

Titre: Machine Learning For Large-Scale Personnel Scheduling Problems
Title:

Auteur: Farin Rastgar Amini
Author:

Date: 2026

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Rastgar Amini, F. (2026). Machine Learning For Large-Scale Personnel Scheduling Problems [Ph.D. thesis, Polytechnique Montréal]. PolyPublie.
Citation: <https://publications.polymtl.ca/76810/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/76810/>
PolyPublie URL:

Directeurs de recherche: Guy Desaulniers, & Claudio Contardo
Advisors:

Programme: Doctorat en génie industriel
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Machine Learning For Large-Scale Personnel Scheduling Problems

FARIN RASTGAR AMINI

Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie industriel

Mai 2026

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Cette thèse intitulée :

Machine Learning For Large-Scale Personnel Scheduling Problems

présentée par **Farin RASTGAR AMINI**

en vue de l'obtention du diplôme de *Philosophiæ Doctor*

a été dûment acceptée par le jury d'examen constitué de :

Nadia LAHRICHI, présidente

Guy DESAULNIERS, membre et directeur de recherche

Claudio CONTARDO, membre et codirecteur de recherche

Ryan O'NEIL, membre

Jérémy OMER, membre externe

ACKNOWLEDGEMENTS

I would first like to express my deepest gratitude to my supervisor, Professor Guy Desaulniers, and my co-supervisor, Professor Claudio Contardo, for their guidance, support, patience, and encouragement throughout my doctoral studies. I am sincerely thankful for the time they devoted to my work, for their insightful comments, and for the high standards of scientific rigor they continuously encouraged me to pursue. Their mentorship has had a profound impact on both this thesis and my development as a researcher.

I would also like to warmly thank my co-authors, Professor Daniel Aloise and Professor Maxime Gasse, for their valuable collaboration, thoughtful feedback, and enriching discussions. Their contributions have greatly strengthened the quality of this research.

I am also sincerely grateful to the members of the jury, Professor Nadia Lahrichi, Professor Ryan O’Neil, and Professor J  r  my Omer, for accepting to evaluate this thesis and for the time and care they devote to this work.

I would like to acknowledge Polytechnique Montr  al for providing me with an excellent academic environment in which to pursue my doctoral studies. I am also grateful to GERAD for offering a stimulating research setting and for fostering exchanges with researchers and students working on challenging and inspiring problems in operations research.

I would also like to express my gratitude to UKG for its support and collaboration throughout this research. Working on problems motivated by real-world applications made this work especially meaningful and gave me the opportunity to connect academic research with practical challenges in personnel scheduling.

My sincere thanks also go to my friends and colleagues who accompanied me throughout this journey. Their kindness, encouragement, and presence made these years lighter and more memorable. Whether through academic discussions, friendly conversations, or simple moments of support, they helped me more than they may realize.

I would like to express my deepest gratitude to my family for their unconditional love, encouragement, and constant belief in me. Their support has been my foundation throughout this journey. I am especially thankful to my parents, whose sacrifices, care, and encouragement have shaped every step of my path. I owe them more than words can express.

Finally, and most importantly, I would like to thank my husband, Saba, for his unwavering support, patience, understanding, and love throughout these years. He stood by me through every high and low of this journey and gave me strength whenever I needed it most. His

encouragement and belief in me carried me through the most difficult moments, and I am deeply grateful to have shared this path with him.

To all of you, thank you.

RÉSUMÉ

Les problèmes de planification d’horaires du personnel consistent à établir les horaires de travail des employés de manière à garantir que les besoins en main-d’œuvre soient satisfaits sur un horizon de planification donné. Ces problèmes font l’objet de nombreuses études dans divers secteurs et revêtent une importance particulière dans les industries de services telles que le commerce de détail, où la demande en main-d’œuvre varie considérablement au cours de la journée, ce qui rend les coûts de main-d’œuvre cruciaux. Dans de tels environnements, les horaires sont généralement élaborés à partir d’un grand nombre de quarts de travail potentiels, caractérisés par des heures de début, des heures de fin et des durées flexibles. Lorsque l’énumération explicite de ces quarts de travail est requise pour des modèles de programmation linéaire en nombres entiers mixtes, ceux-ci peuvent contenir un très grand nombre de variables et devenir difficiles à résoudre à l’aide de solveurs d’optimisation génériques. Par conséquent, réduire la taille de ces modèles tout en préservant la qualité de la solution constitue un défi majeur pour les problèmes de planification du personnel de grande taille. Dans cette thèse, nous étudions comment des techniques de la science des données, notamment l’apprentissage machine et l’exploration de données, peuvent être utilisées pour guider le processus d’énumération des quarts de travail et améliorer l’évolutivité des modèles d’optimisation pour la planification du personnel dans le secteur de la vente au détail.

Tout d’abord, nous proposons un cadre assisté par l’apprentissage pour guider l’énumération des quarts de travail dans les problèmes de planification flexible du personnel avec des quarts de travail mono-tâche. L’approche s’appuie sur des modèles d’apprentissage profond entraînés sur des instances de planification précédemment résolues afin d’identifier des plages horaires prometteuses pour les heures de début et de fin des quarts de travail. Au lieu d’énumérer tous les quarts de travail possibles, la méthode limite le processus d’énumération à ceux qui commencent ou se terminent aux moments prédits, réduisant ainsi considérablement le nombre de quarts de travail potentiels pris en compte dans le modèle d’optimisation. Des expériences numériques menées sur des cas concrets du secteur de la vente au détail, comportant jusqu’à 10 tâches et 50 employés, ont démontré que l’approche proposée peut accélérer le processus de résolution d’un facteur moyen de 47, tout en limitant l’augmentation moyenne du coût à 1.28%.

Deuxièmement, nous étudions comment les décisions de planification historiques peuvent être exploitées à l’aide de techniques d’exploration de données afin d’améliorer à nouveau l’efficacité de la résolution de problèmes de planification du personnel de grande taille. Étant

donné que les cas de planification dans les environnements de vente au détail sont résolus de manière répétée au fil du temps et présentent souvent des modèles de demande similaires, nous proposons une méthode basée sur l’algorithme des k plus proches voisins pour identifier les cas historiques similaires à un nouveau cas. Les modèles de quarts de travail apparaissant dans les solutions de ces instances historiques sont ensuite utilisés pour restreindre l’énumération des quarts de travail candidats dans le nouveau problème. Cette approche permet au processus d’optimisation de tirer parti des structures de solution observées précédemment. Des expériences numériques menées sur des instances impliquant jusqu’à 12 types de tâches et 252 employés montrent que la méthode proposée peut éliminer en moyenne plus de 85% des variables de décision, ce qui conduit à un facteur d’accélération moyen supérieur à 15 tout en maintenant les coûts de solution à environ 1% des coûts des solutions de référence.

Enfin, nous étendons le cadre assisté par l’apprentissage machine au contexte plus complexe de la planification des horaires avec quarts multi-tâches. Dans de nombreux environnements de vente au détail, les employés peuvent effectuer plusieurs tâches différentes au cours d’un même quart de travail, ce qui augmente considérablement le nombre de configurations d’horaires possibles. Nous développons une formulation explicite de couverture d’ensembles qui intègre à la fois les quarts mono-tâche et multi-tâches, chaque quart pouvant se composer de plusieurs segments de tâche. Pour maîtriser la croissance combinatoire du nombre de variables qui en résulte, nous utilisons un modèle d’apprentissage machine qui prédit les périodes de début et de fin prometteuses pour les segments de tâche et filtre les limites de segments peu probables avant l’énumération des quarts de travail. Des expériences numériques menées sur des ensembles de données réels et synthétiques démontrent que la méthode proposée produit des solutions de haute qualité dans des délais de calcul raisonnables. De plus, l’intégration de quarts de travail à tâches multiples permet de réduire les coûts moyens jusqu’à 4.24% par rapport aux horaires limités aux quarts de travail à tâche unique.

Dans l’ensemble, cette thèse démontre que l’intégration de techniques d’apprentissage machine et d’exploration de données à l’optimisation mathématique peut améliorer considérablement l’évolutivité des modèles explicites de planification du personnel basés sur les quarts de travail. Les approches proposées permettent de résoudre efficacement des problèmes de planification à grande échelle dans le secteur de la vente au détail tout en préservant la qualité des solutions, fournissant ainsi des outils pratiques d’aide à la décision pour la gestion des effectifs dans les industries de services.

ABSTRACT

Personnel scheduling problems consist in determining employee work schedules that ensure that staffing requirements are satisfied over a planning horizon. These problems are widely studied for different industries and are particularly important in service industries such as retail, where workforce demand varies significantly throughout the day, making the employment costs crucial. In such environments, schedules are typically constructed from a large number of potential shifts defined by flexible start times, end times, and durations. When these shifts are enumerated explicitly in mathematical programming formulations, the resulting mixed-integer linear programming models can become extremely large and difficult to solve using general-purpose optimization solvers. Consequently, reducing the size of these models while preserving solution quality constitutes a major challenge for large-scale personnel scheduling problems. In this thesis, we investigate how data-driven techniques, including machine learning and data mining, can be used to guide the shift enumeration process and improve the scalability of optimization models for personnel scheduling in the retail sector.

First, we propose a learning-assisted framework to guide the enumeration of shifts in flexible personnel scheduling problems with mono-job shifts. The approach relies on deep learning models trained on previously solved scheduling instances to identify promising time periods for shift start and end times. Instead of enumerating all possible shifts, the method restricts the enumeration process to those starting or ending at the predicted time points, thereby significantly reducing the number of potential shifts considered in the optimization model. Computational experiments conducted on real-life retail instances with up to 10 jobs and 50 employees demonstrated that the proposed approach can accelerate the solution process by an average factor of 47, while keeping the average cost increase limited to 1.28%.

Second, we investigate how historical scheduling decisions can be exploited through data mining techniques to further improve the efficiency of solving large-scale personnel scheduling problems. Since scheduling instances in retail environments are solved repeatedly over time and often exhibit similar demand patterns, we propose a method based on the k -nearest neighbors algorithm to identify historical instances that are similar to a new instance. The shift patterns appearing in the solutions of these historical instances are then used to restrict the enumeration of candidate shifts in the new problem. This approach allows the optimization process to benefit from previously observed solution structures. Computational experiments conducted on instances involving up to 12 jobs and 252 employees show that the proposed method can remove on average more than 85% of decision variables, leading to

an average speed-up factor greater than 15 while maintaining solution costs within approximately 1% of the baseline solution costs.

Finally, we extend the learning-assisted framework to the more complex setting of multi-job shift scheduling. In many retail environments, employees may perform several different jobs during a single shift, which considerably increases the number of possible shift configurations. We develop an explicit set-covering formulation that incorporates both mono-job and multi-job shifts, where each shift may consist of several job segments. To control the resulting combinatorial growth in the number of variables, we employ a machine learning model that predicts promising start and end periods for job segments and filters out unlikely segment boundaries prior to shift enumeration. Computational experiments conducted on both real and synthetic datasets demonstrate that the proposed method produces high-quality solutions within practical computational times. In addition, allowing multi-job shifts leads to average cost reductions of up to 4.24% compared with schedules limited to mono-job shifts.

Overall, this thesis demonstrates that integrating machine learning and data mining techniques with mathematical optimization can significantly improve the scalability of explicit shift-based personnel scheduling models. The proposed approaches enable the efficient solution of large-scale retail scheduling problems while preserving solution quality, thereby providing practical decision-support tools for workforce management in service industries.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
RÉSUMÉ	v
ABSTRACT	vii
LIST OF TABLES	xiii
LIST OF FIGURES	xv
LIST OF SYMBOLS AND ACRONYMS	xvi
LIST OF APPENDICES	xvii
CHAPTER 1 INTRODUCTION	1
1.1 Definitions and Basic Concepts	1
1.1.1 Personnel Scheduling Problem	1
1.1.2 Problem Variants	2
1.1.3 Problem Application Areas	4
1.2 Solution Methods	5
1.2.1 Mathematical Programming and Personnel Scheduling	5
1.2.2 Machine Learning and Data Mining	6
1.3 Research Objectives	8
1.4 Thesis Outline	9
CHAPTER 2 LITERATURE REVIEW	10
2.1 Introduction to Personnel Scheduling Problems	10
2.2 Variations of Personnel Scheduling Problems	12
2.3 Solution Methods for Personnel Scheduling Problems	14
2.3.1 Mathematical Programming Models	14
2.3.2 Mathematical Programming Based Solution Methods	16
2.3.3 Heuristics, Metaheuristics, and Other Approaches	17
2.4 Machine Learning and Data Mining for Combinatorial Optimization	19
CHAPTER 3 STUDY'S ORGANIZATION	24
CHAPTER 4 ARTICLE 1: LEARNING TO ENUMERATE SHIFTS FOR LARGE- SCALE FLEXIBLE PERSONNEL SCHEDULING PROBLEMS	26

4.1	Introduction	26
4.2	Literature review	28
4.2.1	Integer programming models	28
4.2.2	Heuristics and metaheuristics	30
4.2.3	Machine learning application	31
4.3	Problem description and mathematical modeling	33
4.3.1	Problem statement	33
4.3.2	Notation	36
4.3.3	Mathematical model	37
4.4	Methodology	39
4.4.1	Real-life instances and data generation	39
4.4.2	Features and labels	40
4.4.3	Learning strategy	42
4.4.4	Machine learning models	42
4.4.5	Optimization heuristic pipelines	43
4.5	Computational experiments	46
4.5.1	Experimental setup	46
4.5.2	Evaluation of the predictors	46
4.5.3	Evaluation of the optimization heuristics	50
4.5.4	Out of distribution results	53
4.6	Conclusion	55
CHAPTER 5 ARTICLE 2: DATA MINING-DRIVEN SHIFT ENUMERATION FOR ACCELERATING THE SOLUTION OF LARGE-SCALE PERSONNEL SCHEDULING PROBLEMS		57
5.1	Introduction	57
5.2	Related works	59
5.2.1	Learning-based optimization algorithms	59
5.2.2	Personnel scheduling	60
5.3	The flexible personnel scheduling problem in retail	62
5.3.1	Problem statement	62
5.3.2	Notation	64
5.3.3	Mathematical model	65
5.4	Methodology	66
5.4.1	Instance representation	67
5.4.2	Dissimilarity Measure	68

5.4.3	K-Nearest Neighbors Algorithm	70
5.4.4	Optimization Heuristic	71
5.5	Experimental results	72
5.5.1	Data generation	72
5.5.2	Tuning parameter k	74
5.5.3	Performance of MILP- k NN heuristic	75
5.5.4	Robustness analysis of MILP- k NN heuristic	79
5.6	Conclusion	80
CHAPTER 6 ARTICLE 3: A DATA-DRIVEN METAHEURISTIC FOR PERSONNEL SCHEDULING IN THE RETAIL SECTOR CONSIDERING MULTI-JOB SHIFTS		84
6.1	Introduction	84
6.2	Literature Review	87
6.3	A Multi-Job Tour Scheduling Problem	91
6.3.1	Problem Description	91
6.3.2	Notation	92
6.3.3	Mathematical Model	94
6.4	Methodology	95
6.4.1	Step 1: Shift Enumeration	96
6.4.2	Step 2: Data Generation	97
6.4.3	Step 3: Machine Learning Model Development	98
6.4.4	Step 4: Restricted Shift Enumeration	98
6.4.5	Step 5: Optimization with Reduced Shift Set	99
6.5	Experimental Analysis	99
6.5.1	Experimental Setup	100
6.5.2	Datasets and Instances	100
6.5.3	Solution Methods and Evaluation Metrics	101
6.5.4	Calibration of Parameter ζ	102
6.5.5	Evaluation of the Heuristics	104
6.5.6	Robustness Analysis	106
6.6	Conclusion	107
CHAPTER 7 GENERAL DISCUSSION		108
CHAPTER 8 CONCLUSION		112
8.1	Summary of Contributions	112

8.2 Limitations and Future Research Directions	113
APPENDICES	126

LIST OF TABLES

Table 4.1	Variety of candidate shifts for the example of Figure 4.1	36
Table 4.2	Sets and subsets	36
Table 4.3	Parameters	37
Table 4.4	Decision Variables	37
Table 4.5	Number of jobs and employees in the reference instances and number of generated artificial sub-instances from each instance	39
Table 4.6	FCN architecture	43
Table 4.7	CNN architecture	44
Table 4.8	UNet architecture	45
Table 4.9	Cost increase (in %) with respect to best solution cost	52
Table 4.10	Computational times (in seconds)	53
Table 4.11	MILP model size	54
Table 4.12	Results on the out-of-distribution instances	56
Table 5.1	Sets and subsets	64
Table 5.2	Parameters	65
Table 5.3	Decision Variables	65
Table 5.4	Number of jobs and employees in the datasets	72
Table 5.5	MILP model size	78
Table 5.6	Error (in %)	78
Table 5.7	Computational time (in seconds)	80
Table 5.8	Experiments with a 20% coefficient of variation	80
Table 5.9	Experiments with a 30% coefficient of variation	80
Table 6.1	Sets and Subsets	93
Table 6.2	Decision Variables	93
Table 6.3	Parameters	94
Table 6.4	Numbers of jobs and employees in the training datasets	100
Table 6.5	Number of jobs and employees in the out-of-distribution datasets . .	101
Table 6.6	Number of constraints and variable ratios before and after presolve. .	104
Table 6.7	Average computing times (seconds) of Exact-Mono, Exact-Multi, and heuristics.	105
Table 6.8	Cost deviations (%) for Exact-Multi and heuristics relative to Exact- Mono and Exact-Multi.	105
Table 6.9	Performance metrics on out-of-distribution datasets.	107

