 \qquad j \in \bar{J},\ i = i_j^{last} \tag{4.34}$$

$$x_{ij1} = a_{j1} \qquad j \in \bar{J},\ i = i_j^{last} \tag{4.35}$$

$$x_{ijm} = a_{j,m-1} + a_{jm} \qquad j \in \bar{J},\ i = i_j^{last},\ m = 2, \dots, |M_{ij}| - 1 \tag{4.36}$$

$$x_{ij,|M_{ij}|} = a_{j,|M_{ij}|-1} \qquad j \in \bar{J},\ i = i_j^{last} \tag{4.37}$$

$$a_{jm} \in \{0, 1\} \qquad j \in \bar{J},\ m = 1, \dots, |M_{ij}| - 1,\ i = i_j^{last} \tag{4.38}$$

Constraint (4.32) ensures that, for special jobs, all stages other than the last one (solidification) stage are assigned to exactly one eligible machine. Constraint (4.33) imposes that special jobs must be assigned to exactly two machines at the solidification stage, representing the simultaneous use of two pits. Constraint (4.34) enforces the selection of exactly one adjacency position within the ordered set of eligible pits. Constraints (4.35)–(4.37) define the special order set of the second type (SOS2) structure linking variables a_{jm} and x_{ijm}, so that exactly two consecutive pits are selected. Finally, constraint (4.38) specifies the binary domain of the adjacency variables.

The above constraints exploit an SOS2-like structure to enforce the selection of exactly two adjacent pits for special jobs. This formulation assumes that the eligible pits are indexed according to their physical layout, so that consecutive indices correspond to adjacent pits.

4.4.3 Inter-machine eligibility

An additional set of constraints is required to capture inter-machine eligibility dependencies across different stages. In particular, the selection of a machine at a given stage may restrict the set of feasible machines at subsequent stages. This situation arises when the physical layout of the plant imposes accessibility limitations between stages. As a result, not all machines that are individually eligible for a job can be jointly selected across its processing flow.

A relevant example occurs between the casting and solidification stages. Casting machines are served by cranes with limited spatial coverage, and therefore each casting machine can only reach a subset of the available solidification pits. Consequently, assigning a job to a specific casting machine restricts the set of feasible pits in the subsequent solidification stage. Similarly, selecting a given pit may implicitly restrict the set of compatible casting machines. To ensure feasibility, these inter-stage dependencies must be explicitly enforced in the model through additional constraints:

$$x_{ijm} \leq x_{ujm} \qquad j \in J,\ u \prec i \in S_j,\ m \in M_{ij} \tag{4.39}$$

Constraint (4.39) enforces consistency between machine assignments across dependent stages by ensuring that the selection of a machine at a later stage is compatible with the machine chosen at a preceding stage.

4.4.4 Valid Inequalities

To strengthen the formulation, we introduce earliest-start valid inequalities. For each job j and stage i, we define a parameter ES_{ij} representing a lower bound on the start time of stage i of job j, derived only from the job flow and ignoring machine conflicts.

The value ES_{ij} is computed sequentially along the ordered stages of job j. Let u denote the stage immediately preceding i in the flow of job j. Then,

$$ES_{i_j^{\text{first}}} = 0, \tag{4.40}$$

$$ES_{ij} = ES_{uj} + p_{uj} + t_{ui}, \qquad i \neq i^{\text{first}}. \tag{4.41}$$

where p_{uj} denotes the processing time of stage u of job j, and t_{ui} is the transfer time between consecutive stages. Thus, ES_{ij} represents the earliest possible start time of stage i, assuming that all preceding stages are executed as soon as possible. Using this parameter, we add the valid inequalities

$$s_{ijm} \geq ES_{ij} \cdot x_{ijm}, \qquad j \in J,\ i \in S_j,\ m \in M_{ij}. \tag{4.42}$$

A similar strengthening can be derived for completion times. Since the completion time is defined as the sum of the start time and the processing time, we obtain the inequalities

$$c_{ijm} \geq \left(ES_{ij} + p_{ijm}\right) x_{ijm}, \qquad j \in J,\ i \in S_j,\ m \in M_{ij}. \tag{4.43}$$

These inequalities strengthen the formulation by tightening the bounds of the continuous variables. By embedding precedence-based information directly into the model, they improve the quality of the linear relaxation.

In addition, stronger inequalities can be introduced for the sequencing variables involving pairs of jobs:

$$y^v_{ijm} \leq x_{ijm}, \qquad j, v \in J,\ i \in S_j \cap S_v,\ m \in M_{ij} \tag{4.44}$$

$$y^v_{ijm} \leq x_{vim}, \qquad j, v \in J,\ i \in S_j \cap S_v,\ m \in M_{ij}. \tag{4.45}$$

These constraints ensure that a sequencing decision between two jobs on a machine is activated only if both jobs are assigned to that machine, thereby eliminating infeasible configurations.

A similar strengthening can be applied to the variables z:

$$z^{uv}_{ijm} \leq x_{ijm}, \qquad j, v \in J,\ i \in S_j, u \in S_v, u \neq i\ , m \in M^{jv}_{iu}, \tag{4.46}$$

$$z^{uv}_{ijm} \leq x_{ium}, \qquad j, v \in J,\ i \in S_j, u \in S_v, u \neq i\ , m \in M^{jv}_{iu}, \tag{4.47}$$

$$z^{uv}_{ijm} \leq x_{vim}, \qquad j, v \in J,\ i \in S_j, u \in S_v, u \neq i\ , m \in M^{jv}_{iu}, \tag{4.48}$$

$$z^{uv}_{ijm} \leq x_{vum}, \qquad j, v \in J,\ i \in S_j, u \in S_v, u \neq i\ , m \in M^{jv}_{iu}. \tag{4.49}$$

These inequalities ensure that the variables z can take positive values only when all corresponding assignments of both jobs to the relevant machines and stages are satisfied. Overall, these valid inequalities tighten the linear relaxation, reduce the feasible region, and are expected to improve the computational performance of the model.

4.4.5 Symmetry Breaking

To mitigate symmetry in the model, we introduce a new binary variable l_k associated with each ladle $k \in K$, which activates the usage of ladle k. These variables enable explicit control over ladle utilization and allow the introduction of ordering constraints. This introduction of new variables changes some already defined constraints, as in Equations (4.13) and (4.16), these can now be defined in the following way:

$$\sum_{j \in V^-} h^k_{\alpha j} = l_k \qquad k \in K \tag{4.50}$$

$$\sum_{j \in V^+} h^k_{j\beta} = l_k \qquad k \in K \tag{4.51}$$

$$\sum_{k \in K} \sum_{v \in J \setminus \{j\}} h^k_{vj} = l_k \qquad j \in J \tag{4.52}$$

$$\sum_{k \in K} \sum_{v \in J \setminus \{j\}} h^k_{jv} = l_k \qquad j \in J \tag{4.53}$$

$$h^k_{vj} \leq l_k \qquad v, j \in J, k \in K \tag{4.54}$$

In particular, symmetry among identical ladles is reduced by enforcing the following monotonicity condition:

$$l_k \geq l_{k+1}, \qquad k = 1, \ldots, |K| - 1. \tag{4.55}$$

This constraint ensures that ladles are activated sequentially according to their indices, thereby eliminating equivalent solutions arising from permutations of ladle labels.

Additionally, symmetry is further reduced by imposing an ordering on the number of jobs assigned to each ladle. Specifically, we require that the number of arcs associated with each ladle is non-increasing with respect to its index:

$$\sum_{j\in J}\sum_{v\in J\setminus\{j\}} h_{jv}^{k} \geq \sum_{j\in J}\sum_{v\in J\setminus\{j\}} h_{jv}^{k+1}, \qquad k = 1,\dots,|K|-1. \tag{4.56}$$

Together, these constraints enforce some ordering of ladle activation and utilization, effectively removing part of the symmetric solutions from the feasible region. Overall, the proposed symmetry-breaking scheme preserves at least one representative solution while significantly reducing redundant exploration of the solution space, thereby enhancing computational efficiency.

4.4.6 Lower bounds

In order to strengthen the formulation, some problem-specific lower bounds on the makespan are introduced.

A first lower bound is obtained by considering, for each job, the total amount of processing and transfer time required to complete all its stages. In the considered hybrid flow shop, machines at each stage are identical in terms of processing capability, although eligibility restrictions may limit the set of machines that can process a given job. Hence, the processing time of job j at stage i does not depend on the specific machine. So to ease the notation, we define p_{ij} as $p_{ij} = p_{ijm}$ for all $m \in M_{ij}$. Therefore, under ideal conditions with no waiting time between stages, the completion time of each job is at least equal to the sum of its processing plus the necessary transfer times. Since the makespan is defined as the maximum completion time over all jobs, this leads to the following lower bound, that corresponds to the longest job.

$$LB_1 = \max_{j\in J}\left\{\sum_{i\in S_j} p_{ij} + \sum_{i=i_j^{first}}^{i_j^{last}} t_{i,i+1}\right\}. \tag{4.57}$$

A second lower bound uses the presence of a bottleneck stage in the production system. In the considered setting, this role is played by the Electric Arc Furnace (EAF), which constitutes the capacity-limiting resource. This stage is modeled as the first stage of the production process and is equipped with a single machine.

As a consequence, every job $j \in J$ must necessarily be processed on this unique machine before proceeding to subsequent stages. Since no parallel processing is possible at this stage, all jobs must be handled sequentially. Therefore, the makespan cannot be smaller than the cumulative processing time of all jobs at the bottleneck stage.

Hence, the following lower bound is obtained:

$$LB_2 = \sum_{j\in J} p_{i_j^{first}j}. \tag{4.58}$$

A third lower bound can be derived by considering the completion of the bottleneck stage. Once all jobs have been processed on the first-stage bottleneck, the job that completes last on this stage must still undergo all its subsequent processing stages before the schedule can be completed. Since the identity of this job is not known a priori, a valid lower bound can be obtained by augmenting LB_2 with the minimum possible remaining workload over

all jobs. More precisely, this residual workload consists of the sum of processing times of the remaining stages and the associated inter-stage transfer times. By considering the minimum such quantity across all jobs, we obtain the following lower bound:

$$LB_3 = LB_2 + \min_{j \in J} \left\{ \sum_{i \in S_j \setminus \{i_j^{\text{first}}\}} p_{ij} \; + \sum_{i=i_j^{first+1}}^{i_j^{last}} t_{i,i+1} \right\} \tag{4.59}$$

The overall lower bound is obtained as the maximum among the previously defined bounds. This aggregation ensures that all limiting factors are simultaneously taken into account. Accordingly, the makespan must satisfy the following constraint:

$$C_{\max} \geq \max\{LB_1, LB_2, LB_3\}. \tag{4.60}$$

4.4.7 Upper bounds

Upper bounds play a fundamental role in the proposed model for several reasons. In particular, they are essential for defining a finite time horizon and for appropriately calibrating the big-M coefficients appearing in the formulation, whose magnitude strongly impacts the numerical performance of the model.

In this work, upper bounds are obtained through a combination of approaches. On one hand, some bounds are derived from relaxations of the original problem, providing analytical estimates of the makespan. On the other hand, tighter bounds are computed via constructive heuristics that explicitly generate feasible schedules.

A first upper bound is obtained by considering the case in which all parallelism is removed. In this fully sequential setting, jobs are processed one at a time without overlap, i.e., a job can start only after the completion of all stages of the preceding job. Under this assumption, the Hybrid Flow Shop reduces to a classical flow shop with a single machine per stage. This construction eliminates synchronization decisions across parallel resources. Consequently, no non-overlapping constraints are required, and feasibility is guaranteed by design.

In this case, the makespan can be readily computed as the cumulative sum of the processing times of all jobs across all stages, augmented by the corresponding transfer times between consecutive stages.

$$UB_1 = \sum_{j \in J} \left(\sum_{i \in S_j} p_{ij} + \sum_{(i,i+1)} t_{i,i+1} \right). \tag{4.61}$$

A tighter upper bound can be obtained through a constructive scheduling heuristic that incrementally builds a feasible solution. Jobs are first sorted in nondecreasing order of their total workload, defined as the sum of processing and transfer times. Subsequently, each job is inserted into the schedule by assigning a starting time to its first stage and verifying the existence of a feasible machine allocation for all subsequent stages.

Given a fixed starting time for the first stage, the start times of the remaining stages are uniquely determined by the no-wait constraint, which enforces a strict temporal sequencing between consecutive stages. Machine assignment is then performed by selecting, at each stage, the first available eligible machine that satisfies the job-specific constraints.

Due to the machine eligibility restriction that occurs between the casting and solidification stages, the machine assignment for these stages is carried out in reverse order with

respect to the processing sequence. Specifically, a feasible solidification configuration is first identified, and subsequently, a compatible casting machine is selected that is available at the required time.

If no feasible assignment exists for the current starting time of the first stage, the starting time is incrementally postponed, and the feasibility check is repeated. Once a feasible allocation is found, machine occupation times are updated, and the job is permanently inserted into the schedule. Repeating this procedure for all jobs yields a feasible schedule, whose makespan defines the upper bound. A high-level description of the constructive heuristic is provided in Algorithm 4.1.

The second upper bound is therefore defined as:

$$UB_2 = C_{\max}^{\text{heuristic}}. \tag{4.62}$$

4.4.8 Basic and Advanced models

To better assess the impact of the proposed strengthening mechanisms, two alternative MILP formulations are considered. The first one, denoted as the *basic model*, consists only of the core Hybrid Flow Shop formulation and the constraints required to ensure feasibility of the production process. The second one, denoted as the *advanced model*, extends the basic model by including all the proposed improvement components, namely the lower-bound constraint on the makespan, the tightened upper bounds used to reduce the planning horizon and calibrate the big-M parameters, the valid inequalities, and the symmetry-breaking constraints for ladle assignment and utilization. Hence, the advanced model preserves the structure of the basic formulation while providing a stronger representation that is expected to improve computational performance. The list of constraints of the two models is defined in Table 4.6. A component-wise analysis of the strengthening mechanisms is presented in Section 4.5.2 using a representative subset of instances, providing further justification for the two model configurations adopted in the computational study.

Table 4.6: Comparison between the basic and the advanced MILP formulations

Component	**Basic model**	**Advanced model**
Objective	(4.1)	(4.1)
Core HFS constraints	(4.2)–(4.23)	(4.2)–(4.23)
Ingot management	(4.32)–(4.38)	(4.32)–(4.38)
Inter-machine eligibility	(4.39)	(4.39)
Ladles modeling	(4.13)–(4.16) (4.17)–(4.23)	(4.50)–(4.54) (4.17)–(4.23)
Lower bound		(4.60)
Upper bound	UB_1	UB_2
Valid inequalities		(4.42)–(4.49)
Symmetry breaking		(4.55)–(4.56)

Algorithm 4.1 Constructive heuristic for the computation of UB_2

```
Sort jobs in nondecreasing order of total time
Initialize machine availability times
for each job j in the ordered list do
    t ← 0
    scheduled ← false
    while scheduled = false do
        Compute the start times of all stages of job j from t
        feasible ← true
        for each stage i ∈ S_j \ {casting, solidification} do
            Select the first available eligible machine
            if no eligible machine is available then
                feasible ← false
                break
            end if
        end for
        if feasible then
            Select a feasible solidification configuration
            if no feasible solidification configuration is available then
                feasible ← false
            end if
        end if
        if feasible then
            Select an available casting machine compatible with solidification
            if no compatible casting machine is available then
                feasible ← false
            end if
        end if
        if feasible then
            Fix machine assignments for job j
            Update C_max^heuristic from the schedule
            Update machine availability times
            scheduled ← true
        else
            t ← t + 1
        end if
    end while
end for
return C_max^heuristic
```

4.5 Experimental Results

This section presents the computational experiments conducted in this study. First, we describe the construction of the test instances, which are derived from real-world company data to ensure their practical relevance. We then conduct a component-wise analysis on a representative subset of instances to evaluate the individual and cumulative effects of the proposed strengthening mechanisms, including the makespan lower bound, tightened upper

bounds, valid inequalities, and symmetry-breaking constraints. This preliminary analysis provides additional justification for selecting the basic and advanced model configurations used in the subsequent experiments.

Next, we examine the effectiveness of the proposed lower bounds across the complete set of test instances, followed by an assessment of the tightened upper bounds and their ability to reduce the planning horizon and improve the calibration of the big-M parameters. Finally, we compare the basic and advanced formulations over the complete instance set, evaluating their performance in terms of solution quality and computational efficiency. Together, these experiments provide a comprehensive assessment of the contribution of the proposed strengthening mechanisms and the overall performance of the advanced formulation.

The mathematical model was implemented in Java 21 and solved using Gurobi 10.0.1. All experiments were conducted on a system running Ubuntu 22.04 and equipped with an AMD Ryzen 9 3950X 32-core processor and 24 GB of RAM. All available threads were used.

4.5.1 Instances generation

The computational study is based on benchmark instances generated from real-world data provided by an industrial partner over a 30-day period, as detailed in Section 3.7.1. To ensure a consistent evaluation framework, the same dataset introduced is adopted for assessing the proposed optimization model. Specifically, the used benchmark is comprises by the same five sets of instances, namely $D1$, $D2$, $D3$, $D4$, and $D6$, representing the same planning horizons of one, two, three, four, and six consecutive days, respectively.

From an experimental perspective, increasing the planning horizon naturally results in larger scheduling instances, as more production jobs are included in the optimization problem. This provides a systematic framework for evaluating both the scalability and the robustness of the proposed optimization model. In particular, the larger instance sets ($D4$ and $D6$) are used to assess the computational performance of the model under increasingly demanding scenarios, providing insights into its effectiveness and limitations when solving large-scale industrial scheduling problems. Overall, this experimental design enables the analysis of the model's behavior as the problem size grows while preserving the characteristics of real production data.

Regarding the plant configuration, some parameters are fixed and common to all instances considered. In particular, the number of machines available at each stage is defined as follows. The production system under consideration consists of a multi-stage process with both mandatory and optional stages. The first stage is the Electric Arc Furnace (EAF), which comprises a single machine. The second stage is the Ladle Furnace (LF), equipped with two identical parallel machines, followed by the third stage, Vacuum Degassing (VD), which also consists of two identical parallel machines. In addition to these core stages, two optional stages may be required depending on the specific job characteristics. The fourth stage corresponds to a second pass through the Ladle Furnace (LFII), while the fifth stage represents a second pass through the Vacuum Degassing units (VDII). Both optional stages utilize the same machines as those employed in the second and third stages, respectively. The sixth stage is the ingot casting stage (IC), which is composed of three identical parallel machines. At this stage, certain jobs may require processing on a restricted subset of machines in order to meet specific quality requirements, as not all machines provide the same level of product quality. Finally, the seventh stage is the solidification stage (SOLID), consisting of 26 available machines (pits). Machine eligibility at this stage is highly constrained: not all jobs can be assigned to all pits, as the physical and dimensional characteristics of the ingots limit the

feasible allocation of jobs to solidification units.

The pits are distributed across three macro-areas within the facility and are used during the ingot solidification stage. Their utilization is subject to accessibility constraints depending on the casting machine employed. In particular:

- IC1 can access 19 out of the 26 pits,
- IC2 can access 5 out of the 26 pits,
- IC3 can access 7 out of the 26 pits.

This induces a machine–pit eligibility structure that restricts the feasible allocation of casting operations to solidification resources. A total of 51 different ingot mold types are available. Each mold type is characterized by a specific number of compatible pits, ranging from a minimum of 2 to a maximum of 15. On average, and disregarding machine–pit eligibility constraints, each mold type is compatible with approximately 8 pits. Regarding the flow of jobs through the plant, three alternative processing routes are considered:

- *Standard*, accounting for approximately 96% of the jobs;
- *Re-heating*:, accounting for approximately 3% of the jobs;
- *Extended*:, accounting for approximately 1% of the jobs.

These proportions reflect the empirical distribution observed in plant operations. Table 4.7 reports summary statistics of the processing times at each stage of the production process. For each stage, the minimum, maximum, and average processing times (in minutes) are provided.

Table 4.7: Processing times statistics at each production stage (in minutes).

p_{ijm}	Min	Max	*Avg.*
EAF	57.0	401.0	85.1
LF	30.0	206.0	84.8
VD	30.0	128.0	71.4
IC	34.0	498.0	59.2
SOLID.	180.0	4930.0	676.0

The number of available ladles is fixed to 8 for all instances. Each ladle requires a cleaning and final transportation time of τ equal to 15 minutes.

Transportation times between consecutive stages are also considered and range from 0 to 15 minutes, depending on the specific stages involved.

The benchmark instances considered in this study are reconstructed from the historical production data collected from the industrial partner. The production orders, processing routes, processing times, and resource eligibility constraints are derived from the industrial dataset. However, some operational parameters are fixed across all instances or defined according to company estimates to ensure a controlled and consistent computational analysis. Consequently, the benchmark instances should be regarded as representative industrial case studies.

4.5.2 Analysis of the strengthening mechanisms

To assess the contribution of each strengthening mechanism without exhaustively evaluating every model configuration over the complete benchmark, a representative sample of instances is used. Specifically, three instances are randomly selected from each of the five instance sets $D1$, $D2$, $D3$, $D4$, and $D6$, yielding a sample of 15 instances that spans all planning horizons considered in the computational study. The same sampled instances are used across all model configurations to ensure a consistent and fair comparison.

Figure 4.5 and Figure 4.6 summarize the effect of applying each strengthening mechanism individually, as well as of combining them in the Advanced formulation, in the figures crosses denote mean values. Given the limited sample size, these figures are intended to reveal general performance patterns rather than a statistically conclusive comparisons.

Figure 4.5 shows that the lower bounds (LB) provide the most consistent improvement when introduced individually. For $D1$ and $D2$, the LB configuration reduces the final optimality gap to approximately zero, and substantial reductions are also observed for $D3$, $D4$, and $D6$, where the average gap decreases by more than 50% relative to the Basic formulation. The lower bounds therefore appear to perform well overall, even as instance size and difficulty increase.

The effects of the remaining components are less uniform. The upper bounds (UB) yield only limited drive by just that components a noticeable reduction in $D4$, while in the other cases seems not so impactful. The valid inequalities (VI) and symmetry-breaking constraints (SB) produce moderate improvements for some instance sets, particularly $D1$ and $D2$. SB also performs well on some $D3$ instances, but its large dispersion indicates that its effectiveness is strongly instance-dependent.

The Advanced formulation, which combines all strengthening mechanisms, provides the most robust overall gap reduction: it closes the gap almost completely for $D1$ and $D2$, and achieves markedly smaller gaps than the Basic formulation for $D3$, $D4$, and $D6$. Its performance is consistently close to, or better than, that of the LB configuration alone, indicating that LB is the principal source of improvement in solution quality, while the remaining components contribute complementary, instance-specific benefits.

?? shows that gap improvements do not necessarily translate into uniformly shorter solution times, and no single component dominates across all instance sets. For $D1$, the Basic formulation is relatively fast, whereas UB and VI require considerably more time; nonetheless, the Advanced formulation achieves the lowest TTB for this set, indicating that the combined mechanisms can interact beneficially even when some are individually ineffective.