Table A.1	Dataset 1: 5 jobs, 85 employees	126
Table A.2	Dataset 2: 6 jobs, 95 employees	126
Table A.3	Dataset 3: 7 jobs, 128 employees	127
Table A.4	Dataset 4: 8 jobs, 95 employees	127
Table A.5	Dataset 5: 8 jobs, 119 employees	127
Table A.6	Dataset 6: 9 jobs, 133 employees	127
Table A.7	Dataset 7: 10 jobs, 165 employees	128
Table A.8	Dataset 8: 12 jobs, 190 employees	128
Table B.1	Dataset S1: 2 jobs, 14 employees	130
Table B.2	Dataset S2: 2 jobs, 29 employees	130
Table B.3	Dataset S3: 2 jobs, 31 employees	131
Table B.4	Dataset S4: 2 jobs, 25 employees	131
Table B.5	Dataset S5: 2 jobs, 36 employees	132
Table B.6	Dataset L1: 5 jobs, 85 employees	133
Table B.7	Dataset L2: 6 jobs, 95 employees	133
Table B.8	Dataset L3: 7 jobs, 128 employees	134
Table B.9	Dataset L4: 8 jobs, 95 employees	134
Table B.10	Dataset L5: 8 jobs, 119 employees	134
Table B.11	Dataset L6: 9 jobs, 133 employees	135
Table B.12	Dataset L7: 10 jobs, 165 employees	135
Table B.13	Dataset L8: 12 jobs, 190 employees	135

LIST OF FIGURES

Figure 4.1	Examples of demand curves of two jobs	36
Figure 4.2	The ROC curves of the six predictors	47
Figure 4.3	Sensitivity of predictors to threshold levels (α)	48
Figure 4.4	Sensitivity of optimization heuristics to α	49
Figure 4.5	Heuristic performance summary	55
Figure 5.1	Demand curves of two example jobs	69
Figure 5.2	Employees' schedule assigned to jobs of example 5.1	70
Figure 5.3	Impact of k on problem size, computational time, and solution quality for the instances in Dataset 1 (in the left) and in Dataset 2 (in the right).	82
Figure 5.4	Search for the value of k for one validation instance in Dataset 2 . . .	83

LIST OF SYMBOLS AND ACRONYMS

PSP	Personnel Scheduling Problem
FPSP	Flexible Personnel Scheduling Problem
MILP	Mixed-Integer Linear Programming
ML	Machine Learning
AI	Artificial Intelligence
DM	Data Mining
DNN	Deep Neural Network
FCN	Fully Connected Network
CNN	Convolutional Neural Network
MJSS	Multi-Job Shift Scheduling
MASS	Multi-Activity Shift Scheduling

LIST OF APPENDICES

Appendix A	Detailed results of Chapter 5	126
Appendix B	Detailed Results of Chapter 6	129

CHAPTER 1 INTRODUCTION

Personnel scheduling represents a fundamental challenge in operational management for organizations in diverse industries, where the efficient allocation of human resources is essential for maintaining competitiveness, controlling costs, and ensuring service quality. In large-scale scenarios, these problems involve coordinating numerous employees across varying demands, constraints, and objectives, often leading to computational complexities that traditional methods struggle to handle. This thesis investigates the application of machine learning techniques to enhance the solving of large-scale personnel scheduling problems, with a particular emphasis on flexible scheduling in retail contexts. By leveraging data-driven approaches, we aim to reduce solution times while preserving high-quality outcomes, addressing gaps in scalability and efficiency.

1.1 Definitions and Basic Concepts

This section introduces the foundational concepts underpinning the research, spanning personnel scheduling, mathematical programming formulations tailored to these problems, machine learning and data mining methods, and heuristics/metaheuristics for optimization. These elements form the backbone of the methodologies developed in subsequent chapters.

1.1.1 Personnel Scheduling Problem

Personnel scheduling problem (PSP), often referred to as workforce scheduling or rostering, entails the assignment of employees to specific work periods, tasks, or activities over a defined planning horizon, such as a day, week, or month. The primary goal is to meet operational demands while adhering to constraints like labor regulations, employee availability, skill sets, and preferences, all while optimizing objectives such as minimizing costs or maximizing fairness. In service industries such as retail, where labor is often precarious and part-time contracts are prevalent, the dominant scheduling objective is to minimize labor costs while adequately covering service demand which leads to a tension that makes efficient scheduling both economically critical and organizationally complex.

At its core, PSP balances supply (available workforce) and demand (required coverage for operations). Demand can be aggregated on a daily basis for long-term planning or broken down into time-specific tasks for short-term timetabling. Poor scheduling can result in overstaffing, leading to idle time and increased labor costs, or understaffing, which may cause service dis-

ruptions and lost revenue. Effective scheduling is particularly vital in service-oriented sectors, where labor constitutes a significant portion of expenses and where demand fluctuations are common due to factors like seasonality or unexpected events. Key components include:

1. **Employees:** Individuals characterized by attributes such as skills, contractual hours (e.g., full-time or part-time), availability windows, and personal preferences (e.g., preferred shifts or days off).
2. **Shifts:** Time blocks of work that may include start and end times, breaks, and durations. Shifts can be fixed or flexible, with rules governing minimum and maximum lengths.
3. **Tasks, Activities, and Jobs:** Tasks are discrete, uninterruptible units of work with fixed durations and time windows (e.g., setting up a store display). Activities are ongoing duties with varying demand over time (e.g., cashiering during peak hours). Jobs represent categories of work associated with specific skills or roles; they define the type of work employees can perform and group together related tasks and activities (e.g., cashier, stocker, or customer service associate).
4. **Constraints:** The constraints in PSPs are generally divided into hard and soft constraints. Hard constraints stem from laws or contracts (mandatory, e.g., no shift overlaps or compliance with rest periods), while soft ones weigh schedule quality (desirable, e.g., minimizing overtime, which may incur penalties if violated).
5. **Objectives:** Common goals include cost minimization (wages, overtime premiums, undercoverage penalties), demand coverage maximization, and employee satisfaction through fair distribution of workloads.

Personnel scheduling encompasses a variety of elements that define its scope, variants, and applications. This section delves into these, highlighting how they interrelate in practical contexts and drawing on insights from subsequent chapters, which develop models and methods for specific retail scenarios.

1.1.2 Problem Variants

Different variants of personnel scheduling problems arise from the decomposition of the overall process into interconnected subproblems, each addressing distinct aspects of workforce allocation.

One primary variant is day-off scheduling, also known as rostering or rotating schedules. This determines work and rest days for employees over the planning horizon to meet daily personnel requirements or task-based demands. A roster typically outlines a weekly pattern, which can be cyclic in rotating systems where employees rotate through predefined weeks. This variant ensures balanced workloads and compliance with rest rules, as studied in works like those on nurse rostering.

Shift scheduling focuses on defining shift parameters, such as start/end times and break placements, without assigning them to specific employees. It aims to generate enough shifts to cover demands, considering constraints on shift lengths, types, and break rules. In most sectors, shifts have predetermined start and end times — for example, 8:00–12:00, 13:00–17:00, or 8:00–17:00 — with only a handful of predefined options per day. In retail, by contrast, shifts may begin and end at almost any quarter-hour within operating hours, giving rise to a much larger candidate set. Integration of flexible shifts allows variability to better match fluctuating demands.

Shift assignment allocates predefined shifts to employees, respecting sequence restrictions, preferences, skills, and bounds like minimum rest between shifts. This variant emphasizes individualization, ensuring no shift is double-assigned.

Tour scheduling combines day-off scheduling, shift scheduling, and assignment, creating comprehensive timetables over the horizon. It is holistic but computationally demanding, often requiring integrated optimization.

Extensions include flexible working environments, incorporating overtime, temporary workers, or adjustable shift/break times to handle demand variability. Overtime provides quick flexibility without hiring, though fatigue risks apply. Temporary workers buffer fluctuations, often sourced from agencies, but may cost more than overtime. Flexible hours allow shifts to adapt, increasing availability during peaks.

Another extension is preference satisfaction, prioritizing employee contentment through incorporating preferences for shifts, jobs, or days off. This enhances productivity and retention, linking to work-life balance.

In this thesis, the PSP is studied in a large-scale retail context that considers shift scheduling and shift assignment simultaneously. The problem allows both mono-job and multi-job shifts and incorporates day-off rules, rest requirements, and other labor regulations through constraints. Staffing requirements are defined at the activity level to capture time-varying demand, and employees may perform different jobs within a shift subject to eligibility conditions. This problem structure reflects practical retail operations but leads to a very large

set of feasible schedules. The main objective of the thesis is to address the resulting computational challenges, which motivates the use of machine learning, data mining, and hybrid optimization approaches presented in the following chapters.

1.1.3 Problem Application Areas

PSPs arise in a wide variety of organizations, including hospitals, airlines, call centers, retailers, and many others. The specific characteristics of these problems depend strongly on the nature of the organization.

Healthcare, particularly nurse rostering, ensures round-the-clock coverage while respecting skills and fatigue limits, impacting patient care quality. In hospitals, for example, patient care must be provided continuously, 24 hours a day and 7 days a week. Nurses and other staff typically work fixed-length shifts of 8 or 12 hours, with starting times usually restricted to a small number of predetermined slots. In such settings, personnel scheduling primarily involves determining the appropriate number of employees to assign to each predefined shift, ensuring that the minimum number of staff required to meet patient demand is satisfied.

Transportation sectors like airlines, railways, and public transit agencies schedule crews to cover flights, trains or bus trips, considering regulations on rest and duty hours. Manufacturing and call centers manage shifts to match production or call volumes, minimizing downtime.

Importantly, the focus of this thesis is strictly on stationary PSPs — that is, problems in which employees are assigned to fixed workplace locations (such as stores, hospitals, or call centers) and do not travel between tasks or sites during their shift. We therefore deliberately exclude crew scheduling and driver scheduling problems (common in airlines, railways, urban transit, and long-haul transportation), where employees (pilots, flight attendants, train drivers, bus drivers, etc.) travel to different locations to perform their duties, and the scheduling must additionally account for crew pairing, vehicle routing, legal rest requirements during travel, and recovery from disruptions.

Large retail stores and many service-oriented industries optimize their operations amid variable customer flows, covering cashiering, stocking, and promotions. Flexible multi-job shifts allow employees to switch jobs, reducing idle time. While this arrangement benefits operational efficiency, it also has implications for working conditions: employees may face increased cognitive load from switching roles, variable hourly patterns, and limited predictability in their schedules — factors that are relevant when evaluating the broader impact of scheduling policies. These industries operate only during fixed business hours, and customer demand

exhibits substantial variability — often changing significantly from hour to hour and from day to day. In these environments, shifts are much more flexible: they can begin and end at almost any time during operating hours, and their durations can vary widely (typically between 4 and 9 hours). Rather than working with only a few predefined shift templates (as in traditional nurse rostering, where there may be only 3–5 possible shifts per day), retailers must consider a very large number of potential shifts — often more than 900 per day when the operating window spans multiple days. Shifts can start and end every 15 minutes, and durations vary flexibly. Moreover, demand in these contexts is not defined per shift, but per time period (typically every 15 minutes), requiring the schedule to closely track hourly or even sub-hourly fluctuations.

In the remainder of this thesis, we refer to the class of problems we address — characterized by flexible start/end times, variable durations, period-based demand coverage, and stationary work locations — as flexible personnel scheduling problems (FPSPs).

1.2 Solution Methods

PSPs are inherently combinatorial, as the number of possible assignments grows exponentially with the scale of employees and time periods. This complexity necessitates advanced solution techniques, as introduced in the following sections.

1.2.1 Mathematical Programming and Personnel Scheduling

Within the realm of personnel scheduling, mathematical programming provides a structured framework for modeling and solving these challenges. These problems are typically formulated as optimization models, where decisions (e.g., shift assignments) are represented by variables, constraints ensure feasibility, and an objective function guides the search for optimal solutions. Given the discrete nature of assignments, mixed-integer linear programming (MILP) is a prevalent approach, as it combines continuous and integer variables to capture real-world nuances. In personnel scheduling, MILP models often adopt set-covering or set-partitioning formulations. In a set-covering model, the problem is viewed as selecting a subset of predefined "sets" (e.g., possible shifts or rosters) that cover all required demands at minimal cost. Each set represents a feasible assignment for an employee, including coverage for specific time periods or tasks. The objective is to minimize the total cost while ensuring every demand element (e.g., a time slot's staffing need) is covered at least once. These formulations are encapsulated within personnel scheduling by generating a large pool of candidate shifts or rosters a priori, then selecting an optimal combination. However, for

large-scale problems, the enumeration of all possible sets is computationally prohibitive due to the NP-hard nature of MILPs. To mitigate this combinatorial explosion, several advanced optimization techniques have been proposed in the literature. Decomposition methods, such as column generation and branch-and-price, avoid full enumeration by generating promising shifts or rosters dynamically as needed. In these approaches, a restricted master problem selects among a limited subset of shifts, while pricing subproblems generate new shifts with negative reduced cost. Although highly effective, these exact methods remain computationally demanding when applied to large-scale, real-world PSPs characterized by complex rules, multi-skill environments, and fine-grained time discretizations. The resulting pricing problems are often themselves NP-hard, and their repeated solution can dominate computational effort.

Consequently, purely optimization-based approaches face scalability limitations as problem size and structural complexity increase. This challenge is particularly pronounced in retail and service-sector settings, where demand variability and flexible job assignments dramatically expand the space of candidate shifts, while labor regulations simultaneously reduce this set by filtering out configurations that violate rest-period requirements, minimum and maximum shift durations, and other contractual constraints — leaving a large but bounded feasible space that nonetheless remains computationally prohibitive to enumerate exhaustively. Heuristics and metaheuristics play a crucial role in solving large-scale optimization problems, such as those encountered in personnel scheduling, where exact methods like MILP become intractable due to exponential computational requirements. These approaches prioritize finding good, feasible solutions in reasonable time frames, often trading optimality guarantees for efficiency. In contrast to end-to-end generative methods, there is growing interest in complementary methodologies that can reduce problem size, guide the search toward high-quality solutions, or approximate key decision components without exhaustive enumeration. In recent years, machine learning and data mining have attracted increasing attention in combinatorial optimization research, particularly as complementary approaches to mathematical programming for large-scale problems.

The following subsection, therefore, introduces machine learning and data mining techniques and discusses how they can be leveraged to address the scalability challenges inherent in mathematically programmed personnel scheduling problems.

1.2.2 Machine Learning and Data Mining

Machine learning (ML) and data mining (DM) are pivotal in addressing the scalability issues of personnel scheduling by learning patterns from data to inform decision-making. ML

involves algorithms that improve performance on tasks through experience, typically by training on datasets to make predictions or decisions without explicit programming.

Core methods include supervised learning, where models learn from labeled data (e.g., input features like demand curves paired with optimal shifts) to predict outcomes; unsupervised learning, which identifies patterns in unlabeled data (e.g., clustering similar scheduling instances); and reinforcement learning, where agents learn optimal actions through trial-and-error rewards. Supervised learning is central to this thesis, using techniques like regression for continuous predictions or classification for discrete ones.

Deep neural networks (DNNs) extend these techniques by stacking layers of interconnected nodes (neurons) to process complex data. A DNN is designed with an input layer (receiving features), hidden layers (performing transformations via weights and activation functions like ReLU), and an output layer (producing predictions). Training involves forward propagation to compute outputs, backpropagation to adjust weights using gradients, and optimization via algorithms like Adam to minimize loss functions (e.g., mean squared error). Architectures such as fully connected networks (FCNs), convolutional neural networks (CNNs) for grid-like data (e.g., time-series demands), and U-Nets for segmentation tasks are utilized in Chapters 4 and 6 to predict effective shifts.

DM complements ML by extracting knowledge from large datasets, involving processes like association rule mining, clustering, and anomaly detection. A key aspect is data preprocessing, which ensures quality and suitability for models. This includes data scaling (e.g., normalization to $[0,1]$ or standardization to zero mean and unit variance) to prevent feature dominance; transformations (e.g., logarithmic for skewed data or one-hot encoding for categorical variables); handling missing values; and feature selection to reduce dimensionality. Preprocessing is crucial, as poor data quality can lead to biased or inaccurate models, impacting the reliability of scheduling predictions. These concepts enable hybrid approaches, where ML predicts subsets of high-quality shifts to feed into MILP solvers, accelerating solutions for large-scale problems as explored in the thesis.

PSPs are inherently recurrent, as similar planning decisions are made repeatedly over comparable time horizons (e.g., weekly or monthly schedules) under related operational conditions. In many practical settings, demand patterns, feasible shift structures, labor rules, and workforce characteristics evolve slowly over time, leading to recurring solution structures across instances. As a result, historical schedules and demand realizations contain valuable information about which shifts are frequently selected, how activities are covered, and how constraints interact in optimal or near-optimal solutions. This historical information can be leveraged to anticipate effective scheduling decisions in future periods, enabling the prediction

of promising shifts or coverage patterns before solving a new instance.

This observation motivates the integration of data-driven techniques into personnel scheduling. Rather than relying solely on mathematical programming to explore the entire solution space, machine learning and data mining methods can exploit historical data and structural regularities to identify promising shifts, patterns of demand coverage, or decision rules that narrow the search space a priori. Such approaches offer a scalable means to enhance traditional optimization models by providing informed inputs, warm starts, or reduced formulations.

1.3 Research Objectives

The primary objective of this research is to develop efficient methodologies integrating machine learning and data mining with mixed-integer linear programming to solve large-scale FPSPs, particularly in retail settings with flexible and multi-job shifts. By addressing the computational burden of shift enumeration in set-covering MILP models, the work aims to achieve significant speedups while maintaining near-optimal solution quality. Specific objectives include: (1) Designing deep learning models, such as deep neural networks, to predict promising shift start and end times from historical data, reducing MILP variables for mono-job variants; (2) Employing data mining techniques like k-nearest neighbors to enumerate shifts based on instance similarity, enhancing robustness across datasets; (3) Extending the first approach to multi-job shifts, incorporating segment predictions and hybrid. The methods are evaluated on real and synthetic retail instances, measuring metrics such as cost deviations, runtime, and variable reductions. The generalizability of our approaches is analyzed through out-of-distribution testing to ensure applicability in diverse scenarios. Ultimately, this thesis seeks to bridge the gap between academic optimization and practical software solutions, providing tools that enable faster, cost-effective scheduling for industries facing dynamic demands. An important consideration when developing data-driven approaches is the risk of reproducing inefficient practices embedded in historical data. If past schedules reflect suboptimal or intuitively motivated but ineffective decisions, learning from them risks perpetuating such patterns. The proposed methods mitigate this risk by relying on near-optimal historical solutions obtained from a MILP solver, ensuring that the learned patterns correspond to high-quality scheduling decisions rather than ad hoc practices.

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 reviews the relevant literature related to personnel scheduling and the methodological foundations of this work. Chapter 3 briefly describes the overall organization of the three main research chapters. Chapter 4 addresses the first research objective and corresponds to an article published in *Journal of Scheduling*. Chapter 5 focuses on the second research objective and presents an article published in *ACM Transactions on Evolutionary Learning*. Chapter 6 is devoted to the third research objective and presents an article submitted for possible publication in *INFOR: Information Systems and Operational Research*. Chapter 7 provides a general discussion of the results and contributions of the thesis. Finally, Chapter 8 summarizes the main conclusions and outlines directions for future research.

CHAPTER 2 LITERATURE REVIEW

Personnel scheduling problems have been an important subject of research for several decades and remain central to workforce management in many industries. Modern organizations, particularly in the service and retail sectors, must simultaneously cope with fluctuating demand, heterogeneous staff skills, complex labor regulations, and service-level requirements. This environment creates large-scale, highly constrained optimization problems where classical mathematical programming approaches can struggle to deliver high-quality solutions within acceptable computational times.

At the same time, the recent progress in computing power and data gathering capabilities has opened new opportunities to combine traditional optimization methods with machine learning (ML) and data mining techniques. These developments enable the extraction of structural patterns from historical data or previously solved instances, and their integration into optimization models to reduce problem size or guide search procedures. In this chapter, we review the literature on personnel scheduling problems, their main variants, their mathematical formulations and solution methods—including explicit and implicit models, heuristic and exact approaches—and the contributions of ML and data mining to large-scale optimization in general and personnel scheduling in particular.

2.1 Introduction to Personnel Scheduling Problems

Personnel scheduling problems were introduced by Edie [1] to decrease vehicles' waiting time for service at toll booths with the minimum number of toll collectors. Since then, different types of problems have arisen for various application areas. Among the most popular research areas on personnel scheduling problems, Özder et al. [2] point to health services, manufacturers, and call centers. However, the retail sector has been less explored.

Most studies aim to determine schedules for employees at the lowest cost. There are usually three components to the cost of a solution: (1) the cost of assigning employees to jobs; (2) the penalty associated with over-coverage; and (3) the cost resulting from under-coverage, which results in a degradation of the service quality. For generating employee schedules, mixed-integer programming-based methods are the most popular approaches in the scientific literature.

Mathematical models for personnel scheduling problems can be divided into either explicit set-covering or implicit formulations. In the explicit models, one integer variable is associated

with each shift type where the type of a shift is defined by all its characteristics such as start time, end time, break time, job type, etc. In implicit models, some aspects of the shifts are determined by means of constraints. One example of implicit modeling would be to define shifts solely by their starting times and add additional constraints to the model to limit their lengths. In many cases, the number of decision variables is considerably smaller than for an explicit formulation. According to Thompson [3], in most cases, it is hard to express the employment cost using an implicit formulation. Explicit models are preferable when the cost of a shift is defined by its start time, end time, length, and placement of the breaks. Indeed, when dealing with a heterogeneous workforce, explicit approaches are predominantly employed in the literature. Implicit approaches, on the other hand, primarily focus on establishing anonymous work schedules, where individual employee characteristics are disregarded. However, when these characteristics exhibit significant diversity, the allocation of anonymous shifts to employees becomes a complex and non-trivial task. Furthermore, Quimper and Rousseau [4] points out that it is not possible to formulate multiple-job problems through implicit modeling.

The first explicit set-covering model for personnel scheduling was proposed by Dantzig [5] for the problem of assigning employees to toll booths. In this formulation, all feasible working patterns are enumerated for all possible start times, end times, and breaks. Then, one integer variable x_s is associated with each feasible work pattern s . Here, the problem objective is to find the shift assignment of the minimum total cost that covers the demand for open toll booths at each time interval.

Explicit modeling allows one to formulate different aspects of personnel scheduling problems. To give a few examples, explicit formulations have been used to produce employee schedules for nurse scheduling, for tour scheduling, for personnel scheduling with employee transfers between departments, for a multi-job multi-task shift scheduling problem, and for a stochastic personnel scheduling problem with demand uncertainty. The set-covering formulation provides the advantage of modeling flexibility. Nevertheless, additional options for placement of breaks, start times, lengths, and end times for shifts result in a larger number of possible shifts and consequently larger models. For example, in the retail industry, shifts may start and end every quarter of an hour during a day. Enumerating all possible shifts for all available employees and for multiple jobs results in a very large model that requires efficient algorithms to be solved. These recurrent problems need to be solved in a reasonable amount of time frequently by efficient algorithms.

Commercial solvers fail to achieve near-optimal solutions in a reasonable time for large-sized problems either formulated explicitly or implicitly. In the literature on personnel scheduling

problems, mostly decomposition methods and heuristic algorithms are implemented to solve large-scale personnel scheduling problems. However, practical personnel scheduling problems encounter many shift feasibility constraints and many complicated ways to compute a shift's cost. In this case, it becomes difficult to design the pricing problems that are used to generate shifts dynamically in a column generation algorithm. To alleviate this, some authors use context-free grammar to model these shift features.

To the best of our knowledge, the studies in personnel scheduling that aim at reducing the number of variables in a set-covering model are limited to very early works such as the studies of Bechtold and Brusco [6], Henderson and Berry, [7], and Mehrotra and Trick [8]. Their methods, called working set generation, select a subset of schedules from all the possible work schedules and create only schedule variables for this subset. Different procedures have been developed to determine this subset. The one that seems to be the most efficient has been proposed by Bechtold and Brusco [6] and builds the work schedule subset in three steps: first, it computes a subset of days-off configurations based on the coverage that they offer; second, it finds for each day a subset of shifts based, again, on the coverage that these shifts offer; and, finally, it enumerates all work schedules resulting from a combination of the selected days-off configurations and shifts. Another study by Brusco et al. [9] performs experimental tests to find out how the number of required employees is affected if they restrict the shifts' starting periods to only 3, 4, and 5 periods of the day instead of 24 periods. These works cannot be directly applied to our context as they assume that there is a single job to cover and that the employees are available all the time and can, thus, be considered identical.

2.2 Variations of Personnel Scheduling Problems

Starting from Esie [1], personnel scheduling problems have been considered an interesting subject of research for the past decades. Comprehensive literature reviews for this subject have been written by Ernst et al. [10] and van den Bergh et al. [11], among others. Authors, including Baker [12] classify the personnel scheduling problems into the following categories:

- **Day-off:** Determines working days and days off for employees during the planning horizon. The term “rostering” is also used in this context.
- **Shift Scheduling:** Specifies the start time, end time, and breaks of the shifts with the objective of demand satisfaction at each time interval.
- **Tour Scheduling:** Integrates day-off, shift scheduling, and assigning shifts to employees during the planning horizon.

- **Multi-Job Shift Scheduling:** Schedules shifts, and assigns jobs to the employees, specifying which job must be performed at which time interval in each shift.

Key variants include mono-job shift scheduling, where employees perform a single role throughout a shift, often modeled with fixed time slots and coverage constraints. Multi-job shift scheduling extends this by allowing employees to switch between multiple roles within a shift, enhancing adaptability to fluctuating demands. Related are multi-task shift scheduling, which emphasizes task sequencing within shifts, and multi-department scheduling, involving cross-departmental assignments. General shift scheduling covers broader rostering over planning horizons, while activity assignment focuses on granular task allocation without fixed shifts. Other extensions include tour scheduling (combining shift and days-off decisions) and self-scheduling (incorporating employee preferences).

Common challenges across variants involve balancing conflicting constraints, such as skill matching, fairness in workload distribution, rest periods, and cost minimization, often compounded by uncertainty in demand or availability. Multi-job variants are increasingly relevant in modern industries like retail and healthcare, where workforce flexibility can reduce over-staffing and improve efficiency, as highlighted in recent reviews. For instance, in retail settings with variable customer flows, allowing employees to handle multiple jobs (e.g., cashiering and stocking) mitigates idle time and aligns with the thesis’s focus on large-scale multi-job shifts. However, this added flexibility exacerbates computational complexity, underscoring the need for advanced formulations and solution methods [2, 11, 13].

Multi-job shift scheduling (MJSS) involves assigning employees to shifts where they perform multiple job types, subject to skill qualifications, demand coverage, and operational constraints. Key elements include multiple job types per shift, skill matching to ensure employees are qualified for all assigned jobs, potential shift overlaps for seamless transitions, and objectives like minimizing labor costs, over-coverage penalties, or maximizing coverage. Unlike mono-job variants, MJSS allows intra-shift job switches, which is particularly useful in environments with time-varying demands, such as retail stores where employees might alternate between sales and inventory tasks. The problem is typically defined over a planning horizon (e.g., a week) divided into discrete periods, with demands specified per job and period [14].

Common formulations employ mixed-integer linear programming (MILP) to capture the combinatorial nature of MJSS. Set-covering approaches are prevalent, where shifts are pre-enumerated as decision variables covering demands, incorporating constraints on shift durations, rest rules, and employee availability. For example, variables might represent shift assignments to employees and jobs, with constraints ensuring contiguous segments for each

job and minimum segment lengths (e.g., one hour). Network flow models represent shifts as paths in graphs, facilitating decomposition, while column generation dynamically generates shifts to handle large sets. Variations include stochastic elements for demand uncertainty or non-linear objectives for fairness. The MILP models in literature exemplify this, using binary variables for shift assignments and stepwise functions for costs, highlighting the exponential growth in variables due to multi-job combinations [15–17].

Exact methods for MJSS often rely on branch-and-bound or branch-and-price, integrated with commercial solvers like CPLEX, to solve MILP formulations. For scalability, heuristics such as genetic algorithms evolve shift assignments, while simulated annealing perturbs solutions to escape local optima. Hybrid approaches combine exact optimization with metaheuristics, like matheuristics using Lagrangian relaxation to decompose problems. Large neighborhood search iteratively destroys and repairs schedules, proving effective for multi-activity variants. Under uncertainty, stochastic programming or robust optimization adjusts for variability [4, 18–20].

2.3 Solution Methods for Personnel Scheduling Problems

This section reviews the main solution approaches for personnel scheduling problems, classified into mathematical programming formulations (explicit and implicit models), exact methods, heuristics and metaheuristics, and other advanced strategies.

2.3.1 Mathematical Programming Models

Mathematical models for personnel scheduling problems can be divided into either explicit set-covering or implicit formulations. In the explicit models, one integer variable is associated with each shift type where the type of a shift is defined by all its characteristics, such as start time, end time, break time, job type, etc.

As mentioned in the previous sections, the first explicit set-covering model for personnel scheduling was proposed by Dantzig [5], which allows one to formulate different aspects of personnel scheduling problems. Mehrotra and Trick [8] extended Dantzig’s model, considering different types of breaks and break windows. Caprara et al. [21] enumerated all working patterns (sequences of working days and days-off) explicitly and designed a two-phase method for balanced shift scheduling and task assignment. Bard and Purnomo [22] considered personnel preferences for nurse scheduling and provided a multi-objective problem. The research in [23] addressed the under-coverings by assigning less competent nurses rather than costly outsourced nurses in case of shortage in staff. They also improved the regular staff satisfac-

tion of assigned shifts. Beliën et al. [24] combined nurse and operating rooms scheduling through modifying the right-hand side value of the demand satisfaction constraint. In the new constraint, the right-hand side, instead of being a constant depends on the number of patients' surgery types. Brunner and Stolletz [25] considered flexibility of the breaks for a tour scheduling problem using a set-covering formulation. Attia et al. [26] and Dahmen et al. [27] used set-covering modeling to formulate a shift scheduling problem where the transfer of employees between departments is allowed.

By using explicit enumeration, it is straightforward to generate all feasible shifts that satisfy rules and constraints for each individual shift. Nevertheless, additional options for placement of breaks, start times, lengths, and end times for shifts result in a larger number of possible shifts and consequently larger models. Solving a set-covering model with a large number of explicitly defined variables is not easy.

To avoid handling the enormous number of variables in explicit set-covering models for personnel scheduling, researchers have reformulated the model by defining decision variables implicitly. For example, in one implicit modeling approach, the type of shift is defined by its starting and ending times. Other sets of variables are needed to handle the breaks. Once the number of employees assigned to each shift type and the breaks are determined, a post-processing algorithm handles the creation of the shifts that include breaks. In many cases, the number of decision variables is considerably less than for an explicit formulation. The first implicit formulation was introduced by Moondra [28] with an application to banking. Researches conducting this approach continued with the study in [29, 30], [3], [31, 32], and [33, 34]. In these works, shift types and breaks are not combined at first.

While improvements have been remarkable with respect to the advancement of implicit formulations, they fail to model some attributes of rich personnel scheduling problems. When feasible shifts follow some complex rules, the modeling of the problem using an implicit formulation is much more difficult than an explicit one. According to Thompson [3], in most cases, it is hard to express the employment cost using an implicit formulation. Explicit models are preferable when the cost of a shift is defined by its start time, end time, length, and placement of the breaks. Furthermore, Quimper and Rousseau [4] pointed out that it is not possible to formulate multiple-job problems through implicit modeling.

Côté et al. [35, 36] developed integer programming models for shift scheduling problems using context-free grammars to define shifts explicitly and implicitly, respectively. Having implemented the same method, Boyer et al. [37] contributed both interruptible jobs and uninterruptible tasks in a multi-job multi-task shift scheduling problem. A generalization of these researches has been done by Restrepo et al. [38] for a tour scheduling problem which

combines day-off and shift scheduling.

2.3.2 Mathematical Programming Based Solution Methods

Commercial solvers fail to achieve near-optimal solutions in a reasonable time for large-scale problems, either formulated explicitly or implicitly. In the literature on personnel scheduling problems, mostly decomposition methods and heuristic algorithms are implemented to solve large-scale personnel scheduling problems [11].