For $D2$, $D3$, and $D4$, UB generally attains the lowest average TTB among the tested configurations, most notably for $D2$ and $D3$, despite not producing the smallest final gaps in these sets. This suggests that UB accelerates the identification of a best solution or bound without providing the same degree of relaxation strengthening as LB. Conversely, LB substantially improves final gaps but does not consistently reduce TTB, particularly for $D2$–$D4$.

VI and SB behave more heterogeneously, with large within-set variation indicating sensitivity to instance structure. SB is relatively effective for some $D1$, $D2$ and $D3$ instances but approaches the time limit for the sampled $D4$ and $D6$ instances; VI performs well for some $D6$ instances but is comparatively slow for $D1$–$D4$.

The Advanced formulation presents a clear quality–time trade-off. It is particularly effective for $D1$ and $D6$, combining low gaps with short TTB, whereas for $D2$, $D3$, and especially $D4$ its TTB is relatively high and in some cases approaches the time limit. In all cases, however, the resulting gaps remain substantially smaller than those of the Basic formulation. The main advantage of the Advanced formulation is therefore not a uniform reduction in com-

putational time, but a more reliable improvement in solution quality across instance sets of varying difficulty.

Overall, this preliminary analysis reveals three main trends: first, LB is the primary driver of improvement in the final optimality gap; second, UB is an important driver in reducing the time required to obtain the best solution; and third, the computational effect of the remaining components is instance-dependent, with their combination potentially entailing a trade-off between solution quality and solution time. Building on these observations, the following subsections examine the upper and lower bounds in greater detail and then compare the Basic and Advanced formulations over the complete benchmark.

Figure 4.5: Component analysis: Gap (%)

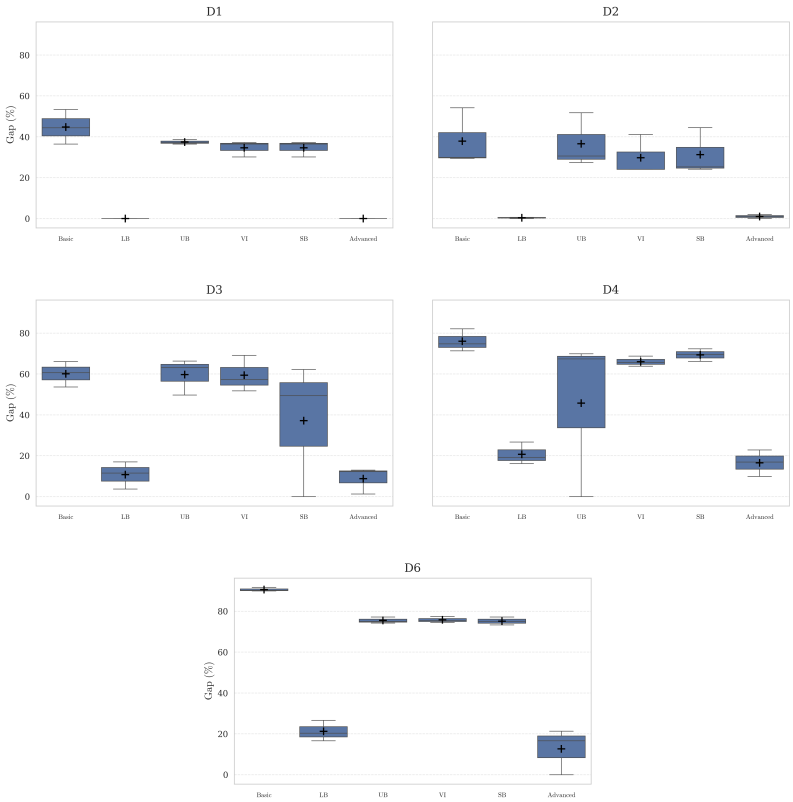
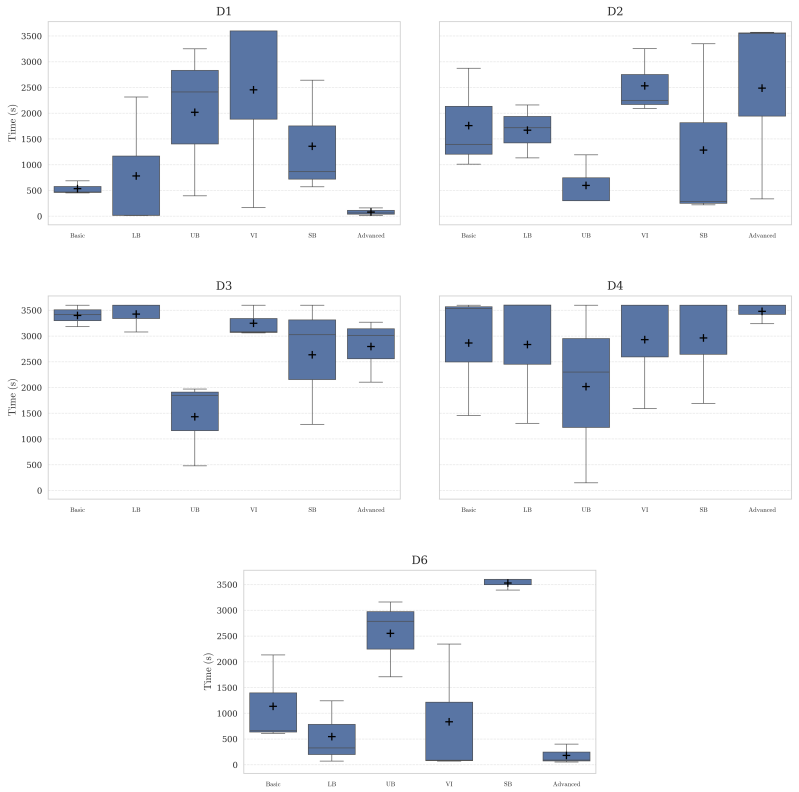


Figure 4.6: Component analysis: Time To Best.



4.5.3 Upper bounds

We now examine one of the key components of the proposed formulation, namely the quality of the upper bounds. In the present setting, upper bounds are particularly important because they affect both the tightness of the formulation, as shown in the constraints: (4.4), (4.5), (4.7), (4.8), (4.9), (4.10), (4.11), (4.12), (4.19), (4.20), and (4.21), and its computational tractability. More precisely, they directly influence the magnitude of the big-M constants appearing in the model. Since overly relaxed big-M values weaken the linear relaxation and may substantially deteriorate solver performance, obtaining tight upper bounds is essential for an effective formulation.

TableTable 4.8 summarizes, for each benchmark set, the average values obtained with the two proposed upper bounds, their corresponding makespan values, the computational times required to compute them, and the percentage reduction achieved by UB_2 with respect to UB_1. The detailed results for each individual instance are reported in Tables 4.9a to 4.9e.

As expected, UB_2 consistently outperforms UB_1 on all instance sets. This behavior is structurally justified by the way the two bounds are constructed. On the one hand, UB_1 is obtained under a fully sequential assumption, and therefore ignores any form of parallelism in the system. On the other hand, UB_2 is derived through a constructive scheduling heuristic that explicitly exploits machine availability, resource compatibility, and the parallel execution opportunities inherent in the problem. As a result, UB_2 is systematically tighter.

The improvement is substantial across all benchmark sets. Considering the average values reported in the tables, the reduction in makespan obtained by moving from UB_1 to UB_2 ranges from approximately 73.9% for the $D1$ instances to about 83.0% for the $D6$ instances. This clearly shows that neglecting parallelism leads to excessively conservative bounds, whereas even a relatively simple constructive procedure is sufficient to obtain tighter bounds.

Focusing on the tightest bound UB_2, its average value increases regularly with the length of the planning horizon, reflecting the natural growth in problem size and workload. In particular, the average makespan for the one-day instances is slightly more than two days. For the two-day instances, the average value increases to approximately 3.3 days. The same trend continues for the larger sets, $D3$ about 4.5 days, $D4$ 5.7 days, and $D6$ roughly 8 days.

These values are coherent with the way the instances are generated. Indeed, each benchmark includes all jobs whose release or start condition falls within a given time window (for example, 24 hours in the one-day case), but the completion of such jobs is not constrained to occur within that same window. Consequently, the final makespan is naturally expected to extend beyond the determined planning horizon. In particular, the fact that even the one-day instances exhibit average completion times beyond 48 hours suggests that, although UB_2 is vastly superior to UB_1, it still remains a weak bound. Therefore, while UB_2 is clearly the more appropriate choice for the formulation, the results indicate that there may still be room for further tightening. The computational times required to obtain the two bounds are, overall, moderate. However, they increase as the instance size grows. For the smallest instances, both bounds are computed in a fraction of a second on average, whereas for the largest set, average times exceed one minute.

The computational time is reported as a first indication on the hardness of the model. All the bounds are executed under a highly simplified setting in which the machine-assignment variables x and the timing variables s and c are fixed. Hence, the solver is not exploring the full decision space of the original scheduling problem. However, the fact that proving optimality may still require a non-negligible amount of time suggests that the resulting model has still some high level of symmetry and degeneracy, making even the proof of optimality computationally challenging, even when solved using the advanced model.

Overall, the analysis confirms that the second bound, UB_2, provides a much tighter and more realistic estimate of the makespan, thereby offering a more suitable basis for defining the big-M coefficients used in the model.

Table 4.8: Average comparison between UB_1 and UB_2 across the benchmark sets.

Instance Set	UB_1		UB_2		$\Delta(\%)$
	$C_{\max}$	Time (s)	$C_{\max}$	Time (s)	$C_{\max}$
$D1$	11285	0.775	2943	0.305	-73.93
$D2$	22570	4.876	4784	4.416	-78.80
$D3$	33856	15.123	6466	11.405	-80.90
$D4$	44971	38.030	8278	24.607	-81.59
$D6$	67711	108.362	11502	119.559	-83.01

Table 4.9: Comparison between UB_1 and UB_2

(a) Instances in set $D1$

Instance	UB_1		UB_2		$\Delta(\%)$
	$C_{\max}$	Time (s)	$C_{\max}$	Time (s)	$C_{\max}$
1	13446	0.170	2531	0.081	-81.18
2	9413	0.270	2510	0.102	-73.33
3	6303	0.124	1865	0.054	-70.41
4	13004	1.476	2839	1.130	-78.17
5	10182	1.122	2592	0.253	-74.54
6	8941	0.581	2562	0.200	-71.35
7	17268	1.193	4056	0.233	-76.51
8	15461	1.446	3318	0.179	-78.54
9	14384	0.979	4174	0.202	-70.98
10	11317	1.415	2481	0.215	-78.08
11	10860	1.264	2520	0.180	-76.80
12	7429	0.214	2434	0.106	-67.24
13	13674	1.231	3928	0.179	-71.27
14	11204	1.404	2737	0.259	-75.57
15	14584	0.285	3898	0.131	-73.27
16	10024	0.227	2824	0.109	-71.83
17	11116	1.754	2447	0.999	-77.99
18	10808	1.730	2396	1.013	-77.83
19	7784	0.205	2562	0.117	-67.09
20	12784	0.403	3780	0.175	-70.43
21	8401	0.210	2515	0.098	-70.06
22	14323	1.296	4004	0.181	-72.04
23	11781	0.297	2785	0.134	-76.36
24	8909	0.461	2309	0.166	-74.08
25	11440	0.718	2458	1.060	-78.51
26	9667	0.176	3798	0.108	-60.71
27	10869	1.616	2562	1.112	-76.43
28	9422	0.227	2916	0.112	-69.05
29	16054	0.530	4023	0.150	-74.94
30	7703	0.229	2454	0.115	-68.14
Avg.	11285	0.775	2943	0.305	-73.93

(b) Instances in set $D2$

Instance	UB_1		UB_2		$\Delta(\%)$
	$C_{\max}$	Time (s)	$C_{\max}$	Time (s)	$C_{\max}$
1	22859	1.724	3685	3.770	-83.88
2	19307	4.806	3687	5.869	-80.90
3	19123	5.543	3973	3.787	-79.22
4	32729	3.409	6106	4.111	-81.34
5	25701	6.172	5546	5.953	-78.42
6	18289	4.832	3950	2.792	-78.40
7	24878	6.395	5550	4.634	-77.69
8	24608	1.886	5464	2.756	-77.80
9	21924	9.167	3861	6.229	-82.39
10	20568	4.267	5272	3.155	-74.37
11	22724	5.141	5471	5.887	-75.92
12	20690	6.440	4016	5.918	-80.59
13	21107	5.682	5253	3.035	-75.11
14	20291	6.012	4425	4.450	-78.19
15	23757	1.663	5508	3.899	-76.82
Avg.	22570	4.876	4784	4.416	-78.80

(c) Instances in set $D3$

Instance	UB_1		UB_2		$\Delta(\%)$
	$C_{\max}$	Time (s)	$C_{\max}$	Time (s)	$C_{\max}$
1	29162	2.197	5209	6.284	-82.14
2	32127	16.396	5832	13.664	-81.85
3	47113	16.420	7521	13.919	-84.04
4	29606	17.011	5363	10.901	-81.89
5	39462	22.055	7636	12.922	-80.65
6	31948	16.698	5696	15.460	-82.17
7	28969	9.458	6747	8.066	-76.71
8	35013	18.408	6965	12.470	-80.11
9	31976	18.005	6725	10.536	-78.97
10	33179	14.582	6969	9.831	-79.00
Avg.	33856	15.123	6466	11.405	-80.90

(d) Instances in set $D4$

Instance	UB_1		UB_2		$\Delta(\%)$
	$C_{\max}$	Time (s)	$C_{\max}$	Time (s)	$C_{\max}$
1	42166	26.561	6973	18.384	-83.46
2	51852	49.780	8866	26.818	-82.90
3	43990	34.652	8269	32.127	-81.20
4	49486	36.654	9033	23.297	-81.75
5	42492	45.992	8150	22.340	-80.82
6	43414	28.933	8430	21.260	-80.58
7	41398	43.636	8222	28.026	-80.14
Avg.	44971	38.030	8278	24.607	-81.59

(e) Instances in set $D6$

Instance	UB_1		UB_2		$\Delta(\%)$
	$C_{\max}$	Time (s)	$C_{\max}$	Time (s)	$C_{\max}$
1	61289	85.401	9891	112.208	-83.86
2	76719	112.872	12207	123.714	-84.09
3	71410	114.508	12197	180.714	-82.92
4	63982	74.730	11809	73.914	-81.54
5	65155	154.297	11405	107.244	-82.50
Avg.	67711	108.362	11502	119.559	-83.01

4.5.4 Lower bounds

Another fundamental aspect of our model concerns the impact of the lower bounds. Unlike the upper bounds, the lower bounds do not directly affects the constraints of the formulation, instead, they contribute by reducing the effective search space. This reduction becomes particularly relevant when proving optimality, as tighter lower bounds shrink the optimality gap. To evaluate their effectiveness, we analyze Tables 4.11a to 4.11e. These tables report all the defined lower bounds together with the linear relaxation values of both the basic and advanced models. The LP relaxations serve as a reference point and allow us to assess the strength of the valid inequalities introduced. The best lower bound is also highlighted in bold.

Starting from the LP relaxations, we observe an average improvement from LP_{basic} to $LP_{advanced}$ of approximately 40.8% for $D1$, 38.8% for $D2$, 30.1% for $D3$, 32.2% for $D4$, and 24.6% for $D6$. These results confirm that the valid inequalities provide a non-negligible strengthening of the formulation. However, independently of the time horizon, both LP relaxations remain weak. Their values are several orders of magnitude smaller than the ad-hoc lower bounds. This highlights that proving optimality remains challenging due to the intrinsic weakness of the underlying relaxation.

We now turn to the comparison among the three lower bounds. The most interesting behavior is observed in set $D1$, where the dominance among bounds is not consistent. In this set, the longest job bound (LB_1), despite its simplicity, can outperform the others in some instances. This can be explained by the presence of stages with significantly larger processing times (e.g., solidification), as shown in Table 4.7. In contrast, the single bottleneck bound (LB_2) is generally weaker than LB_1 in this setting. The combined bound (LB_3), which incorporates both the bottleneck and the minimum residual processing time of subsequent stages, provides the best performance on average.

This variability is primarily driven by the small number of jobs in $D1$, where a single long job can dominate the schedule. As the instance size increases (sets $D2$–$D6$), the behavior becomes more stable, LB_2 consistently outperforms LB_1, and LB_3 is always the strongest bound. This is expected, since with more jobs the bottleneck workload naturally exceeds the processing time of any single job, and the additional refinement in LB_3 further strengthens the estimate. The percentage improvements reported in Table 4.10 confirm this trend. We observe a substantial increase when moving from LB_1 to LB_3, while the improvement from LB_2 to LB_3 is moderate.

Table 4.10: Percentage improvements between lower bounds (LB_1, LB_2, LB_3)

Instance	(LB_1,LB_2)	(LB_2,LB_3)	(LB_1,LB_3)
D1	-9.75%	26.39%	14.08%
D2	46.10%	12.38%	64.21%
D3	108.39%	8.24%	125.60%
D4	138.65%	5.89%	152.68%
D6	262.63%	3.79%	276.37%

Overall, these observations confirm that LB_3 provides a strong and reliable estimate of the optimal solution, especially as the problem size increases.

Table 4.11: Linear relaxation and Lower Bounds values comparison

(a) Instances in set $D1$

Instance	LP_{basic}	$LP_{advanced}$	LB_1	LB_2	LB_3
1	140.00	232.00	**1534**	982	1438
2	360.00	594.00	1390	1187	**1607**
3	733.00	733.00	1160	1438	**1849**
4	480.00	899.00	1095	1456	**1799**
5	234.00	278.50	1341	1469	**1794**
6	123.00	158.00	1245	1347	**1710**
7	317.14	379.86	**2659**	1441	1824
8	228.00	634.00	1536	1509	**1881**
9	317.14	469.00	**2501**	1448	1859
10	136.00	172.00	1110	1399	**1766**
11	360.00	565.00	1167	1426	**1760**
12	589.00	589.00	1078	1430	**1821**
13	360.00	547.00	**2484**	1521	1910
14	135.00	146.00	1402	1405	**1792**
15	558.00	928.00	**2489**	1379	1785
16	401.00	401.00	1263	1463	**1808**
17	160.00	295.33	1047	1472	**1876**
18	120.00	155.00	1071	1400	**1708**
19	510.00	541.50	1158	1492	**1834**
20	399.00	595.00	**2466**	1394	1785
21	399.00	628.00	1256	1411	**1783**
22	360.00	538.50	**2503**	1504	1884
23	250.00	431.00	1212	1514	**1907**
24	166.00	321.00	1090	1335	**1693**
25	108.00	165.00	1090	1443	**1806**
26	740.00	810.33	**2431**	1415	1800
27	146.00	223.00	1186	1472	**1858**
28	271.00	456.00	1535	1416	**1735**
29	332.00	504.00	**2533**	1478	1911
30	459.00	533.00	1066	1455	**1753**
Avg.	329.71	464.07	1570	1417	**1791**

(b) Instances in set $D2$

Instance	LP_{basic}	$LP_{advanced}$	LB_1	LB_2	LB_3
1	360.00	594.00	1534	2169	**2589**
2	733.00	899.00	1160	2894	**3237**
3	234.00	278.50	1341	2816	**3141**
4	317.14	634.00	2659	2950	**3322**
5	317.14	469.00	2501	2847	**3214**
6	589.00	589.00	1167	2856	**3190**
7	360.00	547.00	2484	2926	**3313**
8	558.00	928.00	2489	2842	**3187**
9	160.00	295.33	1071	2872	**3180**
10	510.00	595.00	2466	2886	**3228**
11	399.00	628.00	2503	2915	**3287**
12	250.00	431.00	1212	2849	**3207**
13	740.00	810.33	2431	2858	**3221**
14	271.00	456.00	1535	2888	**3207**
15	459.00	533.00	2533	2933	**3231**
Avg.	417.15	579.14	1939	2833	**3184**

(c) Instances in set $D3$

Instance	LP_{basic}	$LP_{advanced}$	LB_1	LB_2	LB_3
1	733.00	733.00	1534	3607	**4018**
2	480.00	899.00	1341	4272	**4597**
3	317.14	634.00	2659	4398	**4770**
4	589.00	589.00	1167	4255	**4589**
5	558.00	928.00	2489	4305	**4692**
6	401.00	401.00	1263	4335	**4643**
7	510.00	628.00	2466	4297	**4639**
8	360.00	538.50	2503	4353	**4711**
9	740.00	810.33	2431	4330	**4693**
10	459.00	533.00	2533	4349	**4647**
Avg.	514.71	669.38	2039	4250	**4600**

(d) Instances in set $D4$

Instance	LP_{basic}	$LP_{advanced}$	LB_1	LB_2	LB_3
1	733.00	899.00	1534	5063	**5406**
2	317.14	634.00	2659	5766	**6091**
3	589.00	589.00	2501	5703	**6037**
4	558.00	928.00	2489	5768	**6113**
5	510.00	595.00	2466	5758	**6066**
6	399.00	628.00	2503	5764	**6122**
7	740.00	810.33	2431	5746	**6065**
Avg.	549.45	726.19	2369	5653	**5986**

(e) Instances in set $D6$

Instance	LP_{basic}	$LP_{advanced}$	LB_1	LB_2	LB_3
1	733.00	899.00	1534	7879	**8204**
2	589.00	634.00	2659	8653	**8987**
3	558.00	928.00	2489	8640	**8948**
4	510.00	628.00	2503	8650	**8992**
5	740.00	810.33	2533	8679	**8977**
Avg.	626.00	779.87	2344	8500	**8822**

4.5.5 Models comparison

The computational results provide a clear assessment of the impact of strengthening the mathematical formulation through the introduction of lower bounds, improved initial upper bounds, valid inequalities, and symmetry-breaking constraints. Although the basic and advanced models share the same objective function and therefore target identical optimal solutions, their behavior during the solution process differs substantially.

Before discussing the results in detail, it is important to clarify the notation adopted in the tables. In all tables, bold values indicate instances for which the optimal solution has been proven by both models, and underlined bold value indicates that optimality was proven only by the advanced model.

Table 4.12 complements the per-instance tables (Tables 4.13 to 4.17) by reporting, for each instance set, the average values across all corresponding instances, providing a concise overview of how the performance gap between the two models evolves as instance size increases.

For the smallest instances (set $D1$), both models frequently identify solutions with identical or nearly identical values of $C_{\max}$. However, the main difference lies in the quality of the search process and, in particular, in the ability to certify optimality. In this set, optimality is proven in 23 out of 30 instances overall. Of these, only 7 instances are solved to proven optimality by the basic model, while 16 additional instances are certified as optimal only by the advanced formulation.

It is fundamental to understand that, in 21 instances, the best lower bounds directly correspond to the optimal solutions. This means that, in these cases, the optimal value was already identified by the bound itself, although proving optimality remained challenging. Within this subset of instances, 15 optimal solutions are provided by LB_3, while the remaining 6 are obtained from LB_1. The advanced model also reduces the average optimality gap from 26.61% to 0.77%, demonstrating a significantly stronger formulation. In addition, the average time to best solution (TTB) decreases by approximately 29%, although this improvement is not uniform across all instances. Overall, the advanced model tends to reach high-quality solutions earlier and provides substantially stronger optimality guarantees, even when the final solution value remains unchanged.