Mehrotra and Trick [8] developed a branch-and-price-based method that generates a small fraction of columns and implements specialized branching rules to implicitly optimize the linear relaxation of the set-covering model. Bard et al. [22, 23] implemented heuristic column generation to solve the set-covering problem for nurse scheduling. Ni and Abeledo [39] proposed to decompose the variables of a tour scheduling problem into shifts and start times. This idea was used to design an efficient branch-and-price algorithm. Beliën and E. Demeulemeester [24] employed the branch-and-price technique for scheduling nurse resources and surgery room resources of a hospital at the same time. Another research was performed by them [40] to study branch-and-price by decomposing the personnel scheduling problem into the personnel and the jobs. Brunner and Stolletz [25] applied a branch-and-price algorithm for a tour scheduling problem using stabilization strategies to accelerate the convergence of column generation.

The studies in [41–43] studied variants of multi-job shift scheduling problems where the shifts are assigned in advance. Column generation heuristic methods were developed to solve the large-scale task assignment problems. Côté et al. [44] extended the research of Lequy et al. in [41] integrating shift scheduling and task assignment. They formulated the model using a context-free grammar method and implemented a branch-and-price algorithm to solve large problems. In the same context, Boyer et al. [37] proposed a branch-and-price algorithm for a multi-job multi-task shift scheduling problem. The tour scheduling version of this problem is solved by Restrepo [38] through a combination of Benders decomposition and column generation. Legrain et al. [45] developed an integer programming model for a nurse scheduling problem in which each variable represents a rotation and constraints are divided into hard and soft constraints. They proposed a branch-and-price algorithm embedded in a large neighborhood search method.

Besides column generation methods, researchers have also investigated implicit enumeration and cutting plane methods to tackle large-scale personnel scheduling problems. Defraeye and Van Nieuwenhuyse, [46] developed a branch-and-bound algorithm for stochastic shift scheduling with non-stationary demand and arrival rates. Felici and Gentile [47] proposed

valid inequalities and symmetry-breaking branching rules for a personnel rostering problem. At Kronos, practical personnel scheduling problems encounter many shift feasibility constraints and many complicated ways to compute a shift’s cost. In this case it becomes difficult to design sub-problems in a column generation algorithm to generate shifts. They enumerate the set of all feasible shifts, employ an explicit modelling, and solve them by a commercial MIP solver. Accordingly, despite the effective performance of the aforementioned solution methods, they are not applicable to our problem. Instead, we investigate machine learning techniques to reduce the number of shifts and, subsequently, the problem size.

2.3.3 Heuristics, Metaheuristics, and Other Approaches

Most heuristics and metaheuristics for personnel scheduling problems are developed to address complexities resulting from numerous and complicated constraints and not from the flexibility in shift enumeration. Such issues arise mostly in the healthcare domain where the preferences of the staff are important but the possible working shifts are very limited (3 to 5 per day). Generally, in these problems, the constraints are categorized into two groups: hard and soft constraints, depending on the legal regulations and personal preferences of the individual institutions and countries. Hard constraints must be met in order to obtain feasible solutions, whereas soft constraints may be violated with penalties. Burke et al. [48] review the state-of-the-art heuristics and metaheuristics such as simulated annealing, tabu search, and genetic algorithms for nurse rostering problems. Among the more recent research, we can refer to the work of [22], [49], and [50].

Simulated annealing, genetic algorithms, tabu search, and working subset heuristics have been successfully applied to various personnel rostering and scheduling settings [6, 9, 51–53]. A few papers develop heuristics for flexible personnel scheduling problems in the domain of service and retail industry. the research Attia et al. [26] and Dahmen and Rekik [27] focus on large-scale multi-department problems. In the former work, a large neighborhood search heuristic is proposed, where the neighborhoods are explored using a MILP model. In the latter, a two-stage solution method is devised. In the first stage, an aggregated MILP model that considers a coarser discretization of the time horizon for the demand curves is solved to yield an approximate solution. In the second stage, this solution is disaggregated by solving one or several small MILP models restricted to a small subset of shifts.

Dahmen and Rekik [17] consider a similar environment to our problem. Assuming that the employees’ days off are known, they introduce a hybrid heuristic that combines tabu search and branch-and-bound to perform a local search. This algorithm requires average computational times exceeding two hours for instances with 20 to 30 employees and 2 to 4 jobs. Very

recently, Hassani et al. [54] developed a Parallel Ruin and Recreate (PRR) heuristic that generates and assigns shifts during the solution process. A set of generator decisions (e.g., replace/extend/shorten a shift) forms the basic moves that lead to new feasible and infeasible solutions. At each iteration, PRR stimulates a well-targeted disruption (shift shortening) before correcting it with a sequence of basic moves that is computed by a truncated depth-first search. Furthermore, to improve efficiency, the search is performed in parallel by defining at each iteration multiple subproblems, each initiated by a disruption that can be corrected by a selected subset of employees.

Römer [15] introduced novel mixed-integer linear programming formulations for multi-activity shift scheduling problems (MASSP) using block-based state-expanded networks to encode shift feasibility rules, with techniques like aggregate side constraints and decoupling of shift composition from activity assignments to significantly reduce model size and symmetry. Through computational experiments on two literature benchmark sets, the approach solves all instances—including over 70 previously open ones—to optimality on a notebook computer, with most requiring less than 30 minutes.

Hernández-Leandr et al. [18] proposed a matheuristic for the personalized MASSP, where Lagrangian relaxation identifies promising shifts modeled via context-free grammars, and a restricted set-covering problem is solved to obtain feasible assignments that minimize under- and over-covering costs while respecting employee qualifications and activity time windows. Computational tests on Demassey and Lequy benchmark instances show the method achieves optimality gaps below 0.01% for 70% of the former and improves or matches best-known solutions for over 92% of the latter, often in significantly less time than prior branch-and-price approaches.

Dahmen and Rekik [17] addressed a multi-activity multi-day shift scheduling problem in a continuous and flexible environment with a heterogeneous workforce, aiming to minimize under-staffing and over-staffing costs given pre-assigned days-off schedules and a restricted workforce budget. A hybrid heuristic combining tabu search with branch-and-bound is proposed, and computational experiments on 288 generated instances show it provides near-optimal solutions in relatively short times, outperforming CPLEX, which fails to find feasible solutions for over 77% of the cases.

Quimper and Rousseau [4] presented a large neighborhood search approach for the multi-activity shift scheduling problem, enhancing formal languages such as regular and context-free grammars to model complex shift construction regulations and derive specialized graph structures for efficient searching. The method yields near-optimal solutions comparable or superior to mixed-integer programming solvers on single-activity benchmarks and scales effectively to

large instances with up to 10 activities.

Boyer et al. [37] introduced a branch-and-price algorithm for solving the personalized multi-activity multi-task shift scheduling problem, where interruptible activities and uninterruptible tasks with precedence constraints are assigned to heterogeneous employees to meet demand while respecting individual qualifications, preferences, and availabilities, with pricing subproblems modeled using context-free grammars to encode complex shift rules. Computational experiments on large instances inspired by real cases show the approach’s flexibility in handling various problem variants effectively. This work is directly relevant to the present thesis because it addresses a class of multi-job shift scheduling problems where the combinatorial structure of feasible shifts is explicitly modeled — a modeling choice that motivates our use of explicit set-covering formulations and highlights the scalability challenges we aim to address through ML-guided shift enumeration.

2.4 Machine Learning and Data Mining for Combinatorial Optimization

Formal foundations of Artificial Intelligence (AI) in the form of an academic discipline emerged in a 1956 conference. In the late 1950s, the term “machine learning (ML)” was dedicated to a subset of AI that addresses the prediction process through learning from data. Initial machine learning models enumerated all possibilities and designed a decision tree to make choices. Nowadays, ML provides more flexible models and enables the computer to make choices without exploring the entire decision tree.

The machine learning process starts by feeding large amounts of input data to the machine, and the primary predictions are calculated by it. Next, these predictions are tuned by comparing them with the expected results. These input and output data are called training data. The main goal is to generalize the model based on the training data so the model can act automatically on the unseen data.

There are mainly two general forms for machine learning:

- **Supervised learning:** Samples of inputs and the resulting outputs are given to the machine. Therefore the goal is to obtain a rule or pattern that processes the input data and automatically predicts the appropriate outputs.
- **Unsupervised learning:** The machine determines relations or patterns in the input data with no anticipation of the resulting outputs. The aim is to analyze data in order to determine unknown patterns.

In recent years, the computing power and data collection advancements have improved the

efficiency of AI and ML algorithms. Accordingly, there has been a surge of interest in seeking advantage of these technologies by a vast area of industries and research fields including combinatorial optimization problems. A survey on the contribution of ML to combinatorial optimization problems is presented by the study in [55]. The research in [56] summarized the previous works on this subject in the following five paths, and we also added the most recent papers:

1. To improve the performance of an optimization solver by predicting if a reformulation is required for the problem before being solved by a MIP solver. For example, the study in [57] employed ML to predict whether a Dantzig–Wolf decomposition is required for a mixed-integer programming (MIP) problem before solving.
2. To improve the performance of solution methods. For instance, machine learning enhances algorithms performance by finding good initial feasible solutions [58] and accelerates enumerative algorithms in variable selection [59]. ML is also employed to facilitate column selection in column generation algorithms [60] and to select cutting planes [61].
3. To reduce the computational complexity of solving optimization models under uncertainty by implementing dimensionality reduction methods [62].
4. To generate the solutions (ideally optimal) of a combinatorial optimization problem [56] or to predict the value of a subset of decision variables [58, 63].
5. To predict the objective value (ideally optimal) of a problem [64].

Data mining and knowledge extraction have the potential to enhance optimization techniques by leveraging historical data, patterns, and insights. Researchers have applied data mining techniques such as clustering and classification to discover patterns and relationships within the input data of optimization problems, leading to more efficient search strategies and improved decision-making [65]. Such ideas have been largely explored, especially in the metaheuristic community, aiming to create new solutions from good fragments (or blocks) of high quality solutions previously found and stored during the search. In fact, recognizing recurring patterns that often emerge in a collection of solutions can be effectively cast as a frequent pattern mining task [66]. In [67], frequent patterns are mined by the FPmax* algorithm of [68] and used for solution re-initialization within the hybrid genetic algorithm of [69], thus guiding the search through more promising regions in the solution space. In [70], the FPmax* algorithm is also used to mine patterns from the population of a genetic-based search, but is used to guide, in this case, the offspring solution construction. By

integrating mathematical optimization with an evolutionary genetic approach, the study in [71] introduces a “reduce-solve-combine” memetic search method. In each generation, a common pattern mined from two parent solutions is first used to reduce the original instance. The reduced instance is then solved, and an offspring solution is finally obtained by combining the mined common pattern with the solution of the reduced instance.

The author of [72] provides a comprehensive taxonomy of how data mining/ML can be integrated into metaheuristics to improve optimization outcomes. Notably, the survey highlights the use of data mining/ML techniques for problem decomposition, which is especially relevant for tackling complex and large-scale problems. The author discusses various supported approaches for breaking down optimization problems into smaller, more manageable subproblems, which can be solved more efficiently.

Three primary decomposition strategies are commonly employed: (1) *data space decomposition*, where input data is partitioned (e.g., using clustering techniques) to form smaller subproblems which are then solved and their solutions combined to form a final complete solution [see, e.g., 73–75]; (2) *decision space decomposition*, which divides decision variables into subsets, reducing interdependencies and enabling focused problem-solving; this approach relates with the frequent pattern mining methods previously discussed; and (3) *objective space decomposition*, particularly useful in multi-objective optimization, where data mining/ML reduces or clusters objectives, thus increasing the efficiency of computing dominance relations, and guiding the search process [see, e.g., 76–78].

In recent years, the mathematical optimization community has been increasingly interested, as well, in using data mining and ML techniques to improve the performance of mathematical programming solvers. The study in [55] provide a survey of the potential contribution of ML to the combinatorial optimization community. ML methodologies prove effective in addressing extensive continuous and combinatorial problems by acquiring and leveraging problem structures [see, e.g., 57], predicting optimal solutions through the analysis of data extracted from previously solved instances [see, e.g., 58, 63], reducing problem sizes [see, e.g., 62, 79–81], and enhancing the performance of established optimization algorithms [see, e.g., 59–61, 82].

Recent advances further underscore the evolving integration of ML in optimization. For instance, the 2025 GECCO tutorial on “Evolutionary Computation meets Machine Learning” highlights automated algorithm design for combinatorial problems, aligning with heuristic pruning strategies. Similarly, Dagstuhl’s 2025 seminar on ML-augmented algorithms emphasizes hybrid methods for complex optimizations, resonant with our data mining–MILP fusion. In personnel scheduling, recent data-driven robust flexible scheduling approaches (e.g., 2024

work on satisficing frameworks) introduce robust formulations that complement our recurrent instance handling. These developments affirm the timeliness of our contributions.

We focus on the works of [63] and [58] from the fourth category above, as they share several key ideas with our research. Both studies consider recurrent optimization problems that are repeatedly solved with slightly varying input data using MILP formulations. To accelerate solution times, they leverage machine learning techniques to extract information from previously solved instances and fix the values of selected variables in future instances, thereby reducing the problem size and improving solver performance.

The work of [63] considers a reference combinatorial optimization instance that can be modeled and solved optimally as a MILP. When the input data is perturbed, the problem must be re-solved. The authors construct a database of past perturbations and corresponding solutions, either observed or simulated, and use it to support the resolution of new instances. They formulate the task as a supervised learning problem, where a classifier predicts whether all or part of the optimal solution of the reference instance remains valid for the perturbed instance. If only part of the solution can be reused, regression models are used to estimate the remaining variable values. These predictions are incorporated into the MILP model through additional constraints. Their approach, applied to the facility location problem, yields high-quality solutions with reduced computational times.

Similarly, [58] study a class of unit commitment problems in power generation, modeled as MILPs. To enhance computational efficiency, the authors propose learning-based approaches to identify promising regions of the solution space. In particular, they develop models that predict the values of binary decision variables based on patterns observed in previously solved instances. These predictions are then used to fix variable values prior to optimization by adding corresponding constraints to the MILP. Their results show that this reduction in problem size significantly decreases solution times while maintaining solution quality.

For example, the study in [83] propose a learning-based approach for repeatedly solving routing problems that are similar (e.g., only customer demands may vary from one problem to another), particularly the capacitated vehicle routing problem (CVRP). Their method builds a classifier using historical solutions to predict and fix the values (0 or 1) of variables that are likely to take the same value in the reference solution and in an optimal solution for the current problem to solve, thereby reducing the search space and accelerating the problem-solving process. This approach shows significant computational savings while maintaining solution quality, especially for scenarios where problem data vary slightly but retain a common structure.

Our primary motivation is driven by the specific needs of our industrial partner, whose

scheduling operations heavily depend on solving MILP problems. Our proposed shift enumeration method is explicitly designed to reduce the complexity of these problems while ensuring compatibility with their existing MILP-based algorithms. This approach not only improves computational efficiency but also offers flexibility in decision-making, which is critical given the scale and complexity of their scheduling tasks. The industrial partner in this thesis is UKG, a global leader in human capital management and workforce management software serving tens of thousands of organizations worldwide. UKG’s scheduling systems rely on explicit shift-based MILP formulations in which all feasible shifts must be enumerated before the optimization model is solved. For large retail clients — where the number of jobs, employees, and time granularity can reach hundreds of thousands of decision variables — the enumeration step itself becomes a computational bottleneck. This practical constraint directly shaped the research objectives of this thesis: rather than replacing the MILP-based architecture already deployed in production, the goal was to develop data-driven methods that reduce the size of the enumeration while remaining fully compatible with the existing solver pipeline. The problem instances studied in this thesis, including the real-life retail cases used in Chapters 4 and 5, were provided by UKG and reflect the scale and characteristics of scheduling problems encountered in their operational systems.

We conclude this section by highlighting the distinctive nature of our approach in relation to prior research. The number of decision variables in flexible personnel scheduling problems can be extremely large, often reaching millions. However, only a very small subset of these variables takes a positive value in an optimal solution. Designing a model that can accurately predict the values of the decision variables in this context is inherently challenging. Unlike the research in [63] and [58], who focus on predicting the variables’ values, our approach concentrates on predicting specific characteristics of these variables, namely, the start and end times of their associated shifts.

By identifying likely shift start and end times, our method narrows down the search space for the solver, significantly reducing the number of variables that need to be considered. This targeted prediction helps to maintain compatibility with existing MILP-based scheduling algorithms used by our industrial partner, optimizing computational efficiency without compromising on solution quality.