A similar pattern is observed for the $D2$ instances. While the average objective values remain very close between the two formulations, the optimality gap is drastically reduced from 44.12% to 2.11%. This confirms that, for small-to-medium instances, the primary benefit of the advanced model lies in strengthening the formulation, allowing the solver to certify optimality more frequently and reducing the computational effort required to obtain high-quality solutions. The improvements in TTB are more moderate and exhibit variability across instances.

One important remark regarding the $D2$ instance set deserves further discussion. In instances 11 and 12, the basic model is able to identify the optimal solution; however, due to the weakness of the lower bounds computed during the solution process, the solver is unable to certify optimality and terminates at the time limit. In contrast, the advanced model successfully finds and proves optimality for instance 11, whereas for instance 12 it fails to reach the same solution, thereby missing the optimal value.

Although this behavior may seem counterintuitive at first, it is not unexpected. The advanced model provides a stronger formulation, but this comes at the cost of a significantly increased number of constraints, which raises the computational complexity of the problem. Moreover, the addition of constraints substantially alters the underlying structure and coefficient matrix of the model. As a result, even though the formulation is theoretically stronger,

it may in some cases lead to worse computational performance on specific instances.

Moving to the $D3$ instances, the transition becomes evident. At this scale, the advanced model not only significantly reduces the optimality gap from 66.70% to 7.24%, but also achieves better objective values on average, with an improvement of about 3.1% in $C_{\max}$. This indicates that the strengthened formulation is no longer only helping to prove optimality, but is also effectively guiding the solver toward better regions of the solution space within the imposed time limit. Moreover, high-quality solutions are generally found earlier in the search process.

For larger instances ($D4$ and $D6$), the best performance of the advanced model becomes pronounced. The basic formulation struggles to produce competitive solutions, as indicated by very large optimality gaps, above 75% on average for $D4$ and close to 90% for $D6$. In contrast, the advanced model substantially reduces both the gap and the objective value. In particular, the average $C_{\max}$ is reduced by about 29% for $D4$ and more than 56% for $D6$. These results highlight the critical role of valid inequalities and symmetry-breaking constraints in improving scalability, as they effectively reduce the search space and enable the solver to focus on more promising regions.

An important observation is that the benefits of the advanced formulation are strongly dependent on the problem size. For smaller instances, both models often converge to similar objective values, and in a few cases the basic formulation may even yield slightly better solutions. However, the advanced model remains much more reliable and significantly more effective in proving optimality. As the instance size increases, the advantage of the advanced formulation becomes increasingly evident: it not only proves optimality more frequently, but also consistently finds better solutions within the time limit.

Overall, these results confirm that the advanced model provides a much tighter and more effective representation of the problem. This leads to improved bounds, stronger optimality certification, faster convergence on average, and substantially better scalability, making it a more suitable approach for solving larger and more complex instances.

Table 4.12: Summary of average results across instance sets.

Instance Set	Basic				Advanced				$\Delta C_{\max}$		ΔTTB	
	$C_{\max}$	Gap (%)	Time (s)	TTB (s)	$C_{\max}$	Gap (%)	Time (s)	TTB (s)	*Absolute*	*Percentage*	*Absolute*	*Percentage*
$D1$	1988.87	26.61	2644.34	954.92	1990.47	0.77	904.82	385.80	1.60	0.09	-569.12	-29.43
$D2$	3244.80	44.12	3600.00	2525.23	3253.73	2.11	3382.32	2039.99	8.93	0.31	-485.24	-7.58
$D3$	5156.30	66.70	3600.00	3295.02	4984.00	7.24	3600.00	2837.02	-172.30	-3.10	-458.00	-13.42
$D4$	10856.29	77.77	3600.00	2878.48	7213.29	16.44	3600.00	3114.52	-3643.00	-29.08	236.03	20.52
$D6$	26826.60	89.90	3600.00	1325.08	11490.20	17.62	3094.63	130.70	-15336.40	-56.62	-1194.38	-78.90

Table 4.13: Basic and advanced model comparison for $D1$ instances.

Instance	Basic				Advanced				ΔC_{max}		ΔTTB	
	C_{max}	Gap (%)	Time (s)	TTB (s)	C_{max}	Gap (%)	Time (s)	TTB (s)	*Absolute*	*Percentage*	*Absolute*	*Percentage*
1	1807.00	15.11	3600.00	229.85	1807.00	10.46	3600.00	258.94	0.00	0.00	29.09	12.66
2	**1607.00**	13.50	3600.00	1377.11	**1607.00**	**0.00**	52.43	52.41	0.00	0.00	-1324.70	-96.19
3	**1849.00**	40.40	3600.00	2392.29	**1849.00**	**0.00**	5.05	5.03	0.00	0.00	-2387.26	-99.79
4	1842.00	45.60	3600.00	3072.49	1861.00	3.33	3600.00	2050.52	19.00	1.03	-1021.97	-33.26
5	**1794.00**	38.63	3600.00	2093.32	**1794.00**	**0.00**	62.28	62.25	0.00	0.00	-2031.07	-97.03
6	1721.00	32.39	3600.00	1915.55	1721.00	0.64	3600.00	425.41	0.00	0.00	-1490.14	-77.79
7	**2712.00**	0.00	28.60	28.52	**2712.00**	**0.00**	43.91	41.84	0.00	0.00	13.32	46.70
8	1910.00	31.06	3600.00	1737.59	1910.00	1.52	3600.00	2688.29	0.00	0.00	950.70	54.71
9	**2501.00**	0.00	5.11	5.08	**2501.00**	**0.00**	5.21	5.17	0.00	0.00	0.09	1.77
10	**1766.00**	53.37	3600.00	468.66	**1766.00**	**0.00**	161.76	161.72	0.00	0.00	-306.94	-65.49
11	1786.00	36.73	3600.00	2521.07	1794.00	1.90	3600.00	56.00	8.00	0.45	-2465.07	-97.78
12	**1821.00**	40.80	3600.00	2285.44	**1821.00**	**0.00**	72.55	72.52	0.00	0.00	-2212.92	-96.83
13	**2484.00**	0.00	55.60	6.99	**2484.00**	**0.00**	2.76	2.76	0.00	0.00	-4.23	-60.52
14	**1792.00**	21.76	3600.00	1104.58	**1792.00**	**0.00**	29.97	29.92	0.00	0.00	-1074.66	-97.29
15	**2489.00**	0.00	8.45	8.30	**2489.00**	**0.00**	2.44	2.41	0.00	0.00	-5.89	-70.96
16	**1808.00**	44.40	3600.00	451.37	**1808.00**	**0.00**	16.09	16.05	0.00	0.00	-435.32	-96.44
17	**1876.00**	56.17	3600.00	1683.91	**1876.00**	**0.00**	882.65	882.57	0.00	0.00	-801.34	-47.59
18	1762.00	39.23	3600.00	2736.48	1783.00	4.21	3600.00	2788.49	21.00	1.19	52.01	1.90
19	**1834.00**	40.95	3600.00	110.48	**1834.00**	**0.00**	48.73	48.71	0.00	0.00	-61.77	-55.91
20	**2466.00**	0.00	6.57	2.79	**2466.00**	**0.00**	2.16	2.15	0.00	0.00	-0.64	-22.94
21	**1783.00**	29.56	3600.00	1066.88	**1783.00**	**0.00**	16.58	16.55	0.00	0.00	-1050.33	-98.45
22	**2503.00**	0.00	10.81	4.11	**2503.00**	**0.00**	4.33	4.30	0.00	0.00	0.19	4.62
23	**1907.00**	36.44	3600.00	686.23	**1907.00**	**0.00**	62.32	62.29	0.00	0.00	-623.94	-90.92
24	**1693.00**	43.12	3600.00	63.34	**1693.00**	**0.00**	5.42	5.39	0.00	0.00	-57.95	-91.49
25	**1806.00**	51.49	3600.00	2060.60	**1806.00**	**0.00**	333.95	333.89	0.00	0.00	-1726.71	-83.80
26	**2431.00**	0.00	2.15	0.89	**2431.00**	**0.00**	0.81	0.80	0.00	0.00	-0.09	-10.11
27	**1858.00**	36.17	3600.00	220.56	**1858.00**	**0.00**	56.57	56.53	0.00	0.00	-164.03	-74.37
28	**1735.00**	11.53	3600.00	126.29	**1735.00**	**0.00**	42.77	42.75	0.00	0.00	-83.54	-66.15
29	**2551.00**	0.00	12.85	6.85	**2551.00**	**0.00**	33.98	3.48	0.00	0.00	-3.37	-49.20
30	1772.00	39.84	3600.00	179.98	1772.00	1.07	3600.00	1394.73	0.00	0.00	1214.75	674.94
Avg.	1988.87	26.61	2644.34	954.92	1990.47	0.77	904.82	385.80	1.60	0.09	-569.12	-29.43

Table 4.14: Basic and advanced model comparison for $D2$ instances.

Instance	Basic				Advanced				ΔC_{max}		ΔTTB	
	C_{max}	Gap (%)	Time (s)	TTB (s)	C_{max}	Gap (%)	Time (s)	TTB (s)	*Absolute*	*Percentage*	*Absolute*	*Percentage*
1	2593.00	54.18	3600.00	1010.38	2614.00	0.96	3600.00	3552.71	21.00	0.81	2542.33	251.62
2	3351.00	66.75	3600.00	2434.89	3321.00	2.53	3600.00	1215.20	-30.00	-0.90	-1219.69	-50.09
3	3202.00	66.24	3600.00	1109.23	3213.00	2.24	3600.00	1476.60	11.00	0.34	367.37	33.12
4	3567.00	25.46	3600.00	2525.92	3434.00	3.26	3600.00	3348.29	-133.00	-3.73	822.37	32.56
5	3218.00	28.25	3600.00	2204.48	3284.00	2.13	3600.00	2878.36	66.00	2.05	673.88	30.57
6	3244.00	64.94	3600.00	2895.66	3244.00	1.66	3600.00	805.19	0.00	0.00	-2090.47	-72.19
7	3340.00	25.97	3600.00	3582.01	3314.00	0.03	3600.00	3566.15	-26.00	-0.78	-15.86	-0.44
8	3240.00	29.41	3600.00	2873.43	3249.98	1.94	3600.00	3571.06	9.98	0.31	697.63	24.28
9	3412.00	61.05	3600.00	3547.91	3441.00	7.58	3600.00	3600.00	29.00	0.85	52.09	1.47
10	3239.00	28.95	3600.00	1697.08	3289.00	1.85	3600.00	144.25	50.00	1.54	-1552.83	-91.50
11	**3287.00**	29.91	3600.00	1394.87	**3287.00**	**0.00**	334.75	334.67	0.00	0.00	-1060.20	-76.01
12	3207.00	62.21	3600.00	2977.66	3224.00	0.53	3600.00	1651.82	17.00	0.53	-1325.84	-44.53
13	3247.00	25.13	3600.00	3600.00	3235.00	0.43	3600.00	383.31	-12.00	-0.37	-3216.69	-89.35
14	3230.00	63.34	3600.00	3067.24	3316.00	3.29	3600.00	720.26	86.00	2.66	-2346.98	-76.52
15	3295.00	30.02	3600.00	2957.71	3340.00	3.26	3600.00	3351.96	45.00	1.37	394.25	13.33
Avg.	3244.80	44.12	3600.00	2525.23	3253.73	2.11	3382.32	2039.99	8.93	0.31	-485.24	-7.58

Table 4.15: Basic and advanced model comparison for *D*3 instances.

Instance	Basic				Advanced				ΔC_{max}		ΔTTB	
	C_{max}	Gap (%)	Time (s)	TTB (s)	C_{max}	Gap (%)	Time (s)	TTB (s)	*Absolute*	*Percentage*	*Absolute*	*Percentage*
1	4083.00	68.24	3600.00	3589.99	4056.00	0.94	3600.00	1474.44	-27.00	-0.66	-2115.55	-58.93
2	5482.00	71.12	3600.00	3455.03	5196.00	11.53	3600.00	3073.88	-286.00	-5.22	-381.15	-11.03
3	6030.00	60.63	3600.00	3600.00	5434.00	12.22	3600.00	2103.78	-596.00	-9.88	-1496.22	-41.56
4	5102.00	69.40	3600.00	3244.06	4749.00	3.37	3600.00	3588.99	-353.00	-6.92	344.93	10.63
5	5887.00	60.95	3600.00	3340.81	5683.00	17.44	3600.00	3423.61	-204.00	-3.47	82.80	2.48
6	5079.00	66.05	3600.00	3416.41	5334.00	12.95	3600.00	3267.80	255.00	5.02	-148.61	-4.35
7	4726.00	47.82	3600.00	2853.53	4716.00	1.63	3600.00	1613.52	-10.00	-0.21	-1240.01	-43.46
8	5017.00	50.11	3600.00	3121.99	5158.00	8.67	3600.00	3600.00	141.00	2.81	478.01	15.31
9	5242.00	53.62	3600.00	3182.71	4754.00	1.28	3600.00	3014.44	-488.00	-9.31	-168.27	-5.29
10	4915.00	48.46	3600.00	3145.68	4760.00	2.37	3600.00	3209.72	-155.00	-3.15	64.04	2.04
Avg.	5156.30	66.70	3600.00	3295.02	4984.00	7.24	3600.00	2837.02	-172.30	-3.10	-458.00	-13.42

Table 4.16: Basic and advanced model comparison for *D*4 instances.

Instance	Basic				Advanced				ΔC_{max}		ΔTTB	
	C_{max}	Gap (%)	Time (s)	TTB (s)	C_{max}	Gap (%)	Time (s)	TTB (s)	*Absolute*	*Percentage*	*Absolute*	*Percentage*
1	7013.00	74.35	3600.00	2967.30	5692.00	5.02	3600.00	3566.93	-1321.00	-18.84	599.63	20.21
2	9402.00	74.75	3600.00	3600.00	7893.00	22.83	3600.00	3239.48	-1509.00	-16.05	-360.52	-10.01
3	8902.00	74.06	3600.00	3363.29	7745.00	22.05	3600.00	1861.53	-1157.00	-13.00	-1501.76	-44.65
4	12831.00	82.08	3600.00	1454.80	7358.00	16.92	3600.00	3600.00	-5473.00	-42.65	2145.20	147.46
5	12952.00	82.23	3600.00	2013.86	7463.00	18.72	3600.00	2385.19	-5489.00	-42.38	371.33	18.44
6	8032.00	71.31	3600.00	3535.37	6793.00	9.88	3600.00	3600.00	-1239.00	-15.43	64.63	1.83
7	16862.00	85.58	3600.00	3214.76	7549.00	19.66	3600.00	3548.49	-9313.00	-55.23	333.73	10.38
Avg.	10856.29	77.77	3600.00	2878.48	7213.29	16.44	3600.00	3114.52	-3643.00	-29.08	236.03	20.52

Table 4.17: Basic and advanced model comparison for *D*6 instances.

Instance	Basic				Advanced				ΔC_{max}		ΔTTB	
	C_{max}	Gap (%)	Time (s)	TTB (s)	C_{max}	Gap (%)	Time (s)	TTB (s)	*Absolute*	*Percentage*	*Absolute*	*Percentage*
1	28945.00	91.64	3600.00	609.75	9833.00	16.57	3600.00	399.83	-19112.00	-66.03	-209.92	-34.43
2	23703.00	88.13	3600.00	340.38	12207.00	26.38	3600.00	75.77	-11496.00	-48.50	-264.61	-77.74
3	30871.00	89.92	3600.00	660.95	**12197.00**	**0.00**	1073.14	93.16	-18674.00	-60.49	-567.79	-85.91
4	24294.00	89.62	3600.00	2881.84	11809.00	23.85	3600.00	32.11	-12485.00	-51.39	-2849.73	-98.89
5	26320.00	90.19	3600.00	2132.48	11405.00	21.29	3600.00	52.62	-14915.00	-56.67	-2079.86	-97.53
Avg.	26826.60	89.90	3600.00	1325.08	11490.20	17.62	3094.63	130.70	-15336.40	-56.62	-1194.38	-78.90

As a final remark, the optimal schedule obtained for the hybrid flow shop in the case of $D1$, instance 2, is illustrated in Figures 4.7 to 4.9. The figures show, respectively, the overall job sequence, the utilization of the processing machines and ladles, and the allocation of the solidification pits.

Figure 4.7: Classical Gantt chart of the optimal solution.

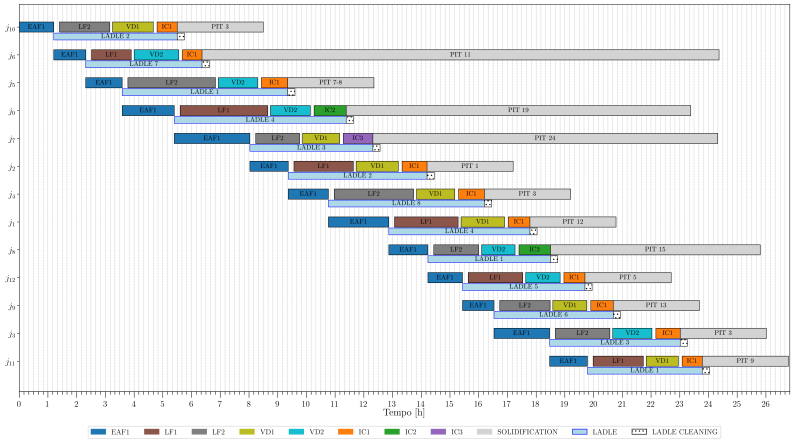


Figure 4.8: Machine and ladle utilization in the optimal solution.

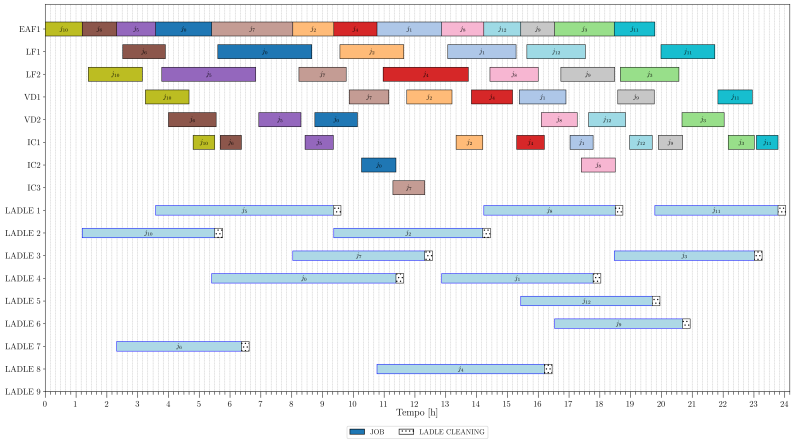
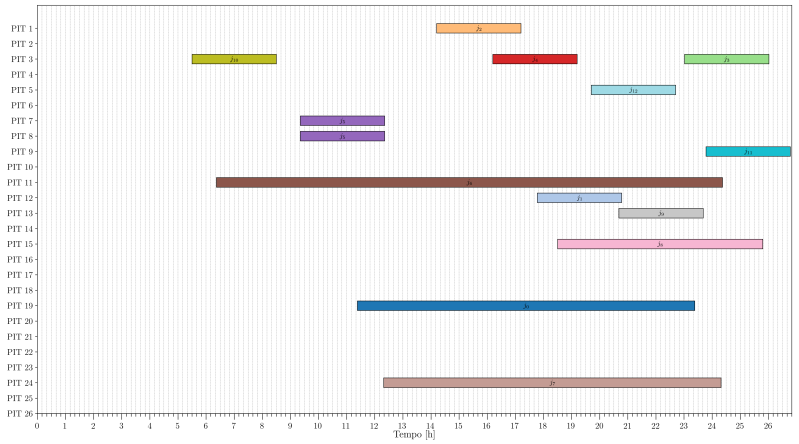


Figure 4.9: Solidification pit allocation in the optimal solution.



4.6 Conclusions

In this chapter, we investigated the Hybrid Flow Shop for Steelmaking and Ingot Casting (HFS-SIC), incorporating the key operational features introduced earlier, namely multiple processing stages with parallel machines, machine eligibility constraints, transportation times, strict no-wait conditions, and the allocation of shared auxiliary resources such as ladles and the ingot management related to the solidification pits. The proposed model explicitly captures these characteristics and provides a realistic representation of the underlying industrial process.

Two mixed-integer linear programming (MILP) formulations were developed to represent the problem, one with the minimal structure necessary for the correctness of the model, and one advanced with the introduction of additional constraints. To improve the formulation in the advanced model, several strengthening techniques were introduced, including valid inequalities, tighter bounds, and symmetry-breaking constraints, which significantly enhance the quality and tractability of the formulation. In addition, two valid upper bounds were developed to generate feasible solutions: a purely sequential scheduling strategy without parallelism, and a constructive heuristic based on a shortest-job-first rule that exploits parallel machine assignment.

The computational experiments provide clear evidence of the effectiveness of the proposed approach. The results show that the advanced formulation substantially improves the solver's performance in terms of both solution quality and computational time. In particular, the model is capable of solving instances of practical relevance and consistently producing high-quality schedules with reduced makespan values. The comparison between the baseline and strengthened formulations highlights the critical role of the proposed enhancements, as they lead to faster convergence and tighter optimality gaps.

From a bounding perspective, the proposed upper bounds provide meaningful and computationally inexpensive feasible solutions, which serve as effective benchmarks for evaluating the quality of the MILP solutions. Moreover, they contribute to improving the tightness of the LP relaxation of the model, particularly given the extensive use of big-M constraints in the formulation.

At the same time, the lower bounds enable a rigorous assessment of the optimality gap

and the overall solution quality. The combination of improved upper bounds and tighter lower bounds highlights the critical role of such bounding strategies in strengthening the formulation, as evidenced by the reduced optimality gaps observed in the computational experiments. This confirms that the proposed approach consistently produces high-quality feasible solutions while providing strong guarantees on their optimality.

In conclusion, the proposed enhanced MILP formulation constitutes an effective and realistic optimization framework for the HFS-SIC. It combines modeling accuracy with strong computational performance, making it a valuable decision-support tool for steelmaking and ingot casting operations. Future research could further enhance the proposed framework by relaxing some of the assumptions adopted in the current study. In particular, the model addresses a deterministic offline scheduling problem in which processing times, resource availability, and production requirements are assumed to be known in advance. While this setting enables an accurate representation of the production process and the assessment of different modeling enhancements, real steelmaking plants are subject to unexpected machine failures, processing delays, changes in production priorities, and other operational disruptions that require dynamic schedule adaptations. Incorporating these sources of uncertainty through stochastic or robust optimization techniques, together with rolling-horizon or real-time rescheduling approaches capable of updating production plans as new information becomes available, would further improve the practical applicability of the proposed methodology. Another promising research direction is the integration of energy-aware scheduling objectives alongside makespan minimization, supporting more sustainable and cost-efficient production planning.

Part II

Distribution scheduling

Chapter 5

Towards Carbon-Neutral Last-Mile Delivery: Balancing Environmental Impact and Service Quality with Autonomous Robots

The content of the following chapter has not yet been published. The work has been submitted for consideration to the journal Computers & Industrial Engineering and is currently under review.