CHAPTER 3 STUDY'S ORGANIZATION

Based on the literature review presented in Chapter 2, it can be observed that although PSPs have been extensively studied, the scalability of explicit shift-based formulations remains a major challenge in large-scale retail environments with highly flexible shifts. In particular, when feasible shifts are enumerated *a priori*, the number of candidate shifts can grow extremely large, leading to very large MILP models. This thesis addresses this challenge by investigating how data-driven techniques can guide the shift enumeration process and reduce the size of these models while preserving solution quality. To this end, we explore the use of machine learning and data mining approaches that exploit information from previously solved instances to identify promising scheduling decisions.

We decided to start with the mono-job shift version of the FPSP, as it represents a simpler problem setting in which each shift is associated with a single job. This simpler structure makes it easier to investigate whether machine learning models can learn useful patterns from historical scheduling instances. As mentioned in Chapter 2, predicting some characteristics of potentially optimal shifts is generally easier than predicting the shifts themselves. In particular, predicting promising shift start and end times can help significantly reduce the number of candidate shifts that must be considered in the optimization model.

Chapter 4 presents a first attempt in this direction. It introduces a learning-based framework for guiding the enumeration of shifts in flexible personnel scheduling problems with mono-job shifts. The proposed approach uses deep learning models trained on previously solved scheduling instances to identify promising time periods for shift start and end times. These predictions are then used to restrict the enumeration of candidate shifts before solving the MILP model. By focusing the enumeration process on promising periods, the method aims to reduce the number of candidate shifts that must be considered in the optimization model. This work is presented in the article *Learning to Enumerate Shifts for Large-Scale Flexible Personnel Scheduling Problems*, published in *Journal of Scheduling*.

Given that, in retail environments, similar personnel scheduling problems are solved week after week, it is natural to exploit past solutions to guide the resolution of new instances. This observation motivated us to investigate whether previously solved instances with similar demand patterns could help identify promising shifts. In particular, we considered the possibility of searching in historical schedules for shifts that appeared in optimal solutions of instances with demand curves similar to those of the current instance.

Chapter 5 explores this idea by introducing a data mining-based approach for shift enumer-

ation. In this chapter, a k -nearest neighbors method is used to identify historical scheduling instances whose demand patterns resemble those of the current problem. The shifts appearing in the optimal solutions of these similar instances are then used to guide the enumeration of candidate shifts. This approach allows the shift enumeration process to benefit from the experience contained in previously solved scheduling instances. This work is presented in the article *Data Mining-Driven Shift Enumeration for Accelerating the Solution of Large-Scale Personnel Scheduling Problems*, published in *ACM Transactions on Evolutionary Learning*.

After studying the mono-job shift variant, we investigated whether the methodology developed in Chapter 4 could be extended to the more complex case of multi-job shift scheduling. In many retail environments, employees may perform multiple types of work within a single shift, which considerably increases the number of possible shift configurations and further exacerbates the combinatorial complexity of the problem.

Chapter 6 addresses this extension by adapting the explicit set-covering formulation to incorporate multi-job shifts composed of several job segments. The learning-based framework developed in Chapter 4 is also extended to predict the start and end times of these segments. By preserving the core principles of data-driven shift enumeration while incorporating job-switching flexibility, the proposed approach allows the shift enumeration process to remain tractable even in this more complex scheduling context. This work is presented in the article *A Data-Driven Matheuristic for Personnel Scheduling in the Retail Sector Considering Multi-Job Shifts*, submitted to *INFOR: Information Systems and Operational Research*, the journal of the Canadian Operational Research Society.

Finally, Chapter 7 provides a general discussion of the contributions of the thesis and situates the proposed approaches within the broader context of personnel scheduling research. Chapter 8 concludes the thesis by summarizing the main findings and presenting possible directions for future research.

CHAPTER 4 ARTICLE 1: LEARNING TO ENUMERATE SHIFTS FOR LARGE-SCALE FLEXIBLE PERSONNEL SCHEDULING PROBLEMS

F. Rastgar-Amini, C. Contardo, G. Desaulniers, and M. Gasse, “Learning to enumerate shifts for large-scale flexible personnel scheduling problems”, This paper was submitted on August 3, 2022, and published on May 21, 2025, *Journal of Scheduling*, vol. 28, no. 4, pp. 425–443, 2025.

4.1 Introduction

The process of employee scheduling focuses on constructing the most cost-effective employee schedules to fulfill the needs of one or multiple jobs over a planning horizon. Various organizations, including hospitals, airlines, call centers, retailers, and others, deal with personnel scheduling problems. These problems can vary according to the type of organization. In hospitals, for instance, employees serve patients 24 hours a day and seven days a week while working in shifts of 8 hours or 12 hours. During the week, shifts usually start at predetermined and fixed times (e.g., 7 a.m., 3 p.m., and 11 p.m.). In this case, personnel scheduling is the task of finding the right number of employees for each shift so that the demands (i.e. the required number of employees to fulfill the needs of the customers or patients) can be met. In contrast, large retailers and service industries receive customers only during business hours. Furthermore, the demand for employees can greatly fluctuate during the planning horizon and even from one hour to the next. In such environments, shifts may begin and end at any time of the day to obtain the least-cost schedules that employ, as much as possible, a minimum number of employees at all times. Consequently, instead of considering only a very limited number of possible shifts for each employee on each day (say, 3 or 5 like in nurse scheduling), a large number of potential shifts need to be taken into account (say, more than 900 per day if a day spans from 7 a.m. to 10 p.m., shifts can start and end at every quarter of an hour, and can last between 4 and 9 hours) and the demand is not expressed for each shift but for each time period (say, for each 15-minute period of the horizon). This high flexibility in shift start time and duration results in very complex problems that require efficient algorithms to produce optimal or near-optimal schedules. In the following, we refer to these problems as *flexible personnel scheduling problems* due to the flexibility offered by the numerous potential shifts.

In this research —proposed to us by our industrial partner UKG that commercializes personnel scheduling optimization software in various domains—, we focus on flexible personnel

scheduling problems in the service industry where the schedules of the employees depend on the service load associated with the presence of customers. Several problem variants are defined to address real-life problems in the retail and service industries. In particular, the variants with multiple jobs can be divided into two categories: with mono-job shifts and with multi-job shifts. In a mono-job shift, the employee is assigned to a single job for the whole shift duration, while multi-job shifts allow changing jobs once or multiple times within the same shift. Also, the problems may include different levels of complexity of days-off rules. For example, in some contexts, just a minimum number of days off assigned each week to each employee is sufficient while, in others where a one-month planning horizon is considered, days-off patterns with strict rules on the number of consecutive workdays or days off must be respected.

In this paper, we revisit a problem previously explored by [84]. This problem encompasses several variants, including the consideration of multiple jobs, the use of flexible mono-job shifts, and a one-week scheduling horizon, with the added requirement of allocating a minimum number of days off to each employee. The adoption of mono-job shifts is a common practice that simplifies operational procedures, particularly in personnel management, and eliminates the need to schedule short breaks during task transitions.

We do not consider breaks in our problem formulation, mirroring real-world scenarios where breaks are often granted in real-time based on immediate work demands. To compensate for the breaks to be added during the operations, the forecasted demand is typically adjusted during specific time intervals. In situations where this no-break assumption does not hold, our proposed heuristic can be adapted by introducing new decision criteria and considering break placement regulatory constraints.

Following an approach similar to that of the study in [84] and used by our industrial partner, we formulate the corresponding flexible personnel scheduling problem as a generalized set-covering model with side constraints. In this model, a binary variable is defined for each shift that can potentially be assigned to each employee for performing a job.

Our contribution is the use of machine learning (ML) techniques to reduce the size of the set-covering model and to obtain faster computational times without compromising too much solution quality. We propose to classify the shifts into promising and non-promising ones and keep in the model only the variables associated with the promising shifts. Instead of trying to classify the shifts themselves, we rather focus on predicting the subsets of start and end times of optimal shifts using a ML model, more precisely, a deep learning one. Promising shifts are then identified as those whose start and end times belong to these predicted times. To obtain good predictions, we test several state-of-the-art ML models, which require a training

dataset. In this respect, we generate artificial instances by applying perturbations to the input parameters of real-life instances and solve the corresponding set-covering models to optimality using a commercial mixed-integer linear programming (MILP) solver. To assess the proposed solution approach, we ran tests on real-life instances, but also on artificial instances that are generated out of the distribution of the training instances.

This paper is organized as follows. Section 4.2 presents a literature review on the subject. Section 4.3 states the problem studied and introduces the MILP model used in the optimization phase. Section 4.4 describes the methodology and the multiple ML models we consider in this study. We compare these models using a large dataset in Section 4.5. Finally, a brief conclusion and possible extensions are discussed in Section 5.6.

4.2 Literature review

Our proposed approach accelerates the solution of the integer programming models for flexible personnel scheduling problems where the shifts are enumerated explicitly. In this section, we present the contributions in the literature that encompass these three concepts: 1) employ integer programming models for personnel scheduling problems; 2) apply a heuristic to the problem variant studied; or 3) apply ML methods to accelerate the solution process of integer programs.

4.2.1 Integer programming models

Personnel scheduling problems were introduced by the study in [1] to decrease the vehicles' waiting time for service at toll booths with the minimum number of toll collectors. Since then, different types of problems have arisen for various application areas. Among the most popular research areas on personnel scheduling problems, the study in [2] point to health services, manufacturers, and call centers. However, the retail sector has been less explored. Most studies aim to determine schedules for employees at the lowest cost. There are usually three components to the cost of a solution: 1) the cost of assigning employees to jobs; 2) the penalty associated with over-coverage; and 3) the cost resulting from under-coverage, which results in a degradation of the service quality. For generating employee schedules, mixed-integer programming-based methods are the most popular approaches in the scientific literature [11].

Mathematical models for personnel scheduling problems can be divided into either explicit set-covering or implicit formulations. In the explicit models, one integer variable is associated with each shift type where the type of a shift is defined by all its characteristics such as start

time, end time, break time, job type, etc. In implicit models, some aspects of the shifts are determined by means of constraints. One example of implicit modeling would be to define shifts solely by their starting times and add additional constraints to the model to limit their lengths [3]. In many cases, the number of decision variables is considerably smaller than for an explicit formulation. According to the study in [3], in most cases, it is hard to express the employment cost using an implicit formulation. Explicit models are preferable when the cost of a shift is defined by its start time, end time, length, and placement of the breaks. Indeed, when dealing with a heterogeneous workforce, explicit approaches are predominantly employed in the literature [84]. Implicit approaches, on the other hand, primarily focus on establishing anonymous work schedules, where individual employee characteristics are disregarded. However, when these characteristics exhibit significant diversity, the allocation of anonymous shifts to employees becomes a complex and non-trivial task. Furthermore, the study in [4] point out that it is not possible to formulate multiple-job problems through implicit modeling.

The first explicit set-covering model for personnel scheduling was proposed by the study in [5] for the problem of assigning employees to toll booths. In this formulation, all feasible working patterns are enumerated for all possible start times, end times, and breaks. Then, one integer variable x_s is associated with each feasible work pattern s . Here, the problem objective is to find the shift assignment of the minimum total cost that covers the demand for open toll booths at each time interval.

Explicit modeling allows one to formulate different aspects of personnel scheduling problems. To give a few examples, explicit formulations have been used to produce employee schedules for nurse scheduling [e.g., 22, 23], for tour scheduling [e.g., 25], for personnel scheduling with employee transfers between departments [e.g., 26, 27], for a multi-job multi-task shift scheduling problem [e.g., 37, 38], and for a stochastic personnel scheduling problem with demand uncertainty [e.g., 85].

The set-covering formulation provides the advantage of modeling flexibility. Nevertheless, additional options for placement of breaks, start times, lengths, and end times for shifts result in a larger number of possible shifts and consequently larger models. For example, in the retail industry, shifts may start and end every quarter of an hour during a day. Enumerating all possible shifts for all available employees and for multiple jobs results in a very large model that requires efficient algorithms to be solved. These recurrent problems need to be solved in a reasonable amount of time frequently by efficient algorithms. Commercial solvers fail to achieve near-optimal solutions in a reasonable time for large-sized problems either formulated explicitly or implicitly. In the literature on personnel scheduling problems,

mostly decomposition methods and heuristic algorithms are implemented to solve large-scale personnel scheduling problems [11]. However, practical personnel scheduling problems encounter many shift feasibility constraints and many complicated ways to compute a shift's cost. In this case, it becomes difficult to design the pricing problems that are used to generate shifts dynamically in a column generation algorithm. To alleviate this, some authors [see, e.g., 36, 38] use context-free grammar to model these shift features. Heuristics are reviewed next.

4.2.2 Heuristics and metaheuristics

To the best of our knowledge, the studies in personnel scheduling that aim at reducing the number of variables in a set-covering model are limited to very early works such as the studies in the [6], [7], and [86]. Their methods, called *working set generation*, select a subset of schedules from all the possible work schedules and create only schedule variables for this subset. Different procedures have been developed to determine this subset. The one that seems to be the most efficient has been proposed by the study in [6] and builds the work schedule subset in three steps: first, it computes a subset of days-off configurations based on the coverage that they offer; second, it finds for each day a subset of shifts based, again, on the coverage that these shifts offer; and, finally, it enumerates all work schedules resulting from a combination of the selected days-off configurations and shifts. Another study in the [87] performs experimental tests to find out how the number of required employees is affected if they restrict the shifts' starting periods to only 3, 4, and 5 periods of the day instead of 24 periods. These works cannot be directly applied to our context as they assume that there is a single job to cover and that the employees are available all the time and can, thus, be considered identical.

Most heuristics and metaheuristics, for personnel scheduling problems, are developed to address complexities resulting from numerous and complicated constraints and not from the flexibility in shift enumeration. Such issues arise mostly in the healthcare domain where the preferences of the staff are important but the possible working shifts are very limited (3 to 5 per day). Generally, in these problems, the constraints are categorized into two groups: hard and soft constraints, depending on the legal regulations and personal preferences of the individual institutions and countries. Hard constraints must be met in order to obtain feasible solutions, whereas soft constraints may be violated with penalties. The study in [48] review the state-of-the-art heuristics and metaheuristics such as simulated annealing, tabu search, and genetic algorithms for nurse rostering problems. Among the more recent research, we can refer to the studies in the [23], [49], [88], and [50].

A few papers including [17, 26, 27, 84] develop heuristics for flexible personnel scheduling problems in the domain of service and retail industry. The studies in [26] and [27] focus on large-scale multi-department problems. In the former work, a large neighborhood search heuristic is proposed, where the neighborhoods are explored using a MILP model. In the latter, a two-stage solution method is devised. In the first stage, an aggregated MILP model that considers a coarser discretization of the time horizon for the demand curves is solved to yield an approximate solution. In the second stage, this solution is disaggregated by solving one or several small MILP models restricted to a small subset of shifts.

The studies in the [17] and [84] consider a similar environment to our problem. Assuming that the employees' days off are known, the study in [17] introduce a hybrid heuristic that combines tabu search and branch-and-bound to perform a local search. This algorithm requires average computational times exceeding two hours for instances with 20 to 30 employees and 2 to 4 jobs. Very recently, the study in [84] developed a Parallel Ruin and Recreate (PRR) heuristic that generates and assigns shifts during the solution process. A set of generator decisions (e.g., replace/extend/shorten a shift) forms the basic moves that lead to new feasible and infeasible solutions. At each iteration, PRR stimulates a well-targeted disruption (shift shortening) before correcting it with a sequence of basic moves that is computed by a truncated depth-first search. Furthermore, to improve efficiency, the search is performed in parallel by defining at each iteration multiple subproblems, each initiated by a disruption that can be corrected by a selected subset of employees.

4.2.3 Machine learning application

Recently, there has been an increasing interest within the mathematical optimization community in applying machine learning techniques to improve the performance of MILP solvers. A survey on the contribution of ML to combinatorial optimization problems is presented by the study in [55].

The study in [56] summarized the previous works on this subject in the following five avenues:

1. To improve the performance of an optimization solver by predicting if a reformulation is required for the problem before being solved by a MILP solver. For example, the study in [57] employed ML to predict whether a Dantzig–Wolfe decomposition is required for solving a MILP problem.
2. To improve the performance of solution methods. For instance, ML enhances the performance of some algorithms by finding good initial feasible solutions [58] and accelerates enumerative algorithms in branching variable selection [59, 82]. ML is also employed

to facilitate column selection in column generation algorithms [60] and to select cutting planes [61].

3. To reduce the computational complexity of solving optimization models under uncertainty by implementing dimensionality reduction methods [62].
4. To generate the solutions (ideally optimal) of a combinatorial optimization problem [56] or to predict the value of a subset of decision variables [58, 63].
5. To predict the objective value (ideally optimal) of a problem [64].

We focus on the studies of [63] and [58] from the fourth category in which we observe some aspects that are common to our research. They studied recurrent problems for which there are minor differences between the instance input data and are solved by a MILP solver. To accelerate the solution time, they used ML methods to get information from previously solved instances and fixed the value of some variables in similar instances to be solved in the future. Subsequently, the computational performance of MILP solvers is improved due to the elimination of some variables and the reduction of the problem size.