Contents

5.1 Introduction . . . 117
5.2 Related literature . . . 119
5.3 Problem Definition and Mathematical Formulation . . . 121
5.3.1 Modeling Assumptions . . . 124
5.4 Solution Method . . . 125
5.4.1 Bi-Objective approach . . . 125
5.4.2 Two-Phase Kernel Search . . . 126
5.5 Experimental Analysis . . . 128
5.5.1 Benchmark instances . . . 129
5.5.2 Managerial Insights and Sensitivity Analysis . . . 131
5.5.3 2P-KS performance analysis . . . 136
5.5.4 Analysis of a realistic scenario . . . 138
5.6 Conclusions and Future Work . . . 141

The rapid growth of e-commerce has contributed to the environmental and operational challenges associated with last-mile delivery. This chapter examines the potential of integrating autonomous delivery robots (ADRs) alongside fully electric vehicles to facilitate the transition towards carbon-neutral urban logistics, while ensuring that service quality remains at acceptable levels. The Green Two-Tier Robot-Aided Delivery Problem (G2T-RADP)

is defined as a bi-objective MILP aimed at concurrently minimizing greenhouse gas emissions and the number of late deliveries. The model focuses on the environmental impact while addressing facility compatibility constraints and granting soft time windows for customers. In order to tackle large-scale instances, we implemented the Two-Phase Kernel Search (2P-KS) heuristic, which effectively balances the exploration of feasibility alongside the improvement of solutions. Comprehensive computational experiments conducted on randomly generated instances reveal that minor sacrifices in service quality (an increase of 15% or less in late deliveries) can yield environmental costs very close to the unconstrained emission optimum, revealing substantial potential for decarbonizing last-mile operations. The example of Brescia, Italy, demonstrates the managerial and policy implications associated with the implementation of shared electric fleets and ADRs for sustainable last-mile delivery, while also validating the feasibility of the approach within a realistic urban network. The findings indicate that collaborative efforts between public and private sectors can markedly decrease emissions from urban logistics while maintaining service reliability, presenting a feasible approach to achieving CO_2-neutral city logistics.

5.1 Introduction

The rapid growth of e-commerce (Weltevreden, 2025) has significantly reshaped urban logistics, increasing the number of parcels to be delivered and generating operational and environmental challenges. The last-mile segment accounts for a major share of total logistics costs and emissions (Ranieri et al., 2018), making its optimization a key priority in both operations research and sustainable transportation.

Beyond efficiency and cost minimization, the expansion of e-commerce has transformed customer expectations. Consumers now demand deliveries that are not only fast but also reliable, flexible, and sustainable. Service quality is increasingly tied to delivering at the right time and at the right place (Zhong et al., 2022; Amorim et al., 2024). However, meeting these expectations is often hindered by synchronization issues between couriers and customers, which frequently result in missed deliveries, additional trips, and diminished perceived service quality (Bonomi et al., 2025; Zanotti et al., 2025). These inefficiencies highlight the limitations of conventional logistics and have driven the search for adaptive delivery solutions capable of streamlining courier-customer synchronization and mitigating the operational costs of idle waiting times.

The adoption of autonomous delivery robots (ADRs) has emerged as a promising response to these evolving needs. Technological advances in navigation, sensing, and obstacle detection have made these systems increasingly reliable and suitable for dense urban contexts (Boysen et al., 2021; Gajda et al., 2024). Concurrently, public health concerns intensified by the COVID-19 pandemic have accelerated interest in contactless logistics, with recent studies indicating many online shoppers are willing to pay a premium for autonomous, environmentally friendly delivery services (Pani et al., 2020). ADRs offer distinct advantages in this context: operating at moderate speeds (about 5–6 km/h) in pedestrian areas, they ensure safety while efficiently delivering small parcels with limited noise, energy consumption, and direct emissions due to their electric propulsion (Chen and Demir, 2022). Given these characteristics, ADRs are gaining traction in niche markets like food and medical supplies, and can even be integrated into broader urban mobility frameworks (Mourad et al., 2021; De Maio et al., 2024).

However, realizing the full potential of ADRs requires a system-level perspective. Indirect emissions from the electricity generation required to charge ADRs can still represent up

to 40% of the emissions produced by traditional diesel-based systems (Lemardelé et al., 2023). To achieve true efficiency and carbon neutrality (Koh et al., 2023), aligning with the European Union's target of CO_2-free city logistics by 2030 (Kallas, 2011), the limited operational range of ADRs must be addressed through integrated network designs. Specifically, pairing them with electric trucks, which act as mobile carriers to replenish local facilities, creates a powerful two-tier delivery system that combines the environmental benefits of electric propulsion with the operational flexibility required for modern urban logistics.

To practically implement these two-tier networks, municipalities are increasingly exploring collaborative urban logistics frameworks centered around shared Urban Consolidation Centers (UCCs) managed through public-private partnerships (Allen et al., 2012). Under a Robot-as-a-Service (RaaS) or shared-capacity model, the local government manages a shared infrastructure of municipal hubs and robot fleets (Grote et al., 2023). Delivery companies can access this infrastructure via a fixed subscription, allowing them to route their daily last-mile operations through a subset of available public hubs. To incentivize private logistics providers to adopt this shared system, the municipality must offer a broad array of candidate facilities, ensuring companies retain the operational flexibility needed for efficient routing. However, to balance this flexibility with public goals of emissions reduction and congestion mitigation, the municipality restricts the simultaneous activation of certain hubs. Specifically, due to spatial proximity, charging capacity limits, and safety regulations, facilities with overlapping service territories are defined as mutually incompatible. From a modeling perspective, these incompatibilities introduce a complex combinatorial layer of constraints into the selection of facilities. This structure transforms the delivery planning problem into a conflict-constrained two-tier vehicle routing problem.

To address these compounding operational and environmental challenges, we introduce the Green Two-Tier Robot-Aided Delivery Problem (G2T-RADP). The G2T-RADP is formulated as a bi-objective Mixed-Integer Linear Program (MILP) that concurrently minimizes the environmental cost of delivery operations and maximizes a service-quality measure related to soft delivery deadlines. The problem determines the optimal subset of compatible municipal hubs to activate, alongside the routing plan for an electric truck that replenishes these hubs before the robots execute last-mile deliveries.

We depart from existing literature by assuming a fully electric, two-tier fleet and explicitly omitting the acquisition costs of facilities and robots. This design reflects the operational reality of the municipal RaaS model, where infrastructure is offered as an incentivized subscription rather than a capital expenditure for private companies. Furthermore, the problem captures the joint optimization of facility activation under conflict constraints while accounting for the indirect greenhouse gas emissions generated by the electricity consumption of both the truck and the robots.

To evaluate the interplay between the conflicting goals of environmental sustainability and service reliability, we employ a two-step lexicographic approach. This strategy allows decision makers to systematically explore the trade-offs between minimizing emissions and avoiding late deliveries. While commercial solvers can optimally resolve these trade-offs for medium-sized instances, the underlying MILP remains computationally prohibitive for practical, city-sized networks due to the inherent complexity of routing and combinatorial conflict constraints. To tackle such computational challenges, the Kernel Search (KS) matheuristic framework (Angelelli et al., 2010) has proven highly effective across various domains, including finance (Angelelli et al., 2012), healthcare (Gobbi et al., 2019), and city logistics (Guastaroba and Speranza, 2012), and has been successfully integrated into hybrid methods (Gobbi et al., 2023; Mansini et al., 2023) and branch-and-cut algorithms (Hanafi

et al., 2020). For the large-scale instances of the G2T-RADP, we implement a specific variant known as Two-Phase Kernel Search (2P-KS), which was initially proposed for multidimensional knapsack problems (Lamanna et al., 2022) and facility location models (Mansini and Zanotti, 2022). In this framework, the service-quality metric is treated as a hard bound, and the 2P-KS algorithm efficiently minimizes the environmental objective by strategically focusing computational effort on the most promising subsets of variables.

In summary, this paper provides the following main contributions:

(i) We introduce the G2T-RADP, a novel bi-objective last-mile delivery problem that optimizes a fully electric truck-and-robot system to minimize environmental emissions and maximize customer satisfaction.

(ii) We incorporate the practical concept of facility incompatibilities, assessing how spatial and operational conflict constraints impact network design and routing solutions.

(iii) We provide a formal MILP formulation for the proposed problem.

(iv) Through a comprehensive experimental campaign on medium-sized instances, we employ a lexicographic approach to analyze the trade-offs between the two objectives, providing valuable managerial insights for collaborative, environmentally friendly logistics.

(v) We adapt a variant of the Kernel Search matheuristic, specifically the Two-Phase KS (2P-KS), designed to efficiently solve large-scale instances, benchmarking its performance against a state-of-the-art commercial exact solver.

(vi) We apply the proposed model to a real-world case study in the city of Brescia, Italy, validating the feasibility of this shared-infrastructure approach and providing actionable insights for achieving CO_2-neutral city logistics.

The remainder of the paper is organized as follows: Section 5.2 reviews the recent literature on last-mile delivery problems involving ADRs. Section 5.3 introduces the necessary notation and the mathematical formulation of the G2T-RADP. Section 5.4 details the KS-based solution approach. Section 5.5 discusses the results and managerial insights derived from medium and large-sized instances, while Section 5.5.4 analyzes the real-world case study of Brescia. Finally, Section 5.6 outlines conclusions and future research directions.

5.2 Related literature

This section reviews the streams of literature most relevant to the G2T-RADP. We focus on works addressing the environmental assessment of last-mile delivery systems, operational models based on drones and ADRs, and multi-objective approaches that jointly consider sustainability and service quality. Although several other last-mile delivery models exist (Boysen et al., 2021; Gajda et al., 2025; Hwang et al., 2024), our emphasis is on non-human-operated delivery systems and on the gap related to fully electric two-tier truck-and-robot networks operating under facility conflicts.

Concerning the environmental implications of last-mile delivery, contributions such as Bektaş and Laporte (2011) and Demir et al. (2014) study routing and emissions in truck-based Vehicle Routing Problems (VRPs) with conventional vehicles. Purely electric truck fleets have also been analyzed, as in Breunig et al. (2019), although without support from autonomous vehicles. Other studies examine the environmental impact of specific last-mile

configurations, including same-day delivery with narrow time windows (Manerba et al., 2018), the integration with shared mobility solutions (Qi et al., 2018), parcel lockers (Bell et al., 2023; Leung et al., 2023; Hovi and Bø, 2024), and ADRs (Chen and Demir, 2022). Policy-oriented contributions investigate mechanisms that can steer companies toward more sustainable delivery practices (Sánchez-Díaz et al., 2017; Mansini et al., 2024), while consumer-oriented studies demonstrate a willingness to pay for greener delivery options (Biancolin and Rotaris, 2024).

This stream of work also highlights that environmental gains can be overstated if system-wide effects are ignored. For example, parcel lockers may reduce courier travel while inducing customer pickup trips that offset part of the environmental benefit (Bonomi et al., 2022). Related studies also assess energy use, lifecycle costs, and indirect electricity-related emissions in robot-assisted delivery settings (Figliozzi, 2020; Lemardelé et al., 2023). Together, these works motivate the need for environmental models that go beyond distance-only proxies and account for broader system effects.

Many solutions in the literature use some form of assistance from non-human-operated vehicles. In the aerial domain, two main approaches are often considered: one involves drones starting and ending their delivery routes at a distribution center, while the other uses drones launched from a truck that visits customers and retrieves them at later stops (Murray and Chu, 2015; Ulmer and Thomas, 2018; Corberán et al., 2024; Zou et al., 2024). Although these systems can reduce delivery times by circumventing road congestion, they must cope with limited autonomy, payload constraints, and strict regulations regarding flights over urban areas.

ADRs represent a terrestrial alternative. Compared to drones, they operate at lower speeds and are more exposed to traffic and pedestrian interactions, but they can offer higher payload capacity and strong integration with urban logistics operations. When examining truck-and-robot systems, the literature distinguishes three main application models (Srinivas et al., 2022). The first configuration involves a truck acting as a mothership carrying multiple robots and dispatching them within city neighborhoods (Boysen et al., 2018; Simoni et al., 2020; Chen et al., 2021; Yu et al., 2022; Shen et al., 2024). The second framework uses trucks to transport parcels to local hubs, from which robots serve the final customers in a two-tier setting (Bakach et al., 2021; Alfandari et al., 2022). The third approach features road robots operating within AV-friendly zones and joining a lead-vehicle platoon to traverse non-AV areas (Scherr et al., 2019, 2020). Other publications add further operational features, such as delivery restrictions and time windows (Chen et al., 2021), or extend the setting to multiple trucks and multiple robots (Ostermeier et al., 2023).

In the context of multi-objective optimization, Alfandari et al. (2022) study a two-tier setting with a single vehicle, where the primary objective is to minimize tardiness indicators based on customer deadlines. The associated managerial analysis explores the impact of parameters such as the number of available facilities, robot coverage radius, and robot speed on service quality and environmental costs.

Taken together, the literature shows a rapid evolution of optimization models for autonomous last-mile delivery, with increasing attention devoted to truck–robot coordination, routing decisions, and operational efficiency. Nevertheless, existing studies generally focus on a limited subset of operational and sustainability aspects. Environmental impacts are often simplified or treated as secondary objectives, facility conflicts are rarely modeled explicitly, and relatively few works provide solution approaches capable of efficiently solving large-scale bi-objective instances.

Accordingly, this chapter fills the identified research gap by introducing the Green Two-

Tier Robot-Assisted Delivery Problem (G2T-RADP). To the best of our knowledge, this is the first optimization model to jointly consider a fully electric two-tier truck–robot delivery system, explicit facility conflicts, detailed operational emissions—including indirect emissions from electricity generation—and a bi-objective formulation balancing environmental and operational performance. To tackle the resulting computational complexity, the chapter develops a dedicated Two-Phase Kernel Search matheuristic that efficiently solves large-scale instances. Table 5.1 summarizes the positioning of this contribution relative to the main relevant literature discussed above.

Table 5.1: Comparison of optimization-based last-mile delivery models.

Work	Electric Fleet	Drone/ADR	Two-Tier	Multi-Objective	Detailed Emissions	Indirect Emissions	Facility Conflicts
Bektaş and Laporte (2011)	○	○	○	○	◐	○	○
Demir et al. (2014)	○	○	○	○	◐	○	○
Murray and Chu (2015); Ulmer and Thomas (2018); Corberán et al. (2024); Zou et al. (2024)	○	●	◐	○	○	○	○
Manerba et al. (2018)	○	○	○	◐	○	○	○
Qi et al. (2018)	◐	○	○	◐	○	○	○
Boysen et al. (2018); Simoni et al. (2020); Chen et al. (2021); Yu et al. (2022); Shen et al. (2024)	○	●	◐	○	○	○	○
Breunig et al. (2019)	●	○	○	○	◐	○	○
Scherr et al. (2019, 2020)	○	●	◐	○	○	○	○
Figliozzi (2020); Lemardelé et al. (2023)	●	●	○	○	●	●	○
Bakach et al. (2021); Alfandari et al. (2022)	◐	●	●	◐	◐	○	○
Chen and Demir (2022)	◐	●	○	○	◐	◐	○
Bell et al. (2023); Leung et al. (2023); Hovi and Bø (2024)	○	○	◐	○	◐	◐	○
Ostermeier et al. (2023)	○	●	●	○	○	○	○
This work	●	●	●	●	●	●	●

Notation: ● = fully addressed; ◐ = partially addressed; ○ = not addressed.

5.3 Problem Definition and Mathematical Formulation

We consider a set C of customers who require deliveries. Each customer $c \in C$ has a soft deadline u_c by which they prefer to receive their package. Each package has a weight $q_c, c \in C$. The delivery process involves two steps. First, a truck with capacity Q, assumed to be sufficiently large to accommodate the total daily demand ($\sum_{c \in C} q_c$), departs from the central depot to deliver packages to a subset of intermediate facilities and then returns to the depot within a predefined working time $t_{\max}$. We indicate as K the set of all the facilities. Each facility $k \in K$, has r_k homogeneous ADRs for delivering packages to customers. Once a facility receives packages, it loads them onto the robots. This process takes a constant time τ, regardless of the number of packages or their weights. Each robot carries a single package to a customer and operates within a finite battery capacity. We assume each robot performs at most one delivery per facility visit; therefore, the number of robots available r_k in a facility limits the number of customers that can be assigned to that facility k to at most r_k. This defines the subset of customers $C_k \subseteq C, k \in K$ that each facility can serve based on the robots' autonomy, as well as the set of facilities $K_c \subseteq K$ that can serve each customer $c \in C$. We also define Θ as the set of unordered pairs $\langle i, j \rangle, i, j \in K, i \neq j$ of facilities that are mutually incompatible and cannot be concurrently activated.

The G2T-RADP can be defined over a complete graph $G = (V, A)$, where $V = K \cup \{0\}$ is the node set, with node 0 representing the depot, and $A = \{(i, j) | i \in V, j \in V, i \neq j\}$ is the arc set. For each arc $(i, j) \in A$, d_{ij} and t_{ij} represent the distance and the travel time required by the delivery truck to travel from node i to node j, respectively. Both distances and travel times satisfy the triangular inequality. Additionally, we indicate as $h_{kc}, k \in K, c \in C_k$ the travel time for a robot to reach customer c from facility k. To standardize the environmental objective function, we introduce three conversion factors that map distinct operational metrics into a unified measurement unit. In particular, α, β, and γ convert weight transported by a truck over an arc, distance traveled by the vehicle, and duration of the robots'

trips into gCO_2e (grams of CO_2 equivalent), respectively. CO_2e is a standard metric used to compare the emissions from various greenhouse gases according to their global warming potential. From now on, for any subset $S \subset V$, let $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A : i \notin S, j \in S\}$ be the sets of arcs leaving from and entering set S, respectively. To simplify notation, the special case in which $|S| = 1$ is denoted as $\delta^+(i)$ ($\delta^-(i)$) instead of $\delta^+(\{i\})$ ($\delta^-(\{i\})$).

To provide a mathematical formulation of the problem, we introduce the following decision variables.

First, we define binary variables x_{ij} for all pairs of nodes $i, j \in V$, with $i \neq j$. Variable x_{ij} takes value 1 if the vehicle travels directly from node i to node j, and 0 otherwise. These variables describe the routing decisions of the delivery truck. For the same pairs of nodes, we introduce continuous variables w_{ij} representing the quantity of load transported by the vehicle while moving from node i to node j. These variables track how the total load evolves along the route. Next, for each customer $c \in C$, we define a binary variable s_c. This variable takes value 1 if customer c is served after its preferred delivery time u_c, and 0 otherwise. Therefore, s_c captures whether the service to customer c violates its soft deadline. To model the assignment of customers to facilities, we introduce binary variables z_{kc} for all facilities $k \in K$ and customers $c \in C$. Variable z_{kc} is equal to 1 if customer c is assigned to facility k, and 0 otherwise. These variables determine from which facility each customer is served. Finally, to ensure consistency between routing and assignment decisions, we introduce a set of non-negative continuous variables f_{ij}^c for all $i, j \in V$, with $i \neq j$, and for all customers $c \in C$. For each customer c, the variables f_{ij}^c represent a unit flow that originates from the depot and is directed toward the facility selected to serve that customer. These variables are used to enforce connectivity between the depot and the selected facility for each customer, ensuring that assignments are supported by feasible vehicle movements.

A summary of the sets, indices, parameters, and decision variables of the model is provided in Tables 5.2 to 5.5.

Table 5.2: Sets

C	Set of customers
K	Set of facilities
$V = K \cup \{0\}$	Set of nodes (facilities and depot)
A	Set of directed arcs (i, j) with $i \neq j$
$C_k \subseteq C$	Customers that can be served by facility k
$K_c \subseteq K$	Facilities that can serve customer c
Θ	Set of incompatible facility pairs $\langle k_1, k_2 \rangle$

Table 5.3: Indices

i, j	Nodes in V
k	Facility index
c	Customer index

Table 5.4: Parameters

q_c	Demand of customer c
u_c	Soft deadline of customer c
Q	Truck capacity
r_k	Number of robots available at facility k
τ	Loading time at facility
$t_{\max}$	Maximum route duration for the truck
d_{ij}	Distance between nodes i and j
t_{ij}	Travel time between nodes i and j
h_{kc}	Travel time of a robot from facility k to customer c
α	Emission factor per unit weight-distance
β	Emission factor per unit distance traveled
γ	Emission factor per unit robot travel time
M_c	Large constant for deadline constraint

Table 5.5: Decision variables summary

x_{ij}	Binary variable equal to 1 if the truck traverses arc $(i,j) \in A$, 0 otherwise
w_{ij}	Continuous variable representing the load (weight) carried on arc $(i,j) \in A$
z_{kc}	Binary variable equal to 1 if customer $c \in C$ is served from facility $k \in K$, 0 otherwise
s_c	Binary variable equal to 1 if customer $c \in C$ is served after its deadline u_c, 0 otherwise
f_{ij}^c	Continuous flow variable representing the unit flow of commodity $c \in C$ on arc $(i,j) \in A$

Then, the G2T-RADP problem can be defined as follows:

$$f_1 = \min \sum_{(i,j)\in A} (\alpha d_{ij} w_{ij} + \beta d_{ij} x_{ij}) + \sum_{c\in C} \sum_{k\in K_c} 2\gamma h_{kc} z_{kc} \tag{5.1}$$

$$f_2 = \min \sum_{c\in C} s_c \tag{5.2}$$

$$\sum_{(i,j)\in\delta^-(k)} x_{ij} = \sum_{(i,j)\in\delta^+(k)} x_{ij} \leq 1 \qquad k \in K \tag{5.3}$$

$$\sum_{(i,j)\in\delta^-(0)} x_{ij} = \sum_{(i,j)\in\delta^+(0)} x_{ij} = 1 \tag{5.4}$$

$$\sum_{(i,j)\in\delta^-(k)} w_{ij} - \sum_{(i,j)\in\delta^+(k)} w_{ij} = \sum_{c\in C_k} q_c z_{kc} \qquad k \in K \tag{5.5}$$

$$w_{ij} \leq Q x_{ij} \qquad (i,j) \in A \tag{5.6}$$

$$\sum_{(i,j)\in\delta^-(0)} w_{ij} = 0 \tag{5.7}$$

$$z_{kc} \leq \sum_{(i,j)\in\delta^-(k)} x_{ij} \qquad c \in C, k \in K_c \tag{5.8}$$

$$\sum_{k\in K_c} z_{kc} = 1 \qquad c \in C \tag{5.9}$$

$$\sum_{c\in C_k} z_{kc} \leq r_k \qquad k \in K \tag{5.10}$$

$$\sum_{(i,j)\in\delta^-(k)} f_{ij}^c - \sum_{(i,j)\in\delta^+(k)} f_{ij}^c = \begin{cases} -1 & \text{if } k = 0 \\ z_{kc} & \text{if } k \in K_c \\ 0 & \text{otherwise} \end{cases} \qquad k \in V,\ c \in C \tag{5.11}$$

$$f_{ij}^c \leq x_{ij} \qquad (i,j) \in A, c \in C \tag{5.12}$$

$$\sum_{(i,j)\in A} (t_{ij} + \tau) f_{ij}^c + \sum_{k\in K_c} h_{kc} z_{kc} \leq u_c + M_c s_c \qquad c \in C \tag{5.13}$$

$$\sum_{i\in V, j\in K, i\neq j} \tau x_{ij} + \sum_{(i,j)\in A} t_{ij} x_{ij} \leq t_{\max} \tag{5.14}$$

$$\sum_{(i,j)\in\delta^-(k_1)} x_{ij} + \sum_{(i,j)\in\delta^-(k_2)} x_{ij} \leq 1 \qquad \langle k_1, k_2 \rangle \in \Theta \tag{5.15}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{5.16}$$

$$z_{kc} \in \{0,1\} \qquad c \in C, k \in K_c \tag{5.17}$$

$$f_{ij}^{c} \geq 0 \qquad (i,j) \in A, c \in C \tag{5.18}$$

$$w_{ij} \geq 0 \qquad (i,j) \in A \tag{5.19}$$

$$s_c \in \{0,1\} \qquad c \in C \tag{5.20}$$

The objective function (5.1) addresses environmental impact by considering the weight carried by the vehicle on each arc, the distance traveled by the vehicle, and the time spent by robots reaching customers and returning to their starting facility. The objective function (5.2) focuses on service quality, minimizing the number of customers served without satisfying their soft deadline (late customers). Constraints (5.3) guarantee that each facility k is visited at most once, whereas constraint (5.4) imposes that exactly one arc leaves and enters the depot. Constraints (5.5)-(5.7) manage the vehicle's load. In particular, constraints (5.5) ensure that the vehicle's load decreases by the sum of the weights $\sum_{c \in C_k} q_c z_{kc}$ of the packages delivered at each served facility. Constraints (5.6) link variable w and x, guaranteeing that no load can be carried on a non-traversed arc. Constraint (5.7) mandates that the vehicle returns empty to the depot. This constraint is not required when optimizing (5.1) but is needed for consistent results with (5.2). Constraints (5.8) ensure that a facility can serve a customer only if the vehicle visits that facility. Constraints (5.9) require each customer to be served by exactly one facility, while constraints (5.10) limit the number of customers a facility can serve to its capacity. Constraints (5.11) are flow conservation constraints implying that for each customer $c \in C$, a flow equal to 1 starts in the depot and is absorbed by the facility selected to serve the customer, guaranteeing flow conservation in all the other nodes, while constraints (5.12) prevent positive flow on unused arcs. Constraints (5.13) determine the arrival time at each customer $c \in C$ and ensure that s_c takes value 1 when customer c is visited past the deadline u_c. Here, M_c is a customer-specific big-M and can be defined, for example, as $M_c = t_{\max} - \tau + \max_{k \in K_c} h_{kc} - u_c$, which provides a valid upper bound on the maximum time the arrival at customer c can exceed their deadline u_c. Constraint (5.14) ensures that the vehicle returns to the depot within $t_{\max}$, whereas constraints (5.15) prevent visiting two conflicting facilities within the same solution. Finally, constraints (5.16)–(5.20) impose binary and non-negative conditions on the problem variables.