The study in [63] assume the existence of a reference instance to a combinatorial optimization problem. This instance can be modeled as a MILP model and solved optimally. The problem needs to be solved again if a perturbation happens in the input of the initial instance. A database of resolutions can be built by collecting the changes that have happened so far or that can be simulated a priori. The authors study how information can be retrieved from this database to support the analysis of potentially perturbed instances. To predict whether a change will influence the reference optimal solution and to what degree, they formulate this problem as a supervised learning problem. A binary classifier is trained to predict whether the entire or a part of an optimal solution for the reference instance is applicable to the perturbed instance. If only a part of the reference solution is valid for the new instance, the value of the other variables is predicted by regression. These decisions are translated as additional constraints and imposed to the perturbed problem. The authors employed their approach to the facility location problem and obtained satisfactory solutions in a decreased time.

The study in [58] study a class of optimization problems that deal with power-generating unit scheduling. These problems are formulated as MILP models and solved by a MILP solver to determine the schedule and production level of power generators over a planning horizon. To improve the computational performance of the MILP solver, the authors propose three approaches including the prediction of affine subspaces that include the optimal solution,

with a very high likelihood. Their contributions are inspired by intuitive patterns related to these problems which are not included in the MILP modeling. For example, the operators observed that a specific type of generator was mostly scheduled to operate throughout the day whereas other types of generators were usually assigned to peak demand periods. The authors developed an ML model to predict the value of the binary decision variables by learning from previously solved instances. The output of the ML model determines the value that variables should be fixed at before solving the MILP. To do so, supplementary constraints are added to the MILP model that indicates the value of the variables. Results of this research showed that a substantial size reduction of the problem reduced the complexity and solution time of the MILP solver.

We conclude this section by highlighting the distinctive nature of our approach in relation to prior research. The number of decision variables in flexible personnel scheduling problems can be extremely large, often reaching millions. However, only a very small subset of these variables takes a positive value in an optimal solution. Designing a model that can accurately predict the values of the decision variables in this context is inherently challenging. Unlike the studies in [63] and [58] who focus on predicting the variables' values, our approach concentrates on predicting specific characteristics of these variables, namely, the start and end times of their associated shifts. By identifying likely shift start and end times, our method narrows down the search space for the solver, significantly reducing the number of variables that need to be considered. This targeted prediction helps to maintain compatibility with existing MILP-based scheduling algorithms used by our industrial partner, optimizing computational efficiency without compromising on solution quality.

4.3 Problem description and mathematical modeling

We start by describing the flexible personnel scheduling problem studied (Section 4.3.1). Then, Section 4.3.2 provides the notation used in the proposed mathematical model, which is presented afterwards in Section 4.3.3.

4.3.1 Problem statement

In the retail industry, the personnel scheduling process starts with forecasting the workload approximately two to eight weeks before the beginning of the planning horizon [85]. In fact, the forecasts focus on the expected number of sales/transactions by department, taking into account several factors such as seasonality and promotions. These sales are subsequently converted into a demand in employees per job that can vary over the planning horizon.

Several types of jobs may be relevant in a retail context: salespersons, cashiers, clerks, and multiple others. Once these demands are established, personnel schedules can be computed with the goal of covering these demands as best as possible with the available employees. In many organizations, this scheduling process is repeated on a weekly basis.

The flexible personnel scheduling problem to be solved each week can be stated as follows. There is a set of jobs J to be performed by a set of skilled employees E . Among all employees in E , there is only a subset of the employees, denoted E^j , that are qualified to perform job $j \in J$. There is a set of mono-job shifts S , where a shift $s \in S$ is defined by a pair of valid start and end times a_s and b_s (say, at any quarter of an hour) yielding a valid duration g_s (say, between 4 and 9 hours). Because the demands of different jobs may follow different patterns (for example, some jobs may start at 6:00 AM, while others only at 8:00 AM), the subset of candidate shifts S^j for a job $j \in J$ may differ from one job to another. Also, as is often the case in real life, we assume that there is no isolated demand spanning less than the minimum shift duration. Therefore, a shift is considered a *candidate* for a job if it covers periods at which the job's demand is greater than zero without any harm to the feasibility of the problem. Furthermore, given the skills of an employee $e \in E$ and their availability over the planning horizon, they can be assigned to a subset of shifts S^e . For each job $j \in J$, a demand curve (a number of employees required as a function of time) to cover at best is provided. Dividing the planning horizon into a set P of disjoint 15-minute periods, this curve indicates the number of employees d_p^j required on the job j at period $p \in P$. We assume that the periods in P are chronologically numbered from 1 to $|P|$ and that the planning horizon is also divided into 7 days and denoted $H = \{1, 2, \dots, 7\}$.

The demand for employees fluctuates during the time horizon and even during the day. The ideal scheduling for such a problem happens when the number of employees assigned to a job is equal to the job's demand at each period while considering the following working rules: there must be a minimum rest time n^R (in periods) between two consecutive shifts assigned to an employee, and there must be a minimum number of days off n^O assigned to each employee. As mentioned above, there are restrictions associated with the qualifications and time availability of the employees which may make it impossible to cover adequately all the demands. Hence, to ensure demand coverage, anonymous shifts with high costs can be scheduled and assigned subsequently to temporary workers. Additionally, since shifts must have a minimum duration, sometimes there is no choice but to cover some time periods with no demand or to assign more employees than there is demand at a specific period, yielding what is called over-coverings. The employees are unproductive at over-covering time periods, so a large penalty is associated with over-covering. The over-covering penalty associated with each job in a period is calculated using a non-decreasing step-wise function that favors the

distribution of the over-coverings, if any, over the periods.

Employees should also be treated equally over time based on the number of hours they work. In fact, the remuneration of the employees is made on different levels following a strictly increasing function. Each level is made up of certain hours of work. For example, let each 8 hours of work make a level. Then, if an employee e works 20 hours, then he/she will be paid at 3 levels in the following order: c_1^E per period for the first 8 hours, c_2^E per period (with $c_2^E > c_1^E$) for the second 8 hours, and finally c_3^E per period (with $c_3^E > c_2^E$) for the last 4 hours. Observe that this remuneration function does not correspond to the real remuneration of the employees. Indeed, if the real function was used, the more experienced employees who have a larger hourly rate would work less than the junior employees.

The proposed model (see Section 4.3.3) is an explicit one that requires the enumeration of all possible shifts. In the retail industry, there is high flexibility in the start time, the end time, and the duration of the shifts. For instance, when shifts can start and end at every quarter of an hour and can last between 3 and 8 hours by steps of 15 minutes, the set of shift starting times and the set of possible shift lengths are $\text{shift_starts} = \{00:00, 00:15, 00:30, \dots, 23:45\}$ and $\text{shift_lengths} = \{3h, 3h15m, \dots, 8h\}$, respectively. Therefore, the number of possible shifts is very large.

Figure 4.1 illustrates an example of the demand curves for two distinct jobs spanning 672 periods of 15 minutes within a week. Notably, Job 1 exhibits a demand curve that covers a shorter timeframe compared to Job 2, resulting in a reduced number of potential starting times for candidate shifts in Job 1 as opposed to the broader time span available for Job 2. Consequently, increasing the number of potential start times inherently leads to an escalation in the number of candidate shifts. In Table 4.1, we further scrutinize the number of starting times and the number of shifts ultimately selected in an optimal solution for both Job 1 and Job 2. The heightened demand curve fluctuations for Job 2 contribute to a significant increase in the number of shifts selected within the optimal solution. This example underscores two crucial insights:

- Effective shift enumeration requires a high degree of flexibility to minimize employment costs;
- Only a fraction of the candidate shifts ultimately participates in the optimal solution, warranting the development of a method to identify and exclude those shifts less likely to be part of an optimal solution.

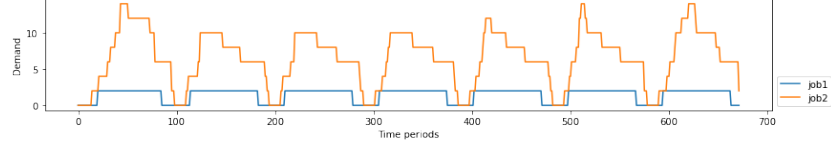


Figure 4.1 Examples of demand curves of two jobs

Table 4.1 Variety of candidate shifts for the example of Figure 4.1

Job	Number of starting times		Number of shifts	
	Potential	Optimal	Candidate	Optimal
1	402	39	8816	46
2	509	141	11889	190

4.3.2 Notation

The notation used in the mathematical model is presented in Tables 4.2 to 4.4.

Table 4.2 Sets and subsets

Set	Description
S	Set of all enumerated shifts
P	Set of all time periods in the planning horizon
E	Set of employees
J	Set of jobs
H	Set of days in the planning horizon
S^j	Subset of shifts that are candidate for job $j \in J$
S^p	Subset of shifts that cover period $p \in P$
S^h	Subset of shifts that start on day $h \in H$
S^e	Subset of shifts that can be assigned to employee $e \in E$
E^j	Subset of employees that are qualified for job $j \in J$
J^e	Subset of jobs that employee $e \in E$ is qualified for
E_p^j	Subset of employees that are qualified for job j and available at period p
K	Set of steps for the over-covering penalty step-wise function
Q	Set of steps for the employee remuneration step-wise function

Table 4.3 Parameters

Parameter	Description
d_p^j	Number of employees required (demand) for job $j \in J$ at period $p \in P$
a_s, b_s	Beginning and end periods of shift $s \in S$
g_s	Length (in number of periods) of shift $s \in S$
n_e^O	Minimum number of days off to assign to employee $e \in E$
n^R	Minimum number of rest periods between two shifts assigned to an employee
c_s^A	Cost of an anonymous shift $s \in S$
c_q^E	Employee remuneration per period on step $q \in Q$
m_q	Maximum number of periods on step $q \in Q$
c_k^V	Cost per over-covering on step $k \in K$
m_k	Maximum number of over-coverings on step $k \in K$

Table 4.4 Decision Variables

Variable	Description
$z_{s,e,j}$	Binary variable equal to 1 if shift $s \in S^j \cap S^e$ is assigned to employee $e \in E$ for job $j \in J$
$u_{s,j}$	Number of anonymous shifts $s \in S$ assigned for job $j \in J$
$v_{p,j,k}$	Number of over-coverings for job $j \in J$ at period $p \in P$ and on step $k \in K$
$w_{e,q}$	Number of periods worked by employee $e \in E$ on step $q \in Q$
$o_{e,h}$	Binary variable equal to 1 if employee $e \in E$ is off on day $h \in H$

4.3.3 Mathematical model

Given the above notation, the flexible personnel scheduling problem can be modeled as the following integer program, which is similar to the study in the [84]:

$$\min \sum_{j \in J} \sum_{s \in S^j} c_s^A u_{s,j} + \sum_{e \in E} \sum_{q \in Q} c_q^E w_{e,q} + \sum_{j \in J} \sum_{p \in P} \sum_{k \in K} c_k^V v_{p,j,k} \quad (4.1)$$

$$s.t. : \quad \sum_{s \in S^p \cap S^j} u_{s,j} + \sum_{e \in E_p^j} \sum_{s \in S^e \cap S^p \cap S^j} z_{s,e,j} - \sum_{k \in K} v_{p,j,k} = d_p^j \quad \forall p \in P, j \in J \quad (4.2)$$

$$\sum_{j \in J^e} \sum_{s \in S^e \cap S^j \cap S^h} z_{s,e,j} + o_{e,h} = 1 \quad \forall e \in E, h \in H \quad (4.3)$$

$$\sum_{j \in J^e} \sum_{s \in S^e \cap S^j} g_s z_{s,e,j} - \sum_{q \in Q} w_{e,q} = 0 \quad \forall e \in E \quad (4.4)$$

$$\sum_{h \in H} o_{e,h} \geq n_e^O \quad \forall e \in E \quad (4.5)$$

$$\begin{aligned} & \sum_{j \in J^e} \sum_{s \in S^e \cap S^j \cap S^{h+1}} a_s z_{s,e,j} + M o_{e,h+1} \\ & - \sum_{j \in J^e} \sum_{s \in S^e \cap S^j \cap S^h} (b_s + 1 + n^R) z_{s,e,j} \geq 0 \quad \forall e \in E, h \in H \setminus \{7\} \end{aligned} \quad (4.6)$$

$$z_{s,e,j} \in \{0, 1\} \quad \forall j \in J, e \in E^j, s \in S^j \cap S^e \quad (4.7)$$

$$u_{s,j} \geq 0, \text{ integer} \quad \forall j \in J, s \in S^j \quad (4.8)$$

$$v_{p,j,k} \in [0, m_k] \quad \forall p \in P, j \in J, k \in K \quad (4.9)$$

$$w_{e,q} \in [0, m_q] \quad \forall e \in E, q \in Q \quad (4.10)$$

$$o_{e,h} \in \{0, 1\} \quad \forall e \in E, h \in H \quad (4.11)$$

The objective function (4.1) minimizes the sum of the personnel costs, the anonymous shift costs, and the over-covering penalties. Constraints (4.2) ensure that the demand for each job at each time period is covered by personalized or anonymous shifts and make the calculation of over-coverings possible. Constraints (4.3) impose that each employee is assigned to a shift or a day off on each day of the planning horizon. The amount of working hours of each employee that falls on each step $q \in Q$ is calculated through the constraints (4.4). Each employee e receives at least n_e^O days off according to constraints (4.5). The big-M constraints (4.6) enforce a minimum rest time of n^R periods between two working shifts that are assigned to the same employee on consecutive days, wherein $M = \max_{s \in S} \{b_s + 1 + n^R\}$. Finally, constraints (4.7)-(4.11) restrict the domain of the variables.

4.4 Methodology

We develop deep learning models to predict a subset of time periods that are likely to be the start or the end of the shifts in an optimal solution of a particular problem. To provide a training dataset, we generate artificial instances from real data and solve the corresponding MILP models (4.1)–(4.11) to near optimality (i.e., with an optimality gap tolerance of 0.01%) using a commercial MILP solver. The parameters and optimal solutions of these instances form the training data, which is used to train a deep learning classifier that can predict simultaneously shift start time periods and shift end time periods. Once the potential shift starting and ending periods are predicted for a new instance, shifts are enumerated according to these subsets of periods, and the MILP model with a reduced number of variables is solved.

In this section, we start by presenting the real-life instances that were made available to us by our industrial partner and how we generated additional instances from them (Section 4.4.1). Then, we present the features of an instance to be considered for prediction and define the output labels (Section 4.4.2). Next, we discuss the learning strategy (Section 4.4.3) used to train the ML models which are then described (Section 4.4.4). Finally, we present the pipeline of the optimization heuristics (Section 4.4.5).

4.4.1 Real-life instances and data generation

For our study, 14 real-life problem instances were made available to us by our industrial partner UKG. Each instance provides all employee information and workload data over one week. Table 4.5 presents the numbers of jobs and employees associated with each instance.

To develop a good ML model, a large training dataset is required. In this regard, we generate additional variations of each original instance by perturbing some of its input data. We consider the following two inputs to perturb:

1. Job demand curves: The demand for jobs changes during the day and the patterns of the demand over different days of a week are also different. To generate variations from an original instance, we perturb the demand curves by adding noises to the level of

Table 4.5 Number of jobs and employees in the reference instances and number of generated artificial sub-instances from each instance

Instance	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Jobs	2	2	2	2	4	4	4	4	5	5	7	7	8	10
Employees	17	27	34	54	34	47	54	94	25	50	36	72	94	50
Sub-instances	1,400	1,400	1,400	1,400	2,800	2,800	2,800	2,800	3,500	3,500	4,900	4,900	5,600	7,000

the demand. More precisely, for each job $j \in J$, we pick uniformly at random between 10% and 20% of each day's periods and we apply a Gaussian noise to its demand as follows: $d_p^j \leftarrow d_p^j + \lfloor \mathbf{x} + 0.5 \rfloor$, where $\mathbf{x}$ is sampled from a normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.2 \times d_p^j$. Note that the demand remains unchanged when it is zero at the beginning or the end of a day.

2. Employees: For the new instances, we pick at random $\lceil 1\% \times |E| \rceil$ employees and remove them from the employee set.

All the remaining data is left unchanged. Note that the parameter values used to generate the perturbations have been selected to reflect the stability that we observe in the employee availability and job demand curves from one week to the next in the retail industry. Nevertheless, these perturbations are sufficiently large to yield a varied set of instances.