5.3.1 Modeling Assumptions

The mathematical model presented in this chapter relies on a set of assumptions that balance operational realism with computational tractability. Although last-mile delivery systems are inherently subject to uncertainty, several operational aspects are represented through deterministic approximations in order to focus on the interaction between truck–robot coordination, environmental sustainability, and delivery performance. The principal assumptions adopted by the model are summarized below together with their motivation.

Deterministic customer demand. Customer locations, parcel demands, and delivery time windows are assumed to be known before the optimization process and remain unchanged throughout the planning horizon. This assumption reflects the offline planning setting considered in this work, where delivery requests are available prior to route generation. Dynamic customer requests and cancellations are not considered.

Deterministic travel and service times. Travel times for trucks and robots, together with customer service times, are assumed to be known and constant. In practice, these

quantities may vary due to traffic conditions, weather, pedestrian interactions, or operational delays. Such variability is neglected in order to focus on the routing and coordination decisions addressed by the proposed optimization framework.

Deterministic vehicle and robot performance. The energy consumption, travelling speed, and operational characteristics of both trucks and autonomous robots are assumed to be known and remain constant throughout the planning horizon. Although battery performance and vehicle efficiency and speed may vary with environmental and operational conditions, these effects are represented through deterministic parameters to maintain model tractability while preserving the main operational characteristics of the delivery system.

Deterministic facility availability. Candidate parking facilities are assumed to remain available throughout the planning horizon, and facility conflicts are represented through predefined deterministic constraints. Temporary parking restrictions, unexpected obstructions, and other operational disruptions are outside the scope of this work.

These assumptions are consistent with the scope of the proposed model, which focuses on evaluating the trade-offs between operational efficiency and environmental sustainability in truck–robot delivery systems. While uncertainty associated with traffic conditions, customer requests, battery performance, and operational disruptions is not explicitly modeled, the proposed formulation captures the principal operational decisions required for strategic delivery planning. Future research may extend this framework by incorporating stochastic or robust optimization techniques, as well as dynamic and real-time decision-making approaches capable of adapting routes to evolving operating conditions.

5.4 Solution Method

This section outlines the framework used to evaluate the bi-objective nature of the problem and introduces the matheuristic method implemented to solve large-scale instances.

5.4.1 Bi-Objective approach

The bi-objective nature of the G2T-RADP is handled through the two-step lexicographic optimization approach introduced in Section 2.5, where one objective f_P is first optimized, and the non-prioritized objective f_{NP} is subsequently minimized under the constraint $f_P \leq f_P^*(1+w)$, with $w \in [0,1]$ controlling the maximum allowed degradation. We refer to this procedure as $(f_{NP}|f_P, w)$.

In the context of the G2T-RADP, either f_1 or f_2 can be prioritized, allowing us to either minimize the emission footprint first and subsequently minimize the number of late deliveries, or reverse the order entirely. By systematically adjusting w, we can evaluate the trade-offs between the two objectives.

When evaluating large-scale instances, calculating the entire Pareto-optimal frontier becomes computationally prohibitive. Because our primary analytical goal in these scenarios is to establish the minimum environmental footprint achievable under optimal service conditions, we streamline the evaluation. Instead of mapping the full frontier, we strictly prioritize service quality and optimally solve f_2. Subsequently, we optimize the environmental objective f_1 without allowing any deterioration in service quality ($w = 0$), summarized as $(f_1|f_2, 0)$.

5.4.2 Two-Phase Kernel Search

To solve the large-scale instances of the G2T-RADP, we develop an approach based on a variant of the Kernel Search (KS) matheuristic framework. Specifically, we implement the Two-Phase KS (2P-KS), which was originally proposed by Lamanna et al. (2022) for the multi-dimensional multiple-choice knapsack problem and later extended to facility location models (Mansini and Zanotti, 2022).

Fundamentally, the standard KS framework operates by iteratively solving a sequence of restricted problems using a commercial MILP solver. The method exploits the information provided by the optimal solution of the continuous LP relaxation of the problem. By analyzing the values of the continuous variables and their corresponding reduced costs, the algorithm identifies the most promising variables, specifically those most likely to take non-zero values in the optimal integer solution. These variables form the initial *kernel set*, while the remaining variables are partitioned into disjoint *buckets*. Each restricted problem is constructed by activating the variables in the current kernel set alongside the variables of one specific bucket, while all excluded variables are fixed to zero. The optimal solution of each restricted problem yields a new incumbent solution and dictates the update of the kernel set, which permanently absorbs any variables from the current bucket that take non-zero values.

For the G2T-RADP, the application of a Kernel Search heuristic, or one of its variant is motivated by the hypothesis that the problem's underlying structure, particularly its LP relaxation, can be exploited to significantly reduce computational effort by focusing on a limited subset of variables. We opted for the two-phase variant, because classical KS can converge slowly when initial feasibility is difficult to achieve.

This variant, is called the Two-Phase KS (2P-KS), and breaks the solution process into two distinct phases. In each phase, a complete classical KS is executed by solving a sequence of restricted problems. The termination criterion for each phase is a defined time limit, with each individual restricted problem being solved either to optimality or until it reaches a specific subproblem time limit. This implies that if, any phase of the method, the time limit is not reached upon analyzing the last bucket, the bucket size is adjusted, and buckets are scrolled again (iterative variant).

Splitting the procedure, can be helpful if the two phases focus on different core aspect of the solution. In particular one phase can focus over the feasibility (first phase), and the other one can focus just on improving the found solution (second phase), yielding near-optimal solutions with greater computational efficiency than exact solvers within a restricted time budget.

Each phase uses different parameters in order to change the two distinct objectives. The first phase that emphasizes *feasibility* addresses sub-problems within a very short time limit, primarily seeking any feasible solution rather than a high-quality one. During this process, KS gathers data from the solved restricted problems to assess the difficulty of the integer problem and classify it. This phase, also adjusts the KS parameters for the second phase to improve its performance. The second phase focuses soley on *solution quality* by aiming to enhance the solution value obtained in the first step. Giving more time to each subproblem, or varying other KS parameters, in order to intensify the search method.

The pseudocode of 2P-KS is presented in Algorithm 5.1. 2P-KS receives as input an instance $\mathcal{I}$ of the G2T-RADP, a time limit T_{lim}, and the main settings of the two phases. In particular, Phase 1 is associated with a tuple of parameters (b_1, Δ_1, t_1), representing the bucket size, the percentage increase of the bucket after each bucket iteration, and the time limit used for resolving each subproblem, respectively. The bucket size corresponds to a multiplicative factor of the kernel set size, allowing it to vary across instances. Depending on the results

Algorithm 5.1 2P-KS($\mathcal{I}, (b_1, \Delta_1, t_1), (b_2, \Delta_2, t_2), (b_3, \Delta_3, t_3), \omega, TH, T_{lim}$)

1: $n_{tl}, n_p \leftarrow 0$
2: $phase_1 \leftarrow true$
3: $T^1_{lim} \leftarrow \omega \cdot T_{lim}$
4: $(b, \Delta, t) \leftarrow (b_1, \Delta_1, t_1)$
5: $(\mathbf{y}^{LP}, \mathbf{rc}^{LP}, obj^{LP}) \leftarrow$ SolveLPRelaxation($\mathcal{I}$) ▷ $\mathbf{y} = (\mathbf{x}, \mathbf{z}, \mathbf{s}, \mathbf{f}, \mathbf{w})$ ▷ $\mathbf{rc} = (\mathbf{rc_x}, \mathbf{rc_z}, \mathbf{rc_s}, \mathbf{rc_f}, \mathbf{rc_w})$
6: $\Lambda, \bar{\Lambda} \leftarrow$ BuildKernelSet($\mathcal{I}, \mathbf{x}^{LP}, \mathbf{rc_x}^{LP}$)
7: $(\mathbf{y}^*, obj^*, \bar{S}, sol^*, t_{elapsed}, status) \leftarrow$ Solve($\mathcal{I}, \bar{\Lambda}, \emptyset, t$)
8: **if** $status = timeLimit$ **then**
9: $n_{tl} \leftarrow n_{tl} + 1$
10: **end if**
11: $n_p \leftarrow n_p + 1$
12: $\bar{\lambda} \leftarrow |\Lambda|$
13: **while** $t_{elapsed} < T_{lim}$ **do**
14: $\mathcal{B} \leftarrow$ BuildBuckets($N \setminus \bar{\Lambda}, b, \bar{\lambda}$)
15: **for all** $B \in \mathcal{B}$ **do**
16: $t \leftarrow min(t, T_{lim} - t_{elapsed})$
17: $(\mathbf{y}^{ILP}, obj^{ILP}, \bar{S}, sol^{ILP}, t_{used}, status) \leftarrow$ Solve($\mathcal{I}, \bar{\Lambda} \cup B, sol^*, t$)
18: $t_{elapsed} \leftarrow t_{elapsed} + t_{used}$
19: **if** $status = timeLimit$ **then**
20: $n_{tl} \leftarrow n_{tl} + 1$
21: **end if**
22: $n_p \leftarrow n_p + 1$
23: **if** $\mathbf{y}^{ILP} \neq NULL$ **and** $\mathbf{y}^{ILP} \neq \mathbf{y}^*$ **then**
24: $(\mathbf{y}^*, obj^*, sol^*) \leftarrow (\mathbf{y}^{ILP}, obj^{ILP}, sol^{ILP})$
25: $\bar{\Lambda} \leftarrow \bar{\Lambda} \cup \bar{S}$
26: **end if**
27: **end for**
28: $b \leftarrow b + \Delta$
29: **if** $t_{elapsed} \geq T^1_{lim}$ **and** $phase_1 = true$ **then**
30: **if** $n_{tl}/n_p \geq TH$ **then**
31: $(b, \Delta, t) \leftarrow (b_3, \Delta_3, t_3)$
32: **else**
33: $(b, \Delta, t) \leftarrow (b_2, \Delta_2, t_2)$
34: **end if**
35: $phase_1 \leftarrow false$
36: $t \leftarrow min(t, T_{lim} - t_{elapsed})$
37: $(\mathbf{y}^{ILP}, obj^{ILP}, \bar{S}, sol^{ILP}, t_{used}, status) \leftarrow$ Solve($\mathcal{I}, \bar{\Lambda}, sol^*, t$)
38: **if** $\mathbf{y}^{ILP} \neq NULL$ **and** $\mathbf{y}^{ILP} \neq \mathbf{y}^*$ **then**
39: $(\mathbf{y}^*, obj^*, sol^*) \leftarrow (\mathbf{y}^{ILP}, obj^{ILP}, sol^{ILP})$
40: **end if**
41: $t_{elapsed} \leftarrow t_{elapsed} + t_{used}$
42: $\bar{\lambda} \leftarrow |\bar{\Lambda}|$
43: **end if**
44: **end while**
45: **return** (sol^*, obj^*)

of Phase 1, the second phase will use parameters for either *fast* (easy) instances (b_2, Δ_2, t_2) or *slow* (hard) ones (b_3, Δ_3, t_3). This classification is based on whether the percentage of sub-problems that reached the time limit in Phase 1 was below or above a specified threshold TH. The final input parameter is ω, which determines which percentage of T_{lim} is assigned to Phase 1. The algorithm first initializes the main variables and parameters (Steps 1-4), such as the counter for the number of restricted problems and the number of those that reach the time limit (t_p and t_{tl}) and the settings of Phase 1. Then, the Linear Programming (LP) relaxation is solved using function SolveLPRelaxation (Step 5), which returns the value of all the variables in the LP optimal solution (we indicate as $\mathbf{y}^{LP} = (\mathbf{x}^{LP}, \mathbf{w}^{LP}, \mathbf{f}^{LP}, \mathbf{z}^{LP}, \mathbf{s}^{LP})$) and the corresponding values of the reduced costs ($\mathbf{rc}^{LP}$). This information is used to define the ordering of the variables. For the problem under consideration, it was decided to focus solely on the x's variables, proceeding with their ordering based on the values they assume in the relaxed solution ($\mathbf{x}^{LP}$) and their reduced cost coefficients ($\mathbf{rc}^{LP}_x$) in the construction of the Kernel Set and buckets (BuildKernelSet).We also investigated the usage of a combination

of others variables, in conjunction with the x's but no real impact, was observed. Thus, the formulation of each resulting restricted problem consider only a subset of these x's variables, along with all the remaining variables of the original problem (ws, fs, zs, ss variables) associated with them (*active variables*). All other variables (non active) are set to zero. Using the most promising x's variables, the kernel set Λ is built (Step 6), and the first restricted problem considering such a set is solved (Step 7). The SOLVE function requires as input an instance, the subset of active variables, a starting solution (if any), and a time limit. In the first resolution, no starting solution is available. The function returns the value of the $\mathbf{y}$ variables ($\mathbf{y}^*$), the objective function value (obj^*), the set of variables that are selected ($\bar{S}$) in the obtained solution (if a solution is available, the empty set otherwise), the time elapsed to solve the restricted problem ($t_{elapsed}$), and the final status. This status could be *timeLimit* if the time limit is reached when solving the restricted problem, *optimal* if the current restricted problem has been solved to optimality producing a solution, or *infeasible* if the restricted problem has been proven to be infeasible. The counters n_{tl} and n_p are updated according to the result of the optimization (Steps 8-11). Next, the main 2P-KS loop begins (Steps 13-44), and it will terminate only when the global time limit is reached. First, the buckets are built using function BUILDBUCKETS (Step 14). This function takes as input the set of variables not included in the kernel set, the size of the original kernel set $\bar{\Lambda}$, and the coefficient b that determines the size of each bucket, which is $b \cdot \bar{\Lambda}$. It produces a collection of equal-sized buckets of variables (except for potentially the last one). As for the Kernel Set, variables x are used to define the buckets and the variables making part of it. A restricted problem concerning each bucket is tackled (Steps 15-27). The function SOLVE is called to solve a restricted problem in which only the variables in the kernel set and the current bucket are considered (Step 17) Notice that the function receives, in this case, a solution from which the optimization will start. Solution sol^* might still not be available at this point if the restricted problem built on the kernel set did not produce a feasible solution. A *NULL* value will then be passed to the function that will begin the optimization without a starting point. The time elapsed and the n_{tl} and n_p counters are updated as before (Steps 18-22). If an improving solution is found, the value of the list of variables y in the best solution $\mathbf{y}^*$, its value obj^*, and the best solution itself sol^* are updated, as well as the kernel set, whose size increases due to the inclusion of the bucket variables that have been selected in the new best solution (Steps 23-26). At Steps 29-43, the swhich between Phase 1 and Phase 2 happens when enough time has passed. Depending on the percentage of subproblems that reached the time limit, the tuple (b, Λ, t) is updated to the slow parameters (Step 31) or the fast ones (Step 33). A subproblem considering only the kernel set variables is solved (Step 37) to identify the best possible solution in the kernel set. If a new improving solution is found, the relevant variables are updated (Step 39). If the elapsed time is below T^1_{lim}, no swhich is performed, and KS considers the buckets again. When T_{lim} is reached, KS exits, producing the best solution and its value.

5.5 Experimental Analysis

This section is devoted to the experiments conducted to derive managerial insights and evaluate the performance of our 2P-KS implementation. In particular, Section 5.5.1 outlines the generation of benchmark instances, whereas Section 5.5.2 explores trade-offs and the relationship between service quality and environmentally focused policies. Section 5.5.3 assesses the effectiveness of 2P-KS in solving large instances of the G2T-RADP Finally, Section 5.5.4 analyses the possible application of the problem to the city of Brescia (Italy) for grocery delivery.

Both the 2P-KS algorithm and the mathematical model were implemented in Java 21. All experiments were performed on an Ubuntu 22.04 virtual machine with a Ryzen 9 3950X 16-core processor and 24 GB of RAM, using Gurobi 10.0.1 as LP and MILP solver. All available threads were used during testing.

5.5.1 Benchmark instances

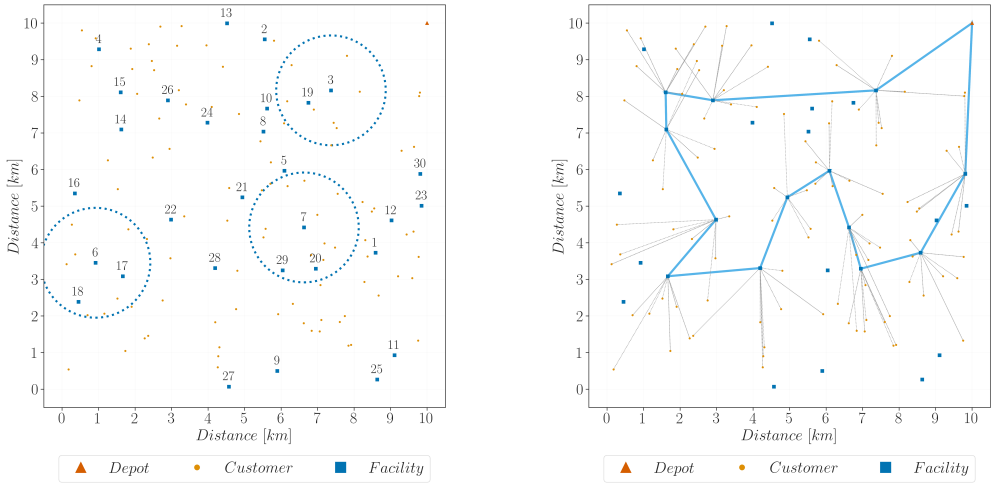
To evaluate the performance of our proposed solution approach for the G2T-RADP and to gather meaningful managerial insights, we conduct experiments on instances that realistically represent operational scenarios. We generate two distinct sets of instances, Set A and Set B, characterized by varying numbers of customers and facilities. The two sets serve different purposes: Set A is used to perform a managerial analysis, while Set B is designed to be computationally challenging and serves as a benchmark to evaluate 2P-KS. The main difference between the two sets consists of the number of customers ($|C| = 40$ for Set A, $|C| = 100$ for Set B) and the number of facilities ($|K| = 10$ for Set A, $|K| = 30$ for Set B). Facilities and customers are randomly positioned within a 5×5 km grid for Set A and a 10×10 km grid for Set B, while the depot is randomly placed in one of the corners of the grid. Facilities' positions are generated ensuring a minimum distance between each pair (200m for Set A, 300m for Set B). Customers' locations are generated ensuring that, for all $c \in C$, $|K_c| \geq k_{min}$, with $k_{min} \in \{1, 3\}$ for Set A and $k_{min} = 3$ for Set B.

Parameters α, β, and γ are computed considering the Ford E-Transit (420 PTT - L2H2) and the Starship ADR. We used the energy consumption parameters presented in (Ford, 2024) for the vehicle and (Figliozzi, 2020) for the robot. We assumed that energy consumption increases linearly with load, with the base coefficient computed as energy consumption divided by the vehicle tare weight. With this assumption, we can transform all the quantities in our objective function into kWh. To convert all kWh values into gCO_2e, we identify a coefficient to estimate the greenhouse gas emissions generated to produce a single kWh of energy. We used the coefficients provided by the 2023 report of the European Environment Agency (European Environment Agency, 2024). In particular, we considered the average value of gCO_2e per kWh of the entire European Union.

Parcels' weights $q_c, c \in C$ are drawn from a uniform distribution with a mean of 10 and a standard deviation of 2. The number of robots available at each facility $r_k, k \in K$, is an integer value randomly generated in the interval $[6, 9]$. The due dates for each customer are defined following the same approach described in (Alfandari et al., 2022), that defines $u_k = t^*_G(0, k) \times (\rho_{\min} + \delta_k(\rho_{\max} - \rho_{\min}))$, where $t^*_G(0, k)$ is the shortest time from the depot to facility k, δ_k is a random number drawn uniformly from $(0, 1]$, and $\rho_{\min} = 1$. In our case, Set A instances have $\rho_{\max} \in \{3, 5\}$, while Set B instances are generated using $\rho_{\max} = 5$. The maximum time allowed for the truck trip $t_{\max}$ is $8h$. Vehicle speed is set at 30 km/h, while robots' speed is set at 6 km/h, per the robot manufacturer's technical information. The time to load robots at a facility τ is set at 0.25 hours (15 minutes).

Conflicts are considered in both Sets. In this case, set Θ is generated assuming all facilities within distance θ are in conflict (*conflict area*). We consider $\theta \in \{1km, 1.5km\}$. When referencing instances from Set B that consider $\theta = 1km$ or $\theta = 1.5km$ as the threshold to determine conflicts, we will use the name $B1$, $A1$ and $B1.5$,$A1.5$ respectively. Figure 5.1a shows an example of an instance in Set B1.5. In particular, we present for facilities 3, 6, and 7 their conflict areas. For instance, facilities 17 and 18 conflict with facility 6, as they fall within its conflict area. In Figure 5.1b, we provide the optimal solution obtained for this instance. The route of the truck is reported in blue, while the association between customers and facilities

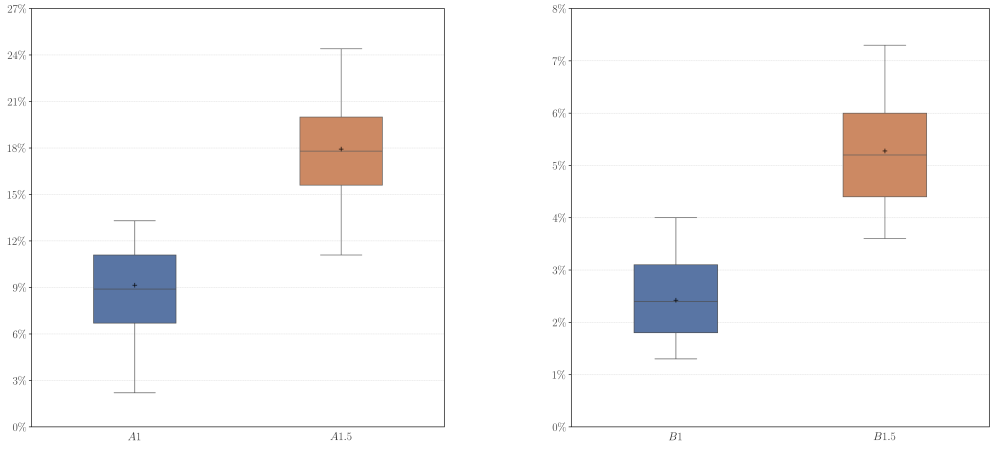
is drawn in grey. Finally, Figure 5.2 shows the density of conflict in all the analyzed instances. Figure 5.2b shows Set B, when $\theta = 1$ it has a median around 2.5% and when $\theta = 1.5$ the median is about twice. Figure 5.2a shows Set A, its density of conflicts preserve the relation that $A1.5$ has double density of conflicts with respect to $A1$, but given the smaller size of the investigated area, the base values of conflicts are around 9% much higher than the one in Set B.



(a) Conflict areas for facilities 3, 6, and 7.

(b) Example of a solution.

Figure 5.1: Example of a Set B1.5 instance.



(a) Set A

(b) Set B

Figure 5.2: Density of conflicts in the analyzed sets, comparison between 1 km ($A1$, $B1$) and 1.5 km ($A1.5$, $B1.5$).

In total, we considered 480 instances in Set A (40 for each parameter combination) and

Parameter	Value
$T_{\max}$	8 h
r_k	$[6, 9]$
q_c	$\mathcal{N}(10, 2)$
τ	0.25 h
α	0.027 $\frac{gCO_2e}{kg\,km}$
β	66.515 $\frac{gCO_2e}{km}$
γ	37.198 $\frac{gCO_2e}{h}$

Table 5.6: Common instance parameters.

Set	Grid	|K|	|C|	min d_{ij}	k_{min}	ρ_{max}	θ	# Inst
Set A	5×5	10	40	200	1,3	3,5	0, 1, 1.5	40
Set B	10×10	30	100	300	3	5	1, 1.5	30

Table 5.7: Set specific parameters.

60 in Set B (30 for each parameter combination). Table 5.6 reports the values of the common parameters used in all the instances. Table 5.7 summarizes the main characteristics of the generated instances.

The benchmark instances are synthetically generated rather than derived from historical logistics operations. However, the generation procedure is calibrated using realistic geographical layouts, vehicle and robot specifications, energy consumption models, and operational parameters reported in the literature and by manufacturers. Consequently, the generated benchmark instances are intended to represent plausible urban delivery scenarios while enabling a systematic evaluation of the proposed optimization framework under controlled experimental conditions.

5.5.2 Managerial Insights and Sensitivity Analysis

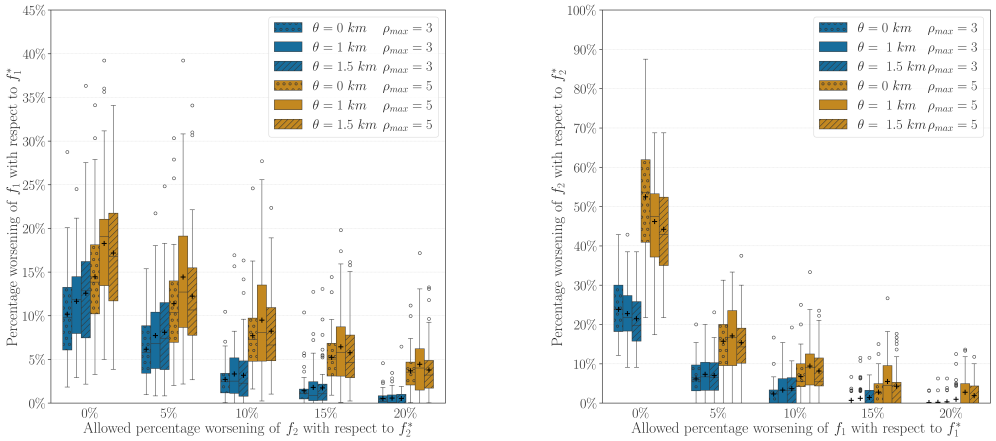
In this section, we evaluate potential strategies for a last-mile delivery company that wants to consider both quality of service (QoS) and environmental impact, in terms of gCO2e emissions, when deciding an operational plan for daily deliveries. In our case, the objective function (5.2) serves as a measure for QoS, while objective function (5.1) is the environmental-oriented goal.

In this case, we assume that the primary focus is the reduction in transportation-related emission, while traditional last-mile delivery costs are not at the core of the optimization analysis. Some of the classical metrics will be briefly discussed as just as a result of the environmental component. All optimizations performed on Set A instances reached optimality.

In Figure 5.3a, we analyze the percentage worsening of f_1 with respect to f_1^* (corresponding to optimally solving the model with the objective function (5.1) and no restrictions on function (5.2)) for different allowed percentage worsening of f_2 with respect to f_2^*. In Figure 5.3b, the roles of the two functions are reversed (in this case, f_2^* corresponds to optimally solving the model with the objective function (5.2) and no restrictions on function (5.1)).

The analysis focuses on both the impact of the two values of ρ_{max} and the impact of the three values of θ. Having lower values for the parameter ρ_{max} ($\rho_{max} = 3$) implies stricter due dates and thus a higher number of customers not served within the due dates. In a symmetrical way, higher values of θ, induce a restriction in the number of available facility, that should increase the distance traveled.

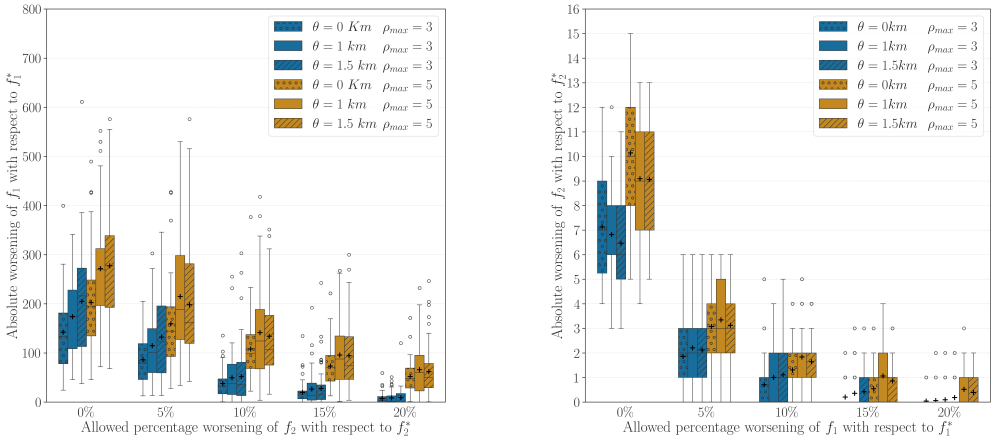
This explains the performance shown in Figure 5.3a where the percentage worsening of f_1 with respect to f_1^* are lower when $\rho_{max} = 3$. When optimizing f_1 with a restriction on f_2, if fewer customers can be served in time, the problem is less constrained, and a better



(a) Worsening of f_1 with f_2 constrained

(b) Worsening of f_2 with f_1 constrained

Figure 5.3: Worsening of one objective function while allowing different percentage worsening of the other



(a) Worsening of f_1 with f_2 constrained

(b) Worsening of f_2 with f_1 constrained

Figure 5.4: Worsening of one objective function, in absolute value, while allowing different percentage worsening of the other

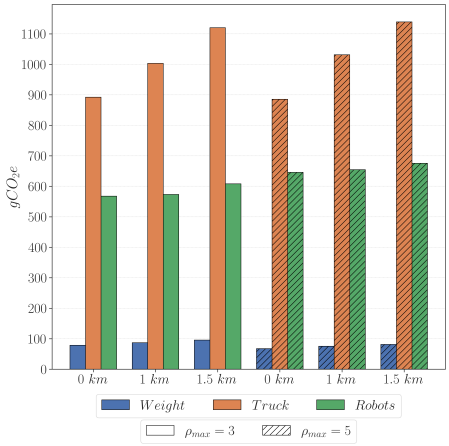
objective function value for f_1 can be obtained. Notice that, for $\rho_{max} = 3$, decreasing the QoS by just 15% produces an environmental cost that is very close to the optimal value f_1^*.

Similar to what has been said regarding ρ_{max} in Figure 5.3a, increasing the value of θ will create worse results with respect to the optimal environmental metric. It is interesting to note that the conflict radius of 1.5 km, in the case of $\rho_{max} = 5$, does not follow this rule; rather, it improves the difference from the optimal solution. This suggest that increasing the conflict radius can reduce the impact of the due-date to final quality of the solution.

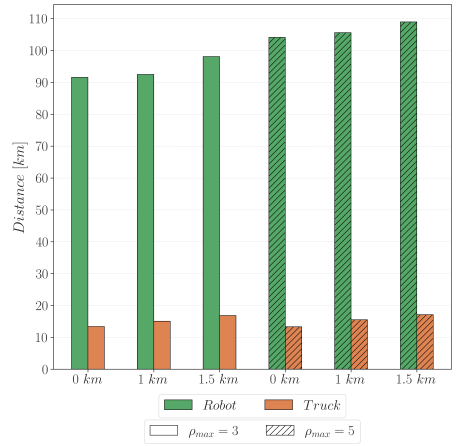
In Figure (5.3b), we can see that if we do not allow f_1 to worsen with respect to its optimal value f_1^*, the worsening of f_2 with respect to f_2^* is much more pronounced, and this is more

evident for higher values of ρ_{max}. An important aspect shown in the figure is that a small worsening of the environmental impact by just 5% can drastically mitigate the poor initial QoS value when the allowed worsening is none.

The role of θ in Figure 5.3b is more nuanced when f_1 is fixed to its optimal value f_1^*. Introducing conflicts forces the selection of fewer facilities, which can yield multiple emission-equivalent optimal solutions. In this setting, changing which facilities are activated can preserve cost optimality while revealing better-suited subsets of hubs for improving service quality, without sacrificing emission performance. This effect is more pronounced when customer due dates are more relaxed ($\rho_{max} = 5$). However, this quality advantage diminishes and eventually disappears once even small increases in emissions are permitted, since relaxing the emission constraint enlarges the solution space and reduces the relative influence of conflict-induced facility selection on overall QoS.



(a) Mean absolute value of function f_1 components



(b) Mean absolute value of km traveled by components

Figure 5.5: Objective function f_1 components

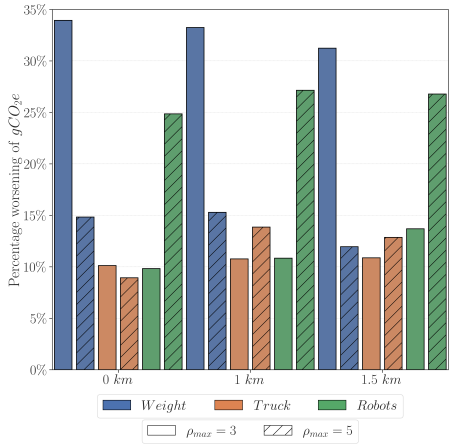
θ	**ΔRobot**	**ΔTruck**	**ΔWeight**	**ΔTotal**
0	78	-7	-10	61
1	80	29	-11	98
1.5	67	17	-13	71
Avg.	75	13	-11	77

Table 5.8: Absolute difference of gCO_2e from $\rho_{max} = 3$ to $\rho_{max} = 5$.

In Figure 5.5a and Figure 5.6a, we analyze how each component, i.e., the truck's distance (Truck) and weight (Weight) and the traveling times of the robots (Robots), impacts the global emissions gCO_2e. In particular, Figure 5.5a analyze the absolute value of such components when f_1 is optimized while restricting f_2 to its optimal value for $\rho_{max} = 3$ and 5, respectively. Figure 5.6a shows the percentage increase of gCO_2e from the case where f_1 is optimized without restrictions on f_2.

Figure 5.5a shows that, on average, the truck accounts for the largest share of gCO_2e emissions, despite traveling a total distance substantially lower than the cumulative distance

covered by the robot fleet, as shown in Figure 5.5b. Interestingly, payload weight does not significantly impact total emissions. Relaxing the due dates to $\rho_{max} = 5$ increases overall emissions by approximately 76 gCO_2e; however, the relative weight contribution diminishes because the truck offloads parcels earlier along its route.

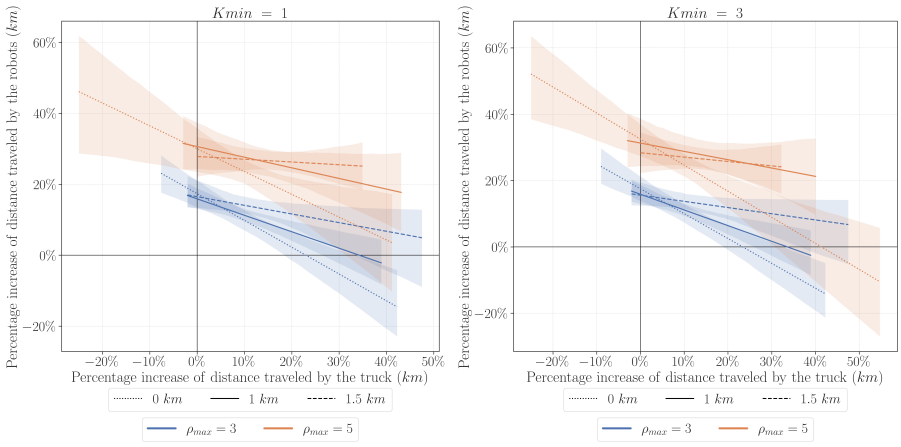


(a) Percentage increase of function f_1 components

Figure 5.6: Objective function f_1 components

Figure 5.6a shows that for $\rho_{max} = 3$, the component with the highest increase is the weight, reaching more than 30%, while the other two components show a much more limited change, about 10% increase for the truck and the robots. For $\rho_{max} = 5$, the most notable change with respect to $\rho_{max} = 3$ is that the component with the highest increase in percentage are the robots, which goes from about 10% to over 20%. This can easily be explained since when optimizing just for f_1 it does not exist a real incentive of respecting the due dates, and by that the parcels could stay on the truck more than is necessary, in order to minimize the emissions, while if we also consider the due dates the parcels could force the truck to partially change the most effective route in order to maintain a good quality of service. This is also more clear when enlarging the due-dates, since in that instance the parcels don't change as much as in the previous case. The increase in the due date however drastically increase the robot impact, this is however obvious given the above stated result about the due-dates. When restricting for the f_2 objective due dates must be respected and by that if they are too narrow the parcels will stay more on the truck in order to save time, vice-versa when the due dates are relaxed it allows a more intensive use of the ADRs, and by that trade time with emissions. With respect to the θ parameter the overall trend stays the same, however the increase of the it mitigates the effects, in simple terms, the impact of each components with respect to the unconstrained objective is less impacted, so the increase is partially mitigated by his introduction.

Figure 5.7 shows the regression lines of the percentage change in the distance traveled by the truck (horizontal axis) and the distance traveled by the robots (vertical axis) across all tested instances, when optimizing f_1 without any constraint on f_2, and then by limiting f_2 to its optimal value f_2^*. Figure 5.7a considers $k_{min} = 1$, while Figure 5.7b reports results for $k_{min} = 3$.In all the images the impact of θ parameter is shown.



(a) $k_{min} = 1$ (b) $k_{min} = 3$

Figure 5.7: Percentage change of distance traveled by the truck and time traveled by the robots when optimizing f_1 with and without constraining f_2 to its optimal value

From Figure 5.7a and Figure 5.7b, one aspect is clear, that no strong difference exists between the case with $k_{min} = 1$ and $k_{min} = 3$, this suggests that the facility coverage of customers does not impact the results. This is interesting since the increase of the cover by the facility should have some impact in the traveled distance by the truck and/or the robots, and having the overall results practically the same suggests that the facility distribution, with respect to the customers is not as fundamental as supposed, when considering the environmental impact. However a strong difference exists when considering different due-dates, as the regression lines are far-away one to the other. Relaxing ρ_{max}, independently of θ, produces a higher traveled distance from the optimal by the robots, but this is not new given the already presented results in Figure 5.3a. Regarding the conflict radius, it is shown that the higher parameter, the more constant the robot impact becomes, that is, the linear relationship becomes less negative. This means that, as we move from the unconstrained solution to the quality-constrained one, the increase in the truck's distance is less and less correlated with the actual variation in the robot's behavior.

Finally, as an example, we report the Pareto frontier generated for one of the instances in Set A. It is worth noticing the abrupt decrease in objective function f_1 value when moving from the point corresponding to the optimal value of f_2 ($f_2 = 15$) to the next one ($f_2 = 16$). This behavior is a general one across all instances, as seen when discussing results reported in Figure 5.3. It is also interesting that for all values of f_2 between 25 and 30, the value of f_1 remains constant, indicating that a large decrease in QoS is needed for a very limited improvement in terms of emissions.

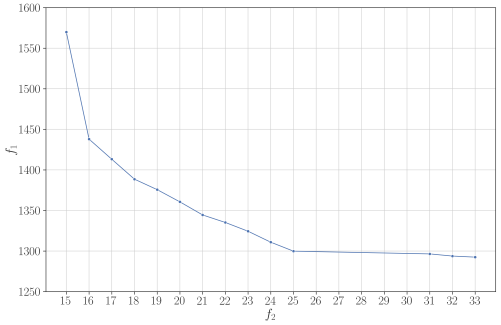


Figure 5.8: Example of a Pareto-efficient frontier

5.5.3 2P-KS performance analysis

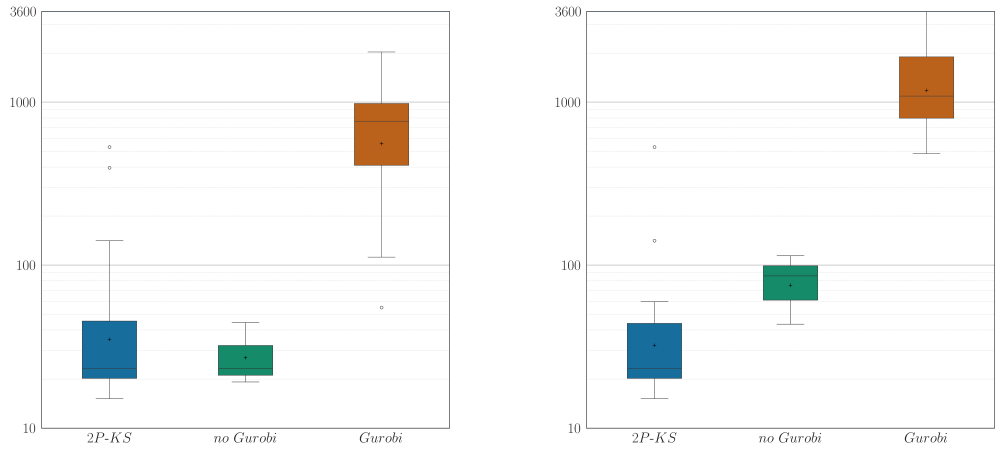
In the following experimental analysis, for each instance, we only consider a reference point of the Pareto frontier where the ϵ value for function f_2 has been set equal to the best solution found by the mathematical model when optimizing only objective function (5.2) within a predefined time limit of 15 minutes. This section aims to assess the performance of the proposed 2P-KS method against Gurobi on the more challenging set of instances (Set B). A time limit of 3600s was defined for Gurobi and 900s for the 2P-KS have been set for instances in Set B. The 2P-KS total time is split between the first phase (300s) and the second phase (600s).