In total, 100 artificial instances for each original instance (i.e., 1,400 instances) were generated in this way and solved. To increase the number of data points in the training set and given that the shift start and end times can be selected almost independently from one day to another and from one job to another, we have chosen to decompose each instance and its associated MILP solution into sub-instances and sub-solutions with respect to each job and each day. Each sub-instance is considered in the training dataset, denoted $\mathcal{D}$, as an individual training sample and represents the shift scheduling problem for one job $j \in J$ over one day $h \in H$. Therefore, we extract $|J| \times |H|$ sub-instances from each instance which gives a further boost to the size of training data (a total of 46,200 sub-instances distributed over the original instances as shown in Table 4.5). Moreover, decomposing the instances per job helps to have a fixed dimension input size for all instances, which can be fed into a standard learning model. Finally, we use 67% of these artificial instances (67% of sub-instances for each original instance) for training and the remaining 33% for validation. Additional instances are also generated for testing purposes as mentioned in Section 4.5.

4.4.2 Features and labels

Each data point $i \in \mathcal{D}$ is associated with a job j and a day h and is defined by a set of features and two labels. For the features, we consider two sets that yield two different ML models. In the first set, we only use the information stemming from the demands for job j on day h , and from the qualifications and the availability of the employees. For the second set, we also consider information obtained from the linear relaxation optimal solution of the MILP model (4.1)–(4.11).

More precisely, denoting by $\tilde{p}(h)$ the index of the first period of day h and L the number of

time periods in a day, the features of a data point i associated with job j and day h , include the two vectors:

- $X_1^{(i)} = (d_{\tilde{p}(h)}^j, d_{\tilde{p}(h)+1}^j, \dots, d_{\tilde{p}(h)+L-1}^j)$, which indicates the demands for job j at each time period of day h in instance i ;
- $X_2^{(i)} = (|E_{\tilde{p}(h)}^j|, |E_{\tilde{p}(h)+1}^j|, \dots, |E_{\tilde{p}(h)+L-1}^j|)$, which specifies the number of qualified and available employees for job j at each time period of day h in instance i .

Beside the feature vectors X_1 and X_2 , we can extract information from solving the linear relaxation of the MILP problems similar to the studies in the [89] and [82]. The second set of features includes two other vectors, denoted X_3 and X_4 , that rely on the notation:

$$\nu_{\tilde{p}(h)+l-1}^j = \max_{e \in E^j, s \in S^j} \{z_{s,e,j}^{R(i)} | a_s = \tilde{p}(h) + l - 1\}, \quad \forall l \in \{1, 2, \dots, L\}, \quad (4.12)$$

$$\mu_{\tilde{p}(h)+l-1}^j = \max_{e \in E^j, s \in S^j} \{z_{s,e,j}^{R(i)} | b_s = \tilde{p}(h) + l - 1\}, \quad \forall l \in \{1, 2, \dots, L\}, \quad (4.13)$$

where $z_{s,e,j}^{R(i)}$ denotes the value of the corresponding $z_{s,e,j}$ variable in the optimal solution of the linear relaxation of the instance from which data point i is derived. These vectors for data point i are defined as:

- $X_3^{(i)} = (\nu_{\tilde{p}(h)}^j, \nu_{\tilde{p}(h)+1}^j, \dots, \nu_{\tilde{p}(h)+L-1}^j)$, which specifies for each period of day h if it is used as a shift starting period in the linear relaxation optimal solution;
- $X_4^{(i)} = (\mu_{\tilde{p}(h)}^j, \mu_{\tilde{p}(h)+1}^j, \dots, \mu_{\tilde{p}(h)+L-1}^j)$, which indicates for each period of day h if it is used as a shift ending period in the linear relaxation optimal solution.

To develop supervised learning models, we must define labels for each data point $i \in \mathcal{D}$. In our case, they are derived from the optimal solution of the MILP problem that contains sub-instance i and correspond to two vectors of binary values, $\underline{Y}^{(i)}, \overline{Y}^{(i)} \in \{0, 1\}^L$, containing one component for each period of day h and indicating whether that period coincides with the beginning or the end of a shift in that optimal solution. More precisely, let $z_{s,e,j}^{*(i)}$ be the value of the corresponding $z_{s,e,j}$ variable in the optimal solution of the problem that contains sub-instance i . Furthermore, let $S_{j,h}^{(i)} = \{s \in S^j \cap S^h | \exists e \in E^j \text{ such that } z_{s,e,j}^{*(i)} = 1\}$ be the subset of shifts in this solution that belong to day h and are assigned to job j . Then, $\underline{Y}_l^{(i)} = 1$, $l \in \{1, \dots, L\}$, if there exists a shift $s \in S_{j,h}^{(i)}$ such that $a_s = \tilde{p}(h) + l - 1$ and 0 otherwise. Similarly for the end periods, we set $\overline{Y}_l^{(i)} = 1$, $l \in \{1, \dots, L\}$, if there exists a shift $s \in S_{j,h}^{(i)}$ such that $b_s = \tilde{p}(h) + l - 1$ and 0 otherwise.

4.4.3 Learning strategy

We frame the problem of predicting the optimal start-of-shift and end-of-shift periods as a supervised multi-label classification problem. Our input space is $\mathcal{X} = \mathbb{R}^{L \times m}$, with either $m = 2$ for simple MILP parameter features or $m = 4$ for the augmented features, as described in Section 4.4.2. Our output space is $\mathcal{Y} = \{0, 1\}^{L \times 2}$, with binary labels that indicate if each time period is start-of-shift and/or end-of-shift. What we aim for is then a mapping from features to labels, $f : \mathcal{X} \rightarrow \mathcal{Y}$ (a.k.a., a classifier). Given a distribution of MILP instances $p(X, Y)$ and a distance function $Dist : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$, the goal of supervised learning is to find an optimal mapping f^* , which makes the most accurate predictions on average,

$$f^* = \arg \min_f \mathbb{E}_{X, Y} [Dist(Y, f(X))]. \quad (4.14)$$

In this work we use the popular Hamming loss as our distance function, which is minimized using a simple binary relevance scheme [90]. We employ parametric probabilistic models $q_\theta : \mathcal{X} \rightarrow [0, 1]^{L \times 2}$ to output the probability for each label to take value one, with θ the model parameters that are to be learned. Our training objective is then the binary cross-entropy (BCE) loss between the probabilistic predictions and the true labels. Finally, we achieve (4.14) via empirical risk minimization from a collection of training data points $\mathcal{D} = \{(X^{(i)}, Y^{(i)})\}_{i=1}^N$, as described in Section 4.4.1, leading to the following learning problem,

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^N BCE(Y^{(i)}, q_\theta(X^{(i)})). \quad (4.15)$$

The individual classification accuracy of each label is then obtained using the binary predictions $f_{\theta^*}(X) = \lceil q_{\theta^*}(X) \rceil$. In practice we solve (4.15) via stochastic gradient descent on the training data set, with early stopping on a validation set to prevent overfitting.

4.4.4 Machine learning models

We experiment with three different model architectures for f_θ , a fully-connected neural network (FCN), a convolutional neural network (CNN) [91], and a U-shaped encoder-decoder convolutional neural network (UNet) [92], inspired from the problem of image segmentation in computer vision [93]. While a wide range of ML models and architectures could be used to tackle our learning problem, our choice of models was guided by the signal-like aspect of the start-of-shift / end-of-shift prediction problem. Indeed, both our input and output spaces $\mathcal{X}$ and $\mathcal{Y}$ exhibit a 1-dimensional temporal structure (due to the ordering of the L time periods). The long-standing ML models of choice for signal processing are CNNs, with UNets

being a more recent contender for specific domains such as image segmentation. Therefore, both architectures appear like a natural fit for our problem. FCNs on the other hand are a more straightforward, naive family of models, which can apply to any kind of problem but usually perform worse when the data exhibit some structure (like a temporal signal), precisely because by design they cannot leverage this structural knowledge. Hence, in our experiments we expect to obtain best results with our CNN and UNet networks, while our FCN network is included as a baseline for comparison.

Our FCN network, described in Table 4.6, simply employs 6 densely connected layers with batch normalization, and a decreasing number of hidden units.

Our CNN network, described in Table 4.7, consists of 6 one-dimensional convolutional layers, with an increasing number of kernels and a decreasing kernel length.

Our UNet network, described in Table 4.8, consists of 4 stacked one-dimensional convolutional encoders, with an increasing number of kernels and down-sampling between each encoder (implemented using a 2-max-pooling), followed by 4 stacked one-dimensional convolutional decoders with a decreasing number of kernels and up-sampling between each decoder (implemented as transposed convolutions), and with skip-connections between each encoder and decoder of the same level (implemented as layer concatenation).

4.4.5 Optimization heuristic pipelines

Once the ML model has been chosen and trained, it can be used to determine subsets $\underline{P}_{j,h}$ and $\overline{P}_{j,h}$ of possible shift start and end times for a given pair of job $j \in J$ and day $h \in H$. Depending on the features used by this model (based or not on the linear relaxation solution), the design of the solution process differs as follows. If only the X_1 and X_2 vector features are considered, then the basic optimization heuristic is the following:

Table 4.6 FCN architecture

Layer	Type	Hidden units	Activation	Shape
0	Input	-	-	$L \times m$
1	Dense	500	relu + batchnorm	500
2	Dense	400	relu + batchnorm	400
3	Dense	300	relu + batchnorm	300
4	Dense	200	relu + batchnorm	200
5	Dense	200	relu + batchnorm	200
6	Dense	$2 \times L$	sigmoid	$L \times 2$

Table 4.7 CNN architecture

Layer	Type	Kernel Size kernels $\times$ length	Activation	Shape length $\times$ channels
0	Input	-	-	$L \times m$
1	1D Conv	32×96	relu	$L \times 32$
2	1D Conv	64×48	relu	$L \times 64$
3	1D Conv	128×24	relu	$L \times 128$
4	1D Conv	256×12	relu	$L \times 256$
5	1D Conv	512×6	relu	$L \times 512$
6	1D Conv	2×1	sigmoid	$L \times 2$

1. For each pair of job $j \in J$ and day $h \in H$, compute $\underline{P}_{j,h}$ and $\overline{P}_{j,h}$ using the trained ML model;
2. Build the MILP model (4.1)–(4.11) considering only personalized shift variables $z_{s,e,j}$ such that $s \in S^j \cap S^h$ for some $j \in J$ and $h \in H$, $a_s \in \underline{P}_{j,h}$, and $b_s \in \overline{P}_{j,h}$;
3. Solve the resulting MILP model using a commercial MILP solver.

Alternatively, if the X_3 and X_4 vector features are also taken into account, then the pipeline of the feature-enhanced optimization heuristic is as follows:

1. Build the MILP model (4.1)–(4.11) considering all personalized shift variables $z_{s,e,j}$;
2. Solve the linear relaxation of this model to collect the features in vectors X_3 and X_4 ;
3. For each pair of job $j \in J$ and day $h \in H$, compute $\underline{P}_{j,h}$ and $\overline{P}_{j,h}$ using the trained ML model;
4. From the MILP model (4.1)–(4.11), filter out all personalized shift variables $z_{s,e,j}$ such that $s \in S^j \cap S^h$ for some $j \in J$ and $h \in H$, and at least one of the following two conditions hold: $a_s \notin \underline{P}_{j,h}$ or $b_s \notin \overline{P}_{j,h}$;
5. Solve the resulting MILP model using a commercial MILP solver.

In the following, we denote this feature-enhanced heuristic by MILP-II* when combined with predictor $\Pi = \text{CNN}$, FCN , or UNet . Its basic counterpart is denoted MILP-II. For all heuristics, we consider tolerance of 1% on the optimality gap when solving the resulting MILP model.

Table 4.8 UNet architecture

Layer	Type	Kernel Size kernels $\times$ length	Activation	Shape length $\times$ channels
0	Input	-	-	$L \times m$
1	1D Conv	32×3	relu	$L \times 32$
2	1D Conv	32×3	relu	$L \times 32$
3	2-Max Pooling	-	-	$L/2 \times 32$
4	1D Conv	64×3	relu	$L/2 \times 64$
5	1D Conv	64×3	relu	$L/2 \times 64$
6	2-Max Pooling	-	-	$L/4 \times 64$
7	1D Conv	128×3	relu	$L/4 \times 128$
8	1D Conv	128×3	relu	$L/4 \times 128$
9	2-Max Pooling	-	-	$L/8 \times 128$
10	1D Conv	256×3	relu	$L/8 \times 256$
11	1D Conv	256×3	relu	$L/8 \times 256$
12	2-Max Pooling	-	-	$L/16 \times 256$
13	1D Conv	512×3	relu	$L/16 \times 512$
14	1D Conv	512×3	relu	$L/16 \times 512$
15	2-Strided 1D Trans Conv	256×3	-	$L/8 \times 256$
16	Concat 11, 15	-	-	$L/8 \times 512$
16	1D Conv	256×3	relu	$L/8 \times 256$
17	1D Conv	256×3	relu	$L/8 \times 256$
18	2-Strided 1D Trans Conv	128×3	-	$L/4 \times 128$
19	Concat 8, 18	-	-	$L/4 \times 256$
20	1D Conv	128×3	relu	$L/4 \times 128$
21	1D Conv	128×3	relu	$L/4 \times 128$
22	2-Strided 1D Trans Conv	64×3	-	$L/2 \times 64$
23	Concat 5, 22	-	-	$L/2 \times 128$
24	1D Conv	64×3	relu	$L/2 \times 64$
25	1D Conv	64×3	relu	$L/2 \times 64$
26	2-Strided 1D Trans Conv	32×3	-	$L \times 32$
27	Concat 2, 26	-	-	$L \times 64$
28	1D Conv	32×3	relu	$L \times 32$
29	1D Conv	32×3	relu	$L \times 32$
30	1D Conv	2×1	sigmoid	$L \times 2$

4.5 Computational experiments

In this section, we perform computational experiments to evaluate the efficiency of the proposed approaches in various instances. In Section 4.5.1, we describe the computational environment and the instances used for benchmarking. In Section 4.5.2, we evaluate the performance of the classifiers. In Section 4.5.3, we compare the performance of the basic and feature-enhanced optimization heuristics between them and against other heuristics used in practice and in the literature. Finally, in Section 4.5.4, we assess how robust our approaches are against instances from different distributions.

4.5.1 Experimental setup

The three proposed ML models (FCN, CNN, UNet) described in Section 4.4 were implemented in Python 3 using the scikit-learn [94] ML library. The libraries of Keras were used to design deep learning models. IBM ILOG CPLEX 20.1.0 was used as the MILP solver. The code responsible for loading the input data, querying the ML models, and constructing the MILP models was also written in Python 3. All MILP problems were solved on a Linux machine equipped with a 12-core (2 threads per core) Intel Core i7 processor clocked at 3.4 GHz with 16 GB RAM.

As mentioned in Section 4.4.1, we have access to 14 real-life instances (see Table 4.5) that we kept for our main tests. From these instances, we have generated 1,400 artificial instances that form the training and validation datasets. Moreover, using the same procedure described in Section 4.4.1, we generated from each original instance 10 additional instances to be used during the testing phase. Finally, to test the robustness of our best solution approaches to larger input data variations, we also generated 10 other artificial instances based on a similar but different distribution. We describe how these instances were generated in Section 4.5.4.

4.5.2 Evaluation of the predictors

The three ML models have been trained using the training and validation datasets. In total, it takes approximately 10 minutes, two hours and one hour for training the feature-enhanced FCN, CNN, and UNet models, respectively, i.e., when considering the X_3 and X_4 feature vectors.

In this section, we evaluate the performance of these three ML models on the test dataset (composed of 140 artificial instances). For the evaluation metric, we use the area under the Receiver Operating Characteristic (ROC) curve [95]. Figure 4.2 shows the average of the curves over the 96 periods, for each predictor. A ROC curve depicts the true and false

positive rates on the y and x axes, respectively. The point in the upper left corner of the ROC space is where the ROC curve of a perfect predictor would touch. A random predictor would yield a diagonal line from the bottom left corner to the top right corner as shown in Figure 4.2. To compare the performance of multiple predictors, one can, thus, compare the areas under their ROC curves. Larger areas (close to 1) indicate a better predictor. Thus, we observe in Figure 4.2 that UNet*, UNet, and CNN* predictors have the highest areas under the ROC curves (approximately 0.947, 0.942, and 0.941), indicating that the other models are outperformed by them.

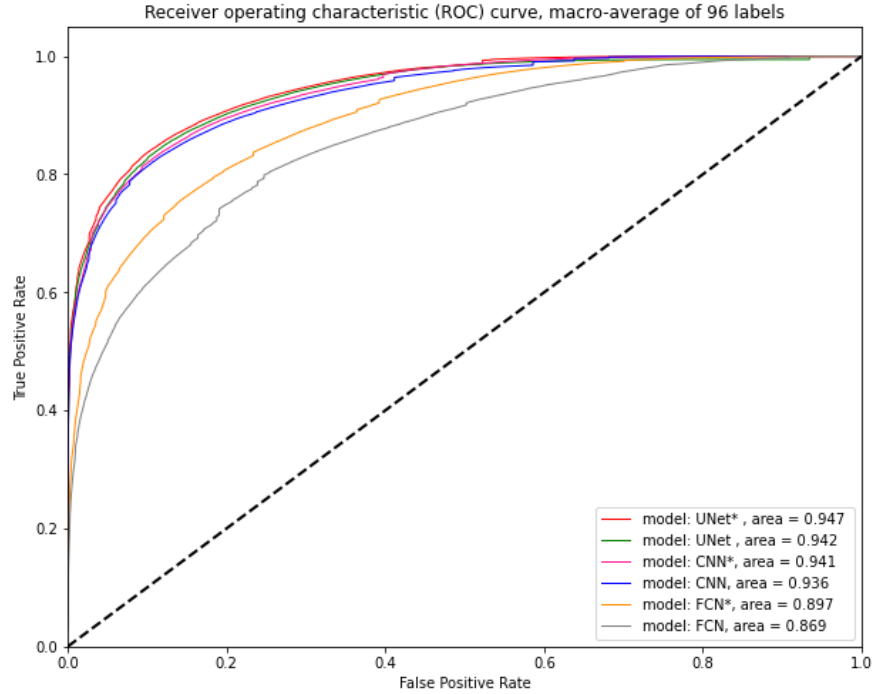


Figure 4.2 The ROC curves of the six predictors

We have observed that the performance of each ML model may vary greatly depending on the period of the day. Indeed, it is easier to predict when shifts should start at the beginning of a day when the demand is increasing, than in the middle of the day when demand is steady but some shifts must start because previous shifts are ending. On the contrary, for ending periods, the predictors have a better performance at the end of the day when demand is decreasing than in the middle of the day. To account for this instability in the prediction performance, we propose to not forward the output of the ML models directly to the shift enumeration procedure. Instead, similar to the approach of the study in [58], for each period, the output of a predictor is accepted only if both the recall and precision metrics of this predictor on the validation dataset surpass a predetermined threshold (α). Otherwise, the predictor is deemed

unreliable for the corresponding period and we apply the safe approach of considering this period as a valid starting/ending period. Hence, the predictor is only used for a subset of periods.

Figure 4.3 illustrates the performance comparison of six ML models in predicting 96 periods of the output vectors. Figures 4.3a and 4.3b correspond to predictors of starting and ending times, respectively. The x -axis represents various thresholds (α) ranging from 0.50 to 0.70, while the y -axis shows the count of time periods for which both recall and precision exceed the specified threshold on the validation set.

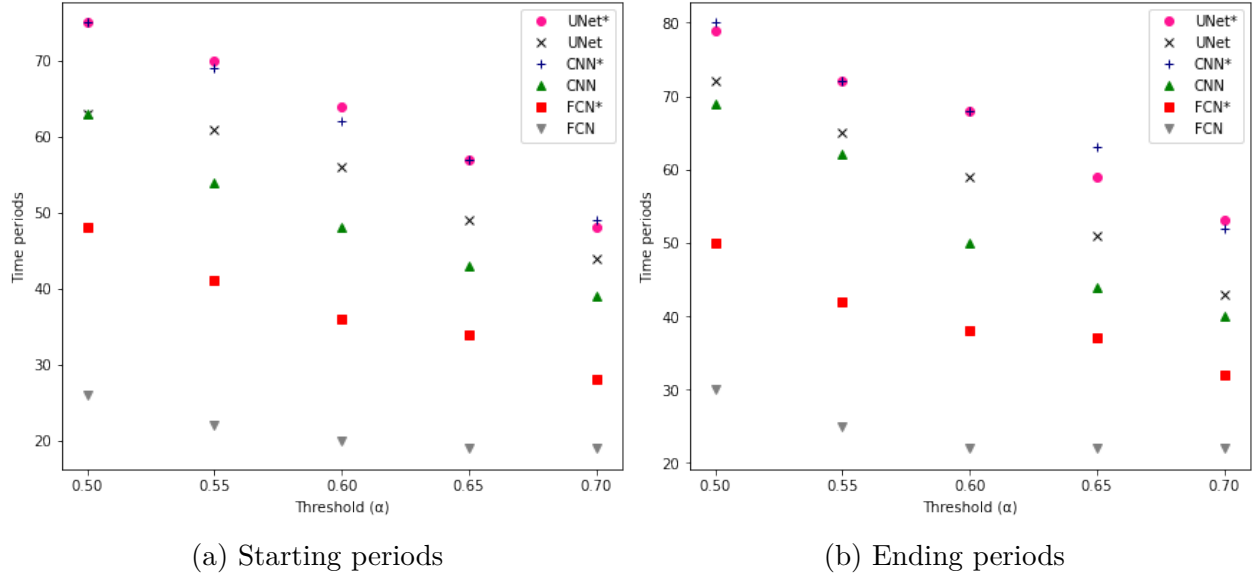


Figure 4.3 Sensitivity of predictors to threshold levels (α)

As α increases, the number of time periods that meet the high recall and precision criteria decreases across all models, demonstrating the trade-off between model stringency and element-wise prediction reliability. For instance, at $\alpha = 0.5$, the precision and recall of the UNet* model — calculated separately for starting and ending periods — are greater than or equal to this threshold for 75 starting periods and 79 ending periods. This means that at this level of reliability, the model's output is trusted only for those specific periods, while the remaining 21 and 17 periods are treated as potential starting and ending times, regardless of the predictor's output. In particular, the UNet* and CNN* models consistently achieve higher counts for all tested thresholds, indicating robust performance.

To show how the value of α impacts the performance of the optimization heuristics, we conducted a sensitivity analysis. First, we solved all 140 artificial test instances using an (almost) exact algorithm, i.e., using the MILP model with fully enumerated shifts and an

optimality gap tolerance of 0.01%. Then, we solved again these instances with MILP-CNN*, MILP-FCN*, and MILP-UNet* using different values of α , ranging between 50% and 70%. The results of these experiments are summarized in Figure 4.4. The box plots in Figure 4.4a provide the percentage of the reduction in the number of variables achieved by the predictors compared to the full model (the higher the better). Figure 4.4b reports the difference in percentage between the values of the solutions computed by the heuristics and the exact algorithm, called the error (the lower the better). Finally, Figure 4.4c specifies the speedup in computational time achieved by the heuristics (the higher the better).

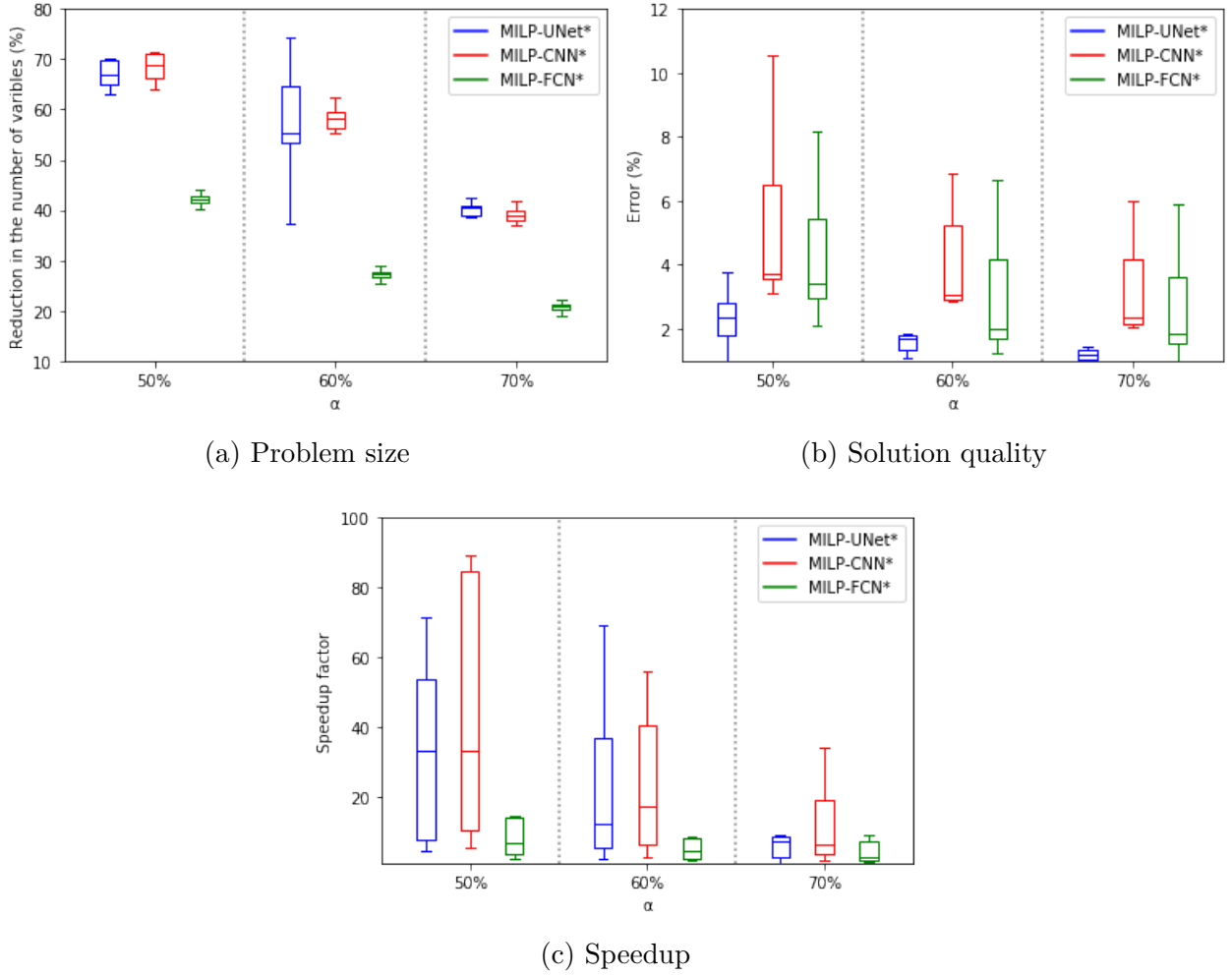


Figure 4.4 Sensitivity of optimization heuristics to α

These results show that setting a larger threshold value for the acceptance of the predictor's output in the optimization heuristics yields, as expected, a better solution quality but at the expense of a smaller computational time speedup. MILP-FCN* reduces fewer variables which indicates the recall or precision of the classifier is less than α for most of the periods and

consequently its output is not accepted. A larger reduction in the number of variables and less error of MILP-UNet* shows its better performance than the other two heuristics, which is consistent with the performance of the predictors observed in Figure 4.2. We conducted the remaining tests using MILP-UNet* and MILP-UNet. Furthermore, to obtain a good speedup, we set $\alpha = 50\%$.

4.5.3 Evaluation of the optimization heuristics

In this section, we evaluate the performance of the proposed optimization heuristics on the reference dataset by comparing the following eight different algorithms:

1. Exact: Exact algorithm where the MILP model (4.1)–(4.11) is solved considering all possible shifts, a tolerance on the optimality gap of 0.01%, and a time limit of 20,000 seconds;
2. Tol-1%: Same as Exact except that the optimality gap tolerance is set to 1%;
3. Tol-5%: Same as Exact except that the optimality gap tolerance is set to 5%;
4. LPF: A heuristic that i) solves the linear relaxation of the MILP model with the complete set of shifts, ii) fixes to 0 all $z_{s,e,j}$ variables taking value 0 in the linear relaxation solution, and iii) solves the resulting model with an optimality gap tolerance of 1%;
5. PRR: The parallel ruin and recreate heuristic of the study in [84];
6. MILP-UNet: The proposed heuristic described in Section 4.4.5 that filters out shifts by applying a UNet ML model considering only the X_1 and X_2 feature vectors;
7. MILP-UNet*: Same as MILP-UNet except that the X_3 and X_4 feature vectors are also considered in the UNet model;
8. LPF-UNet*: Same as MILP-UNet* except that all $z_{s,e,j}$ variables that take a positive value in the linear relaxation solution cannot be filtered out.

Note that we have not considered simple heuristics that would use a coarser discretization of the shift start/end times and shift durations (for instance, at every 30 or every 45 minutes, instead of every 15 minutes) because the study in [84] showed that they were outperformed by PRR.

The 14 real-life instances of the reference dataset were solved using each of these algorithms. We report the results of these experiments in Tables 4.9 to 4.11. For each instance and each algorithm except the Exact algorithm, Table 4.9 provides the increase (in percentage) of the cost of the computed solution with respect to the cost of the solution obtained by the Exact algorithm. For each instance and each algorithm, Table 4.10 indicates the computational time (in seconds) required by each algorithm. For each instance, Table 4.11 specifies in the second and third columns the total numbers of variables and constraints in the MILP model (4.1)–(4.11), when all shifts are considered in the algorithms Exact, Tol-1%, and Tol-5%. The last four columns give the percentage of variables that are removed by the corresponding heuristic. Note that no such statistic is reported for the PRR heuristic as it builds the shifts dynamically and does not rely on a MILP model. In all these tables, averages over all instances are reported in the last row and the best result for each instance and on average is highlighted in bold.

From these results, we make the following observations. The computational times required by the Exact algorithm clearly show the need to develop a more sophisticated exact solution algorithm or to resort to a heuristic for solving the instances in acceptable computational times (say, less than one hour for the largest instances). In particular, we observe that the 20,000-second time limit is reached for 4 of the 14 test instances. A simple heuristic consists of using the same MILP model (with all shift variables) and running the same algorithm but increasing the optimality gap tolerance as in algorithms Tol-1% and Tol-5%. With these heuristics, the average computational time decreases but remains large for some of these instances. Furthermore, the solution costs become less controllable with a larger tolerance, yielding an average cost increase of 1.87% for heuristic Tol-5%.

The heuristic LPF removes the largest number of variables in all instances. It is the fastest for most instances but requires a relatively large amount of time for instance 13 (an instance containing several jobs with the same skill requirements, incurring high symmetry in the branch-and-cut algorithm). Furthermore, it yields the largest average cost increase (2.72%). As for PRR, the average quality of its solutions is similar to that of those produced by LPF, but with a larger variance. It is, however, faster on average, especially because it terminates very rapidly for instance 13 with the worst solution among all heuristics.

The last three heuristics apply ML to filter out variables. In particular, the heuristics MILP-UNet and MILP-UNet* are identical except that the latter considers additional features collected from the linear relaxation solution. We observe that this is highly profitable given that it allows removing more variables on average (74.3% versus 71.2%) yielding faster average computational times (174 versus 262 seconds). Furthermore, the average cost increase is

Table 4.9 Cost increase (in %) with respect to best solution cost

Inst.	Tol-1%	Tol-5%	LPF	PRR	MILP-UNet	MILP-UNet*	LPF-UNet*
1	0.1	0.1	4.5	1.1	0.6	0.2	0.2
2	0.2	0.2	4.0	2.1	4.0	2.3	0.4
3	0.2	4.7	2.0	6.9	1.1	1.1	0.3
4	0.4	2.0	2.2	2.3	1.0	0.4	0.3
5	0.4	0.4	1.8	4.4	3.6	2.1	0.2
6	0.6	2.0	3.9	3.1	1.3	0.8	0.6
7	0.6	1.3	2.0	2.2	2.3	1.2	0.7
8	0.9	3.2	1.7	3.4	1.4	1.3	0.6
9	0.8	0.8	4.5	2.4	0.7	0.5	0.1
10	0.9	0.9	1.4	2.9	1.4	0.9	0.0
11	0.2	1.7	3.8	1.3	0.4	2.3	0.1
12	0.1	4.8	1.9	0.0	0.7	2.9	1.0
13	0.0	2.5	4.0	5.4	5.1	1.3	0.0
14	0.7	1.6	0.4	0.1	0.7	0.6	0.7
Avg.	0.44	1.87	2.72	2.69	1.74	1.28	0.37

reduced by 26%. On the other hand, LPF-UNet* retains only a few more variables than MILP-UNet* but yields much higher quality solutions (with an average cost increase of only 0.37%) in a larger average computational time (674 versus 174 seconds). This much larger average time is again due to the effort spent solving instance 13.

Finally, we observe that LPF-UNet* produces solutions of similar quality than those computed by the Tol-1% heuristic, but in a much faster average computational time (674 vs 3,589 seconds). This is not surprising as they both achieve the same linear relaxation bound and apply the same 1% optimality gap tolerance but LPF-UNet* considers much less variables after solving the root node linear relaxation.

To summarize, Figure 4.5 represents the performance of each heuristic with a point in a Cartesian coordinate system where the (broken) x -axis and the y -axis correspond to the average computational time and average cost increase, respectively. This plot allows us to clearly identify two Pareto-optimal algorithms. MILP-UNet* yields