			Gurobi					**2P-KS**	
Instance	TTF	TTB	Obj Best	Gap(%)	Time	TTF	TTB	Obj Best	Δ(%)
TZM_1	892	930	4237.45	7.7	3600	141	352	4237.45	0.0
TZM_2	527	3337	4360.03	8.0	3600	43	104	4372.12	0.3
TZM_3	513	513	**4328.17**	**0.0**	1150	22	49	**4328.17**	0.0
TZM_4	1093	2763	4291.56	5.6	3600	60	872	4291.56	0.0
TZM_5	266	827	4195.87	3.0	3600	23	83	4195.87	0.0
TZM_6	827	1483	4319.47	5.6	3600	22	94	4319.47	0.0
TZM_7	965	3063	4128.32	7.6	3600	32	577	4155.29	0.7
TZM_8	128	1710	**3916.33**	**0.0**	2815	23	101	**3916.33**	0.0
TZM_9	1394	1710	4133.28	1.8	3600	15	89	4133.28	0.0
TZM_10	998	1232	4575.31	11.1	3600	56	62	4361.02	-4.7
TZM_11	428	2060	**4225.16**	**0.0**	2063	19	84	**4225.16**	0.0
TZM_12	1066	2746	4205.57	1.8	3600	23	86	4205.57	0.0
TZM_13	-	-	-	-	3600	19	42	4134.81	-
TZM_14	892	1037	4209.05	3.3	3600	21	60	4174.05	-0.8
TZM_15	55	425	**4105.91**	**0.0**	869	19	20	**4105.91**	0.0
TZM_16	292	393	**4069.69**	**0.0**	988	20	893	**4069.69**	0.0
TZM_17	613	664	4187.77	1.5	3600	20	43	4192.47	0.1
TZM_18	743	757	**4176.50**	**0.0**	2075	22	86	**4176.50**	0.0
TZM_19	392	1557	4266.66	1.9	3600	20	447	4292.73	0.6
TZM_20	1007	1307	4453.55	9.0	3600	396	900	4453.55	0.0
TZM_21	138	1880	4415.29	6.4	3600	32	92	4415.29	0.0
TZM_22	652	3161	3917.00	2.0	3600	36	825	3917.00	0.0
TZM_23	-	-	-	-	3600	44	233	4374.60	-
TZM_24	112	2312	**3889.03**	**0.0**	2417	18	24	**3889.03**	0.0
TZM_25	1037	1117	4143.37	5.9	3600	18	320	4163.65	0.5
TZM_26	764	3198	4220.78	3.5	3600	47	454	4220.78	0.0
TZM_27	-	-	-	-	3600	23	84	4099.01	-
TZM_28	764	1260	4399.54	4.5	3600	28	394	4399.54	0.0
TZM_29	906	947	4296.58	4.7	3600	54	375	4296.58	0.0
TZM_30	2032	3400	5029.73	17.8	3600	530	898	4820.29	-4.2
Average	722.07	1695.89	4248.04	4.17	3172.83	61.71	291.39	4231.23	-0.28

Table 5.9: Gurobi and 2P-KS on instances from B1 set

	Gurobi					2P-KS			
Instance	TTF	TTB	Obj Best	Gap(%)	Time	TTF	TTB	Obj Best	Δ(%)
TZM_1	1931	2077	4320.26	5.8	3600	43	123	4320.26	0.0
TZM_2	726	726	4595.02	10.4	3600	60	330	4560.67	-0.8
TZM_3	1113	1457	4869.46	9.9	3600	131	465	4940.48	1.5
TZM_4	2570	2769	4754.67	6.9	3600	35	439	4725.32	-0.6
TZM_5	975	1730	4532.82	2.9	3600	22	349	4532.82	0.0
TZM_6	1027	3561	4553.76	0.5	3600	58	423	4553.76	0.0
TZM_7	1142	1355	4500.55	7.7	3600	201	348	4733.63	5.2
TZM_8	736	3309	4517.49	13.7	3600	45	465	4368.61	-3.3
TZM_9	3600	3600	8217.20	50.7	3600	62	197	4367.46	-46.9
TZM_10	709	993	4773.37	7.7	3600	45	381	4773.37	0.0
TZM_11	914	2280	4524.35	5.4	3600	102	465	4546.28	0.5
TZM_12	-	-	-	-	3600	86	465	4653.95	-
TZM_13	1089	1282	4329.18	3.3	3600	49	343	4329.18	0.0
TZM_14	-	-	-	-	3600	114	872	4615.59	-
TZM_15	2531	3597	5391.35	12.6	3600	111	448	4941.53	-8.3
TZM_16	863	980	4446.75	5.2	3600	53	809	4394.61	-1.2
TZM_17	663	2535	4861.17	7.1	3600	96	123	4819.32	-0.9
TZM_18	885	1442	5276.03	14.8	3600	64	87	5162.61	-2.2
TZM_19	892	1874	4417.98	3.9	3600	51	92	4417.98	0.0
TZM_20	-	-	-	-	3600	43	437	4936.28	-
TZM_21	1861	2274	4825.77	2.2	3600	62	857	4825.77	0.0
TZM_22	2236	3077	5615.00	24.0	3600	494	900	4783.29	-14.8
TZM_23	483	646	4619.78	3.8	3600	51	52	4619.78	0.0
TZM_24	2025	3577	4297.62	1.6	3600	39	450	4297.62	0.0
TZM_25	2442	3534	4644.54	7.0	3600	98	798	4599.42	-1.0
TZM_26	1138	2055	4964.12	5.1	3600	62	402	4964.12	0.0
TZM_27	717	1244	4653.89	0.9	3600	120	444	4717.64	1.4
TZM_28	1373	1773	4804.14	7.5	3600	63	100	4804.14	0.0
TZM_29	666	1237	4860.37	5.2	3600	113	900	4860.37	0.0
TZM_30	1222	2171	4924.05	13.7	3600	105	381	4741.84	-3.7
Average	1352.93	2116.85	4855.21	8.87	3600.00	89.27	431.56	4663.59	-2.78

Table 5.10: Gurobi and 2P-KS on instances from B1.5 set



(a) Set $B1$ (b) Set $B1.5$

Figure 5.9: Time To First results

Table 5.9 and Table 5.10 show the performance of Gurobi and 2P-KS on the $B1$ and $B1.5$ instance sets, respectively. In each table, the first column lists the name of the solved instances, followed by the next three columns displaying the time to first TTF (time required to find the first feasible solution), time to best TTB (time required to find the best integer solution), the value of the best integer solution found (Obj Best), which potentially represents

the optimal value, the Mip gap measuring the percentage distance of the solution from the best bound (Gap %) and the total time in seconds (Time). Since Gurobi is stopped when the time limit of 3600 seconds is reached, if it completes in less than 3600 seconds, it indicates that Gurobi found the optimal solution. The last four columns provide the same information related to TTF, TTB, and Obj Best for 2P-KS in the first three, whereas column $\Delta(\%)$ reports the percentage distance of 2P-KS solution value with respect to the best solution found by Gurobi. $\Delta(\%)$ has been computed as 2P-KS Obj Best minus Gurobi's Obj Best, divided by Gurobi's Obj Best (a negative value implies 2P-KS outperformed Gurobi).

Comparing Tables 5.9 and 5.10, it is evident how 2P-KS outperforms the solver in all the instances when TTF is considered. Even with regard to TTB, the performance of 2P-KS is better than that of Gurobi but for one instance (TZM_16 in Set B1). In set B1, Gurobi was able to find the optimal solution in 7 instances (emphasized in bold font) with an average solution time of 1768 seconds. In all such cases, 2P-KS also found the optimal solution in about half the time. Globally, Gurobi and 2P-KS find the same incumbent solutions in 19/30 instances in set B1, and in 12/30 in set B1.5. Gurobi beats 2P-KS in 5 instances of set B1 with an average distance of 0.44% and in 4 instances of set B1.5 with an average distance of 2.15%. However, the Δ (%) column shows that 2P-KS outperforms Gurobi by an average of 0.28% in less challenging instances (Table 5.9), with a peak of 4.7%. This advantage is more pronounced in complex scenarios: in highly constrained settings (Set B1.5), Gurobi occasionally struggles to navigate the branch-and-bound tree within the allotted time, whereas 2P-KS locates solutions up to 46.9% better, with an average improvement of 2.78% (Table 5.10).

The observations made above are further supported by the box plots in Figures 5.9 and 5.10 reporting TTF and TTB, respectively. Figures 5.9a and 5.9b display the TTF for Sets B1 and B1.5. Since not all instances have been solved by both methods, the figures have three columns: the first concerns all instances solved by 2P-KS, the second limits the analysis to those instances where 2P-KS finds a solution but Gurobi does not (no Gurobi), and the last one refers to the time for instances solved with Gurobi (27/30 instances in Set B1 and 27/30 in B1.5). Similarly, in Figures 5.10a and 5.10b for TTB.

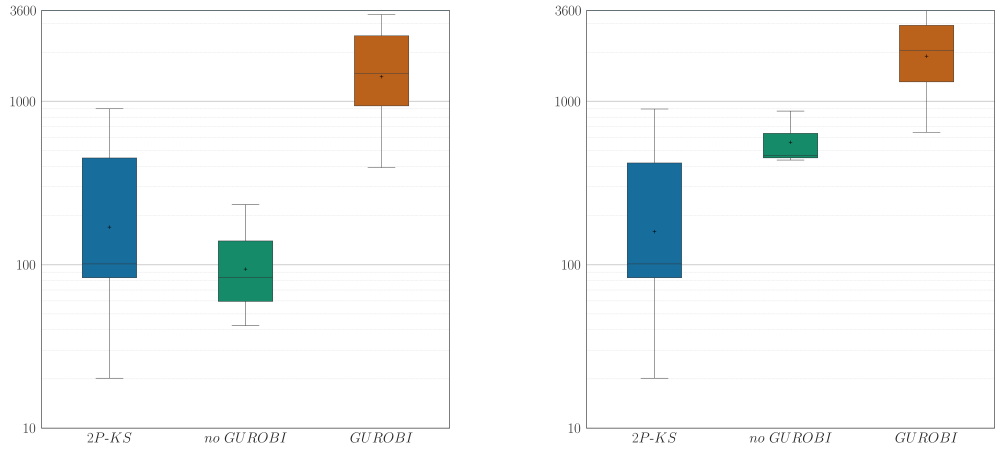
It is worth noticing that Gurobi's TTF values are significantly higher than those of 2P-KS, even when the analysis is restricted to the more complex instances where Gurobi is unable to find a feasible solution (no Gurobi box plot). The gap is more evident when a larger radius of conflict is considered (Set B1.5). It is interesting to notice that for 2P-KS, the median value of TTF is below 20 in Set B1 and below 70 in Set B1.5, suggesting that the first feasible solution has always been found in the first phase of the algorithm, with the exception of 3 cases (2 in Set B1 and one in Set B1.5, where the time limit of the first phase was set to 300 seconds). The increase in median value for Set B1.5 can be explained by the fact that such instances include a larger number of conflicts, making it more challenging to find a feasible solution.

Figures 5.10a and 5.10b show similar results but for TTB, where 2P-KS stands out as always faster than the MILP solver (but in one instance) in finding the best incumbent solution within the time limit.

Regarding Figure 5.11, we can see that the second phase of our 2P-KS obtains, on average, an improvement included between 0.5% and 1% with three outliers above 2% in Set B1, and around 2% with a whisker reaching 6% in Set B1.5.

5.5.4 Analysis of a realistic scenario

This section examines the problem through a realistic perspective. The municipal government provides a local grocery chain conditional access to its fleet of ADRs as an incentive to



(a) Set $B1$ (b) Set $B1.5$

Figure 5.10: Time To Best results

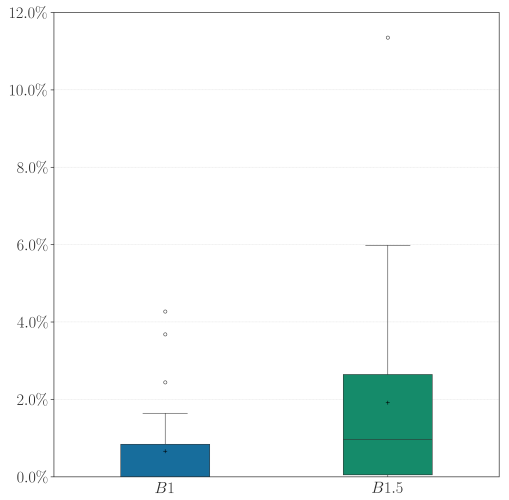
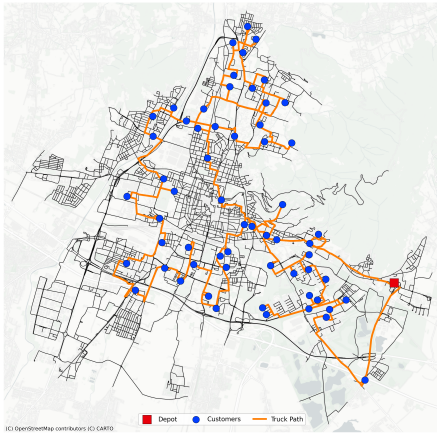
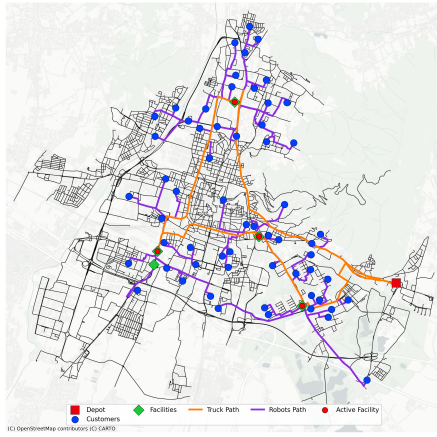


Figure 5.11: Improvement between the first and second phase of 2P-KS.

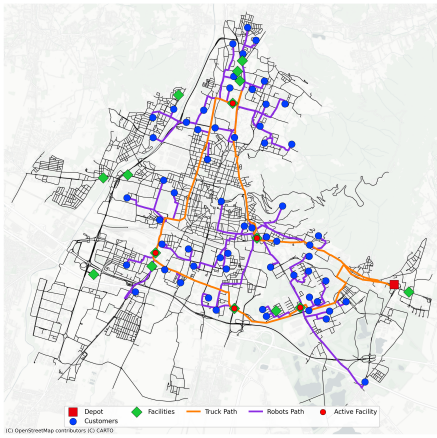
reduce the environmental footprint of deliveries. To curb total CO_2 emissions, the municipality mandates a minimum operational distance between active facilities, thereby preventing service concentration, deterring monopolization of specific urban sectors, and promoting a more uniform spatial distribution of vehicular emissions and traffic. Recognizing the need to maintain the service's appeal to private companies, the city grants retail chains the liberty to select preferred facility locations rather than forcibly assigning them. The company must then appraise these configurations to determine which tier offers the highest operational and environmental efficiency. The municipality may permit 0, 5, 15, or 30 facilities, with a conflict radius of either 1 or 1.5 kilometers. Instances are generated using Open Street Map (OpenStreetMap contributors, 2017) assuming a single-day delivery in the city of Brescia (Italy). Each facility has 20 robots available, and 60 customers are distributed across residen-



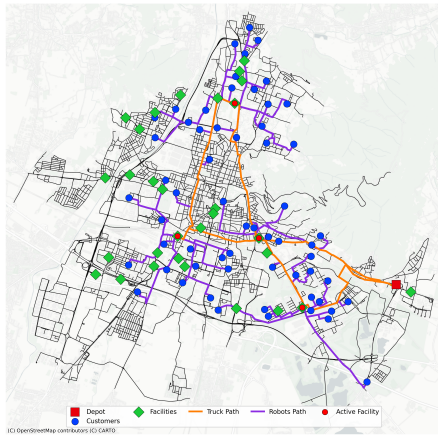
(a) Classical Door-to-Door



(b) G2T-RADP with 5 facilities



(c) G2T-RADP with 15 facilities



(d) G2T-RADP with 30 facilities

Figure 5.12: Optimal solutions for one realistic scenario

tial areas of the city. The depot is located in the south-east, as shown in Figure 5.12. We optimally solved the problem under four scenarios: without any supporting facility (door-to-door delivery with no robots), with 5 facilities, with 15, and with 30. In total, 15 instances are defined and solved to obtain a comprehensive understanding of model behavior. Other parameters are $k_{min} = 3$, $\rho_{max} = 5$, and a minimum distance between facilities of 200m.

Figure 5.13 shows how standard last-mile delivery ($|K| = 0$) produces almost twice the emissions of all robot-assisted cases. It is worth noting that the number of facilities made available by the municipality does not affect the final emissions. This can be explained by the fact that the model, given the high number of robots per facility and the conflict radius, selects only a small subset of the available facilities. The influence of the conflict radius is less pronounced than expected, which suggests that inherent geographic correlations between facility and customer distributions may reduce the relevance of this parameter.

From the municipality's perspective, the introduction of an ADR fleet successfully reduces urban emissions even under realistic topographical conditions. The total truck dis-

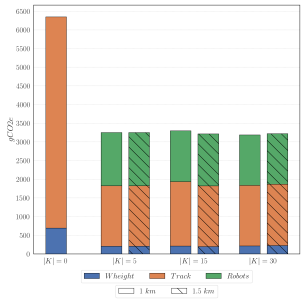


Figure 5.13: Function f_1 components for the realistic scenarios

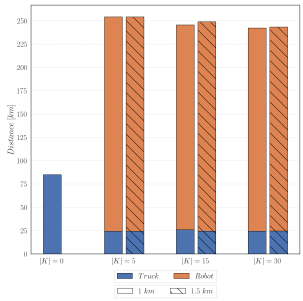


Figure 5.14: Kilometers traveled by components

tance traveled decreases by almost 70%; however, the combined distance covered by both the truck and the robot fleet is substantially higher. Crucially, the company activates far fewer facilities than are available, and those selected are evenly distributed across the operating area without spatial concentration. This confirms that the conflict radius effectively discourages monopolization of specific zones, achieving the municipality's spatial fairness objective. Finally, increasing the conflict radius from 1 km to 1.5 km does not significantly affect total emissions, granting policymakers the flexibility to tighten the constraint and promote equitable access among competing grocery operators.

5.6 Conclusions and Future Work

This study examined a two-tier, fully electric last-mile delivery system integrating ADRs. We developed a bi-objective optimization model that minimizes greenhouse gas emissions while maintaining high service quality and explicitly accounting for conflicts among active urban facilities. Extensive computational experiments revealed how environmental performance and service quality interact, providing actionable insights for both logistics operators and public authorities. The proposed Two-Phase Kernel Search (2P-KS) heuristic efficiently handles large-scale instances, demonstrating strong performance in this problem class and comparing favorably against a state-of-the-art commercial MILP solver.

From a managerial perspective, organizations can get near-optimal service quality while significantly lowering emissions. From a municipal standpoint, the model demonstrates how access to shared electric fleets can facilitate decarbonization in urban areas, while the introduction of conflict zones can readily ensure some level of fairness among companies without

undermining decarbonization objectives or compromising delivery performance. The integration between business efficiency and public sustainability goals underscores the possibility of combined city-industry methods for achieving CO_2-neutral logistics.

Future research could further enhance the proposed framework by relaxing some of the assumptions adopted in the current study. In particular, the model considers an offline planning setting in which customer demands, travel conditions, and operational parameters are assumed to be known when delivery plans are generated. While this assumption enables a clear assessment of the trade-offs between environmental impact and service quality, real urban logistics systems are inherently dynamic, with demand fluctuations, traffic conditions, and operational disruptions continuously affecting routing decisions. Incorporating these sources of uncertainty through stochastic or robust optimization techniques, would further improve the practical applicability of the proposed methodology. Additional research could also investigate how the spatial distribution of robots, the topology of the urban network, and dynamic robot reallocation strategies influence both operational efficiency and environmental sustainability.

Chapter 6

Conclusions

This thesis has investigated the application of advanced mathematical programming techniques to complex scheduling problems arising in manufacturing and distribution systems. The work has been motivated by the increasing need to manage highly interconnected industrial processes, where multiple decision layers, conflicting objectives, and operational constraints must be addressed simultaneously.

From a methodological perspective, the thesis has demonstrated the effectiveness of integrating exact optimization approaches with heuristic and matheuristic techniques. Exact methods, primarily based on Branch-and-Cut frameworks, have been successfully employed to derive optimal solutions and strong bounds for small and medium-sized instances. At the same time, their limitations in large-scale and highly complex settings have been addressed through the design of tailored heuristics and matheuristics, capable of efficiently exploring large solution spaces and producing high-quality solutions within practical computational times.

A central theme of this work is the development of models that balance realism and tractability. In real-world industrial applications, overly simplified models often fail to capture critical operational constraints, while excessively detailed formulations may become computationally intractable. The proposed approaches show that, through careful modeling choices and the exploitation of problem structure, it is possible to achieve an effective compromise between these two aspects.

The first part of the thesis has focused on manufacturing systems, with particular attention to steelmaking processes. Several decision problems have been addressed, including scrap blending, multi-period production planning, and hybrid flow shop scheduling. For each of these problems, novel mathematical formulations have been introduced, explicitly incorporating industrial constraints such as energy consumption, material handling limitations, and process-specific requirements. The results obtained highlight the importance of integrated decision-making in manufacturing environments. In particular, the proposed models show that considering multiple aspects jointly, such as cost, energy, and quality, leads to significantly improved solutions compared to traditional approaches based on isolated optimization problems. Moreover, the analysis of trade-offs in multi-objective settings provides valuable insights for decision makers, supporting more informed and flexible planning strategies.

The second part of the thesis has extended these methodologies to distribution systems, with a focus on last-mile delivery. In this context, the introduction of autonomous delivery technologies has added new dimensions to classical routing and scheduling problems.

The proposed multi-objective optimization framework captures the trade-off between operational efficiency and environmental sustainability, highlighting the potential of integrated vehicle–robot delivery systems.

The computational experiments conducted throughout the thesis confirm the effectiveness of the proposed approaches. The developed methods consistently achieve high-quality solutions, often significantly improving upon baseline or industry-inspired solutions. Furthermore, the analysis of realistic scenarios demonstrates the practical applicability of the models and their ability to generate actionable managerial insights.

Overall, the contributions of this thesis can be summarized along three main directions: (i) the development of innovative mathematical programming models for complex industrial problems, (ii) the design of advanced solution methodologies combining exact and heuristic techniques, and (iii) the application of these approaches to real-world manufacturing and distribution systems.

Despite the advances presented, several avenues for future research remain open. First, the integration of uncertainty represents a natural extension of this work. Many industrial parameters, such as demand, processing times, and material characteristics, are inherently uncertain, and incorporating stochastic or robust optimization techniques could further enhance the realism of the models.

Second, the increasing availability of real-time data opens new opportunities for dynamic and adaptive optimization. Developing approaches capable of reacting to changes in the system, possibly through online or rolling-horizon frameworks, represents a promising research direction.

Third, the integration of different decision levels—strategic, tactical, and operational remains a challenging but important goal. Extending the proposed models to capture long-term planning decisions, while maintaining computational tractability, could provide significant benefits for industrial applications.

Finally, the growing importance of sustainability calls for further research on multi-objective optimization models that explicitly incorporate environmental and social criteria, alongside economic performance.

In conclusion, this thesis demonstrates that advanced mathematical programming, enhanced by decomposition strategies and matheuristic approaches, provides a powerful and flexible framework for addressing complex decision-making problems in modern industrial systems. By bridging the gap between theory and practice, the proposed methods contribute to the development of more efficient, sustainable, and resilient manufacturing and distribution processes.

Bibliography

Laurent Alfandari, Ivana Ljubić, and Marcos De Melo da Silva. A tailored benders decomposition approach for last-mile delivery with autonomous robots. *European Journal of Operational Research*, 299(2):510–525, 2022.

Julian Allen, Michael Browne, Allan Woodburn, and Jacques Leonardi. The role of urban consolidation centres in sustainable freight transport. *Transport Reviews*, 32(4):473–490, 2012.

Pedro Amorim, Nicole DeHoratius, Fredrik Eng-Larsson, and Sara Martins. Customer preferences for delivery service attributes in attended home delivery. *Management Science*, 70 (11):7559–7578, 2024.

Enrico Angelelli, Renata Mansini, and M. Grazia Speranza. Kernel search: A general heuristic for the multi-dimensional knapsack problem. *Computers & Operations Research*, 37(11): 2017–2026, 2010.

Enrico Angelelli, Renata Mansini, and M Grazia Speranza. Kernel search: A new heuristic framework for portfolio selection. *Computational Optimization and Applications*, 51: 345–361, 2012.

Davide Armellini, Paolo Borzone, Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf. Modeling and solving the steelmaking and casting scheduling problem. *International Transactions in Operational Research*, 27(1):57–90, 2020. doi: https://doi.org/10.1111/itor.12595. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12595.

J. Ashayeri, A.G.M. van Eijs, and P. Nederstigt. Blending modelling in a process manufacturing: A case study. *European Journal of Operational Research*, 72(3):460–468, 1994. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(94)90416-2. URL https://www.sciencedirect.com/science/article/pii/0377221794904162.

Iurii Bakach, Ann Melissa Campbell, and Jan Fabian Ehmke. A two-tier urban delivery network with robot-based deliveries. *Networks*, 78(4):461–483, 2021.

Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011. ISSN 0191-2615. Supply chain disruption and risk management.

Lena Bell, Stefan Spinler, Matthias Winkenbach, and Vincent Müller. Assessing economic, social and ecological impact of parcel-delivery interventions in integrated simulation. *Transportation Research Part D: Transport and Environment*, 121:103817, 2023.

Klaus P. Bernatzki, Michael R. Bussieck, Thomas Lindner, and Marco E. Lübbecke. Optimal scrap combination for steel production. *Operations-Research-Spektrum*, 20(4):251–258, Dec 1998. ISSN 1436-6304. doi: 10.1007/BF01539743. URL https://doi.org/10.1007/BF01539743.

Marta Biancolin and Lucia Rotaris. Environmental impact of business-to-consumer e-commerce: Does it matter to consumers? *Research in Transportation Business & Management*, 52:101087, 2024.

Valentina Bonomi, Renata Mansini, and Roberto Zanotti. Last mile delivery with parcel lockers: evaluating the environmental impact of eco-conscious consumer behavior. *IFAC-PapersOnLine*, 55(5):72–77, 2022.

Valentina Bonomi, Renata Mansini, and Roberto Zanotti. Mediating governance goals with patients and nurses satisfaction: a multi-actor multi-objective problem including fairness. *International Journal of Production Research*, 62(10):3753–3780, 2024.

Valentina Bonomi, Daniele Manerba, Renata Mansini, and Roberto Zanotti. Optimizing attended home delivery: Multiple recovery options and customer availability profiles to face synchronization failures. *International Journal of Production Economics*, 279:109463, 2025. ISSN 0925-5273.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. doi: 10.1109/TPAMI.2004.60.

Nils Boysen, Stefan Schwerdfeger, and Felix Weidinger. Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3): 1085–1099, 2018.

Nils Boysen, Stefan Fedtke, and Stefan Schwerdfeger. Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, 43(1):1–58, Mar 2021. doi: 10.1007/s00291-020-00607-8. URL https://doi.org/10.1007/s00291-020-00607-8.

U. Breunig, R. Baldacci, R.F. Hartl, and T. Vidal. The electric two-echelon vehicle routing problem. *Computers & Operations Research*, 103:198–210, 2019. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2018.11.005. URL https://www.sciencedirect.com/science/article/pii/S0305054818302909.

John E. Bringas. *Handbook of Comparative World Steel Standards*. ASTM International, West Conshohocken, PA, 3rd edition, 2004. ISBN 0-8031-3042-2. ASTM Data Series DS67B.

Cheng Chen and Emrah Demir. Drones and Delivery Robots: Models and Applications to Last Mile Delivery. In Saïd Salhi and John Boylan, editors, *The Palgrave Handbook of Operations Research*, Springer Books, pages 859–882. Springer, June 2022.

Cheng Chen, Emrah Demir, Yuan Huang, and Rongzu Qiu. The adoption of self-driving delivery robots in last mile logistics. *Transportation Research Part E: Logistics and Transportation Review*, 146:102214, 2021.

Teresa Corberán, Isaac Plana, José M. Sanchis, and Paula Segura. The multidepot drone general routing problem with duration and capacity constraints. *International Transactions in Operational Research*, 2024. In Press.

Annarita De Maio, Gianpaolo Ghiani, Demetrio Laganà, and Emanuele Manni. Sustainable last-mile distribution with autonomous delivery robots and public transportation. *Transportation Research Part C: Emerging Technologies*, 163:104615, 2024.

Emrah Demir, Tolga Bektaş, and Gilbert Laporte. The bi-objective pollution-routing problem. *European Journal of Operational Research*, 232(3):464–478, 2014. ISSN 0377-2217.

F. Dubeau and M. E. Ntigura Habingabwa. On best corrected mixture problems in metallurgy: A case study. *Mathematical Problems in Engineering*, 2017(1):2742540, 2017. doi: https://doi.org/10.1155/2017/2742540. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2017/2742540.

European Environment Agency. Greenhouse gas emission intensity of electricity generation. European Environment Agency, 2024. URL https://www.eea.europa.eu/en/analysis/maps-and-charts/co2-emission-intensity-15. [Online; accessed 11-October-2024].

Maria Pia Fanti, Giuliana Rotunno, Gabriella Stecco, Walter Ukovich, and Stefano Mininel. An integrated system for production scheduling in steelmaking and casting plants. *IEEE Transactions on Automation Science and Engineering*, 13(2):1112–1128, 2016. doi: 10.1109/TASE.2015.2477362.

Miguel A. Figliozzi. Carbon emissions reductions in last mile and grocery deliveries utilizing air and ground autonomous vehicles. *Transportation Research Part D: Transport and Environment*, 85:102443, 2020.

Christodoulos A. Floudas and Xiaoxia Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, 28(11):2109–2129, 2004. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2004.05.002. URL https://www.sciencedirect.com/science/article/pii/S0098135404001401.

Ford. Ford e-transit brochure. Ford.it, 2024. URL https://www.ford.it/content/dam/guxeu/it/documents/pricelists/commercials/FT_ford_E_transit.pdf. [Online; accessed 11-October-2024].

Barbara Fornai, Marco Vannucci, Valentina Colla, Alessandro Amato, Giacomo Filippo Porzio, Anders Bjork, and Klara Westling. A software tool for comparing the economic value of different scrap mixes in steel production. In *2013 4th International Conference on Intelligent Systems, Modelling and Simulation*, pages 315–320, 2013. doi: 10.1109/ISMS.2013.12.

Mikele Gajda, Nils Boysen, and Olivier Gallay. Impact of bot return policies in van-and-bot delivery systems. *International Journal of Production Research*, 62(17):6360–6379, 2024.

Mikele Gajda, Olivier Gallay, Renata Mansini, and Filippo Ranza. Optimizing last-mile delivery through crowdshipping on public transportation networks. *Transportation Research Part C: Emerging Technologies*, 179:105250, 2025. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2025.105250. URL https://www.sciencedirect.com/science/article/pii/S0968090X25002542.

A. Gobbi, D. Manerba, R. Mansini, and R. Zanotti. A kernel search for a patient satisfaction-oriented nurse routing problem with time-windows. *IFAC-PapersOnLine*, 52 (13):1669–1674, 2019.

Alessandro Gobbi, Daniele Manerba, Renata Mansini, and Roberto Zanotti. Hybridizing adaptive large neighborhood search with kernel search: a new solution approach for the nurse routing problem with incompatible services and minimum demand. *International Transactions in Operational Research*, 30(1):8–38, 2023.

R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. Elsevier, 1979. doi: https://doi.org/10.1016/S0167-5060(08)70356-X.

Matt Grote, Tom Cherrett, Gary Whittle, and Neil Tuck. Environmental benefits from shared-fleet logistics: lessons from a public-private sector collaboration. *International Journal of Logistics Research and Applications*, 26(2):128–154, 2023.

Gianfranco Guastaroba and Maria Grazia Speranza. Kernel search for the capacitated facility location problem. *Journal of Heuristics*, 18(6):877–917, 2012.

Saïd Hanafi, Renata Mansini, and Roberto Zanotti. The multi-visit team orienteering problem with precedence constraints. *European Journal of Operational Research*, 282(2):515–529, 2020.

Jinghua Hao, Min Liu, Shenglong Jiang, and Cheng Wu. A soft-decision based two-layered scheduling approach for uncertain steelmaking-continuous casting process. *European Journal of Operational Research*, 244(3):966–979, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.02.026. URL https://www.sciencedirect.com/science/article/pii/S0377221715001277.

Iiro Harjunkoski and Ignacio E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering*, 26(11):1533–1552, 2002. ISSN 0098-1354. doi: https://doi.org/10.1016/S0098-1354(02)00100-X. URL https://www.sciencedirect.com/science/article/pii/S009813540200100X.

Juntaek Hong, Kyungduk Moon, Kangbok Lee, Kwansoo Lee, and Michael L. Pinedo. An iterated greedy matheuristic for scheduling in steelmaking-continuous casting process. *International Journal of Production Research*, 60(2):623–643, 2022. doi: 10.1080/00207543.2021.1975839.

Inger Beate Hovi and Eirill Bø. Unlocking the potential: How can parcel lockers drive efficiency and environmental friendliness in e-commerce? *Sustainable Futures*, 7:100189, 2024.

F. J. Hwang, Bohan Hu, and M. Y. Kovalyov. Supermarket-chain grocery delivery optimization through crowdshipping. *International Journal of Production Research*, 2024. In Press.

Miguel Iglesias-Escudero, Joaquín Villanueva-Balsera, Francisco Ortega-Fernandez, and Vicente Rodriguez-Montequín. Planning and scheduling with uncertainty in the steel sector: A review. *Applied Sciences*, 9(13), 2019. ISSN 2076-3417. doi: 10.3390/app9132692. URL https://www.mdpi.com/2076-3417/9/13/2692.

Sheng-Long Jiang. A soft encoding-based evolutionary algorithm for the steelmaking scheduling problem and its extension under energy thresholds. *Computers & Operations Research*, 174:106885, 2025. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2024.106885. URL https://www.sciencedirect.com/science/article/pii/S0305054824003575.

Siim Kallas. *White Paper on Transport: Roadmap to a Single European Transport Area–towards a Competitive and Resource-efficient Transport System.* Office for Official Publications of the European Communities, 2011.

J Kim and RL Lewis. A large scale linear programming application to least cost charging for foundry melting operations.(retroactive coverage). *Transactions of the American Foundrymen's Society.*, 95:735–744, 1987.

Seong-in Kim, Junghee Han, Youngho Lee, Bo-Kyung Shim, and Ibrahim A. Hameed. A mixed integer programming model for the multiperiod scheduling problem of blending zinc-bearing materials to minimize setups in the hydrometallurgical process. *Metallurgical and Materials Transactions B*, 39(6):896–904, Dec 2008. ISSN 1543-1916. doi: 10.1007/s11663-008-9194-y. URL https://doi.org/10.1007/s11663-008-9194-y.

S. C. Lenny Koh, Fu (Jeff) Jia, Yu (Jack) Gong, Xiaoxue Zheng, and Alexandre Dolgui. Achieving carbon neutrality via supply chain management: position paper and editorial for ijpr special issue. *International Journal of Production Research*, 61(18):6081–6092, 2023.

Przemysław Kowalik. *A chance-constrained blending problem with amounts of components available in fixed-size portions*, pages 61–77. 12 2019. ISBN 978-83-7947-371-7.

Leonardo Lamanna, Renata Mansini, and Roberto Zanotti. A two-phase kernel search variant for the multidimensional multiple-choice knapsack problem. *European Journal of Operational Research*, 297(1):53–65, 2022.

Myungho Lee, Kyungduk Moon, Kangbok Lee, Juntaek Hong, and Michael Pinedo. A critical review of planning and scheduling in steelmaking and continuous casting in the steel industry. *Journal of the Operational Research Society*, 75(8):1421–1455, 2024. doi: 10.1080/01605682.2023.2265416.

Clément Lemardelé, Sofia Pinheiro Melo, Felipe Cerdas, Christoph Herrmann, and Miquel Estrada. Life-cycle analysis of last-mile parcel delivery using autonomous delivery robots. *Transportation Research Part D: Transport and Environment*, 121:103842, 2023.

Eric K.H. Leung, Zhiyuan Ouyang, and George Q. Huang. Community logistics: a dynamic strategy for facilitating immediate parcel delivery to smart lockers. *International Journal of Production Research*, 61(9):2937–2962, 2023.

Jianyu Long, Zhong Zheng, and Xiaoqiang Gao. Dynamic scheduling in steelmaking-continuous casting production for continuous caster breakdown. *International Journal of Production Research*, 55(11):3197–3216, 2017. doi: 10.1080/00207543.2016.1268277.

Daniele Manerba, Renata Mansini, and Roberto Zanotti. Attended home delivery: reducing last-mile environmental impact by changing customer habits. *IFAC-PapersOnLine*, 51(5):55–60, 2018.

Daniele Manerba, Renata Mansini, Lorenzo Tomasetti, and Roberto Zanotti. Bi-objective optimization in steelmaking: Balancing scrap costs, energy consumption, and steel quality. *IFAC-PapersOnLine*, 59(10):1706–1711, 2025.

Renata Mansini and Roberto Zanotti. Two-phase kernel search: An application to facility location problems with incompatibilities. In *Proceedings of the 11th International Conference on Operations Research and Enterprise Systems - Volume 1: ICORES*, pages 105–111, 2022.

Renata Mansini, Marina Zanella, and Roberto Zanotti. Optimizing a complex multi-objective personnel scheduling problem jointly complying with requests from customers and staff. *Omega*, 114:102722, 2023.

Renata Mansini, Filippo Ranza, and Roberto Zanotti. Towards sustainable last-mile delivery: Introducing an off-peak urban policy to mitigate environmental and social impacts. *IFAC-PapersOnLine*, 58(2):186–191, 2024.

Imke Mattik, Pedro Amorim, and Hans-Otto Günther. Hierarchical scheduling of continuous casters and hot strip mills in the steel industry: a block planning application. *International Journal of Production Research*, 52(9):2576–2591, 2014. doi: 10.1080/00207543.2013.860248.

Hubert Missbauer, Wolfgang Hauber, and Werner Stadler. A scheduling system for the steelmaking-continuous casting process. a case study from the steelmaking industry. *International Journal of Production Research*, 47(15):4147–4172, 2009. doi: 10.1080/00207540801950136.

Abood Mourad, Jakob Puchinger, and Tom Van Woensel. Integrating autonomous delivery service into a passenger transportation system. *International Journal of Production Research*, 59(7):2116–2139, 2021.

Chase C. Murray and Amanda G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015.

OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

Manuel Ostermeier, Andreas Heimfarth, and Alexander Hübner. The multi-vehicle truck-and-robot routing problem for last-mile delivery. *European Journal of Operational Research*, 310(2):680–697, 2023.

Dario Pacciarelli and Marco Pranzo. Production scheduling in a steelmaking-continuous casting plant. *Computers & Chemical Engineering*, 28(12):2823–2835, 2004. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2004.08.031. URL https://www.sciencedirect.com/science/article/pii/S0098135404002637.

Quan-Ke Pan. An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. *European Journal of Operational Research*, 250(3):702–714, 2016. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.10.007. URL https://www.sciencedirect.com/science/article/pii/S0377221715009145.

Quan-Ke Pan, Ling Wang, Kun Mao, Jin-Hui Zhao, and Min Zhang. An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Transactions on Automation Science and Engineering*, 10(2):307–322, 2013. doi: 10.1109/TASE.2012.2204874.

Agnivesh Pani, Sabya Mishra, Mihalis Golias, and Miguel Figliozzi. Evaluating public acceptance of autonomous delivery robots during covid-19 pandemic. *Transportation Research Part D: Transport and Environment*, 89:102600, 2020.

Y K Park and J-M Yang. Enhancing the efficiency of a die casting process using scrap recycling and ingot adjustment. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 225(7):1105–1116, 2011.

David Pisinger and Mikkel Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51, 2007. doi: 10.1287/ijoc.1060.0181.

Wei Qi, Lefei Li, Sheng Liu, and Zuo-Jun Max Shen. Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact. *Manufacturing & Service Operations Management*, 20(4):737–751, 2018.

Said Rahal, Zukui Li, and Dimitri J. Papageorgiou. Proactive and reactive scheduling of the steelmaking and continuous casting process through adaptive robust optimization. *Computers & Chemical Engineering*, 133:106658, 2020. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2019.106658. URL https://www.sciencedirect.com/science/article/pii/S0098135419309196.

Luigi Ranieri, Salvatore Digiesi, Bartolomeo Silvestri, and Michele Roccotelli. A review of last mile logistics innovations in an externalities cost reduction vision. *Sustainability*, 10 (3):782, 2018.

Imma Ribas, Rainer Leisten, and Jose M. Framiñan. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439–1454, 2010. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2009.11.001. URL https://www.sciencedirect.com/science/article/pii/S0305054809002883. Operations Research and Data Mining in Biological Systems.

Aiying Rong and Risto Lahdelma. Fuzzy chance constrained linear programming model for optimizing the scrap charge in steel production. *European Journal of Operational Research*, 186(3):953–964, 2008. ISSN 0377-2217.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

Rubén Ruiz and José Antonio Vázquez-Rodríguez. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1–18, 2010. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2009.09.024. URL https://www.sciencedirect.com/science/article/pii/S0377221709006390.

Ümit Sami Sakallı and Ömer Faruk Baykoç. An optimization approach for brass casting blending problem under aleatory and epistemic uncertainties. *International Journal of Production Economics*, 133(2):708–718, 2011. ISSN 0925-5273. doi: 10.1016/j.ijpe.2011.05.008.

Umit Sami Sakalli and Burak Birgoren. A spreadsheet-based decision support tool for blending problems in brass casting industry. *Computers & Industrial Engineering*, 56(2):724–735, 2009. Design and Management of Production Systems.

Ümit Sami Sakallı, Ömer Faruk Baykoç, and Burak Birgören. Stochastic optimization for blending problem in brass casting industry. *Annals of Operations Research*, 186(1):141–157, Jun 2011. ISSN 1572-9338.

Iván Sánchez-Díaz, Peter Georén, and Märta Brolinson. Shifting urban freight deliveries to the off-peak hours: a review of theory and practice. *Transport Reviews*, 37(4):521–543, 2017.

Yannick Oskar Scherr, Bruno Albert Neumann Saavedra, Mike Hewitt, and Dirk Christian Mattfeld. Service network design with mixed autonomous fleets. *Transportation Research Part E: Logistics and Transportation Review*, 124:40–55, 2019. ISSN 1366-5545.

Yannick Oskar Scherr, Mike Hewitt, Bruno Albert Neumann Saavedra, and Dirk Christian Mattfeld. Dynamic discretization discovery for the service network design problem with mixed autonomous fleets. *Transportation Research Part B: Methodological*, 141:164–195, 2020. ISSN 0191-2615.

Yaohan Shen, Bipan Zou, René De Koster, and T.C.E. Cheng. Performance estimation and operating policies in a truck-based autonomous mobile robot delivery system. *International Journal of Production Research*, 2024. In Press.

Michele D. Simoni, Erhan Kutanoglu, and Christian G. Claudel. Optimization and analysis of a robot-assisted last mile delivery system. *Transportation Research Part E: Logistics and Transportation Review*, 142:102049, 2020. ISSN 1366-5545.

Sharan Srinivas, Surya Ramachandiran, and Suchithra Rajendran. Autonomous robot-driven deliveries: A review of recent developments and future directions. *Transportation Research Part E: Logistics and Transportation Review*, 165:102834, 2022.

George J. Stigler. The cost of subsistence. *American Journal of Agricultural Economics*, 27(2): 303–314, 05 1945. ISSN 0002-9092. doi: 10.2307/1231810. URL https://doi.org/10.2307/1231810.

Pengfei Su, Yue Zhou, and Jianzhong Wu. Multi-objective scheduling of a steelmaking plant integrated with renewable energy sources and energy storage systems: Balancing costs, emissions and make-span. *Journal of Cleaner Production*, 428:139350, 2023. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2023.139350. URL https://www.sciencedirect.com/science/article/pii/S0959652623035084.

Zhangsheng Su, Chao Deng, Raymond Chiong, Sheng-Long Jiang, and Kai Zhang. Hybrid flow shop scheduling with continuous processing and resource threshold constraints: A case of steel plant. *Expert Systems with Applications*, 284:127247, 2025. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2025.127247. URL https://www.sciencedirect.com/science/article/pii/S0957417425008693.

D. W. Sutton and P. A. Coates. On-line mixture calculation system for stainless steel production by bsc stainless: The least through cost mix system (ltcm). *Journal of the Operational Research Society*, 32(3):165–172, 1981. doi: 10.1057/jors.1981.33.

Lixin Tang, Jiyin Liu, Aiying Rong, and Zihou Yang. A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research*, 120(2):423–435, 2000. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(99)00041-7. URL https://www.sciencedirect.com/science/article/pii/S0377221799000417.

Lixin Tang, Peter B. Luh, Jiyin Liu, and Lei Fang. Steel-making process scheduling using lagrangian relaxation. *International Journal of Production Research*, 40(1):55–70, 2002. doi: 10.1080/00207540110073000.

Lixin Tang, Gongshu Wang, and Zhi-Long Chen. Integrated charge batching and casting width selection at baosteel. *Operations Research*, 62(4):772–787, 2014. doi: 10.1287/opre.2014.1278.

Marlin W. Ulmer and Barrett W. Thomas. Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks*, 72(4):475–505, 2018.

Shijin Wang, Ming Liu, and Chengbin Chu. A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling. *International Journal of Production Research*, 53(4):1143–1167, 2015. doi: 10.1080/00207543.2014.949363.

Jesse Weltevreden. European e-commerce report 2025, 2025.

World Steel Association. Fact sheet on scrap, 2021. URL https://worldsteel.org/wp-content/uploads/Fact-sheet-on-scrap_2021.pdf. Accessed: 2024-12-06.

World Steel Association. World steel in figures, 2024. URL https://worldsteel.org/wp-content/uploads/World-Steel-in-Figures-2024.pdf. Accessed: 2026-02-23.

World Steel Association. What is the eaf-bof process split, 2026. URL https://worldsteel.org/about-steel/facts/steelfacts/what-is-steel/what-is-the-eaf-bof-process-split/. Accessed: 2026-02-23.

Ye Yang, Weida Chen, Li Wei, and Xu Chen. Robust optimization for integrated scrap steel charge considering uncertain metal elements concentrations and production scheduling under time-of-use electricity tariff. *Journal of Cleaner Production*, 176:800–812, 2018. ISSN 0959-6526.

Yun Ye, Jie Li, Zukui Li, Qiuhua Tang, Xin Xiao, and Christodoulos A. Floudas. Robust optimization and stochastic programming approaches for medium-term production scheduling of a large-scale steelmaking continuous casting process under demand uncertainty. *Computers & Chemical Engineering*, 66:165–185, 2014. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2014.02.028. URL https://www.sciencedirect.com/science/article/pii/S0098135414000763. Selected papers from ESCAPE-23 (European Symposium on Computer Aided Process Engineering - 23), 9-12 June 2013, Lappeenranta, Finland.

Shaohua Yu, Jakob Puchinger, and Shudong Sun. Van-based robot hybrid pickup and delivery routing problem. *European Journal of Operational Research*, 298(3):894–914, 2022. ISSN 0377-2217.

Roberto Zanotti, Daniele Manerba, and Renata Mansini. The price of customer presence in attended home delivery with customer availability profiles. In *2025 20th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 625–634, 2025. doi: 10.15439/2025F1656.

Shuya Zhong, Carl Lomas, and Tracey Worth. Understanding customers' adoption of express delivery service for last-mile delivery in the uk. *International Journal of Logistics Research and Applications*, 25(12):1491–1508, 2022.

Bipan Zou, Siqing Wu, Yeming Gong, Zhe Yuan, and Yuqian Shi. Delivery network design of a locker-drone delivery system. *International Journal of Production Research*, 62(11): 4097–4121, 2024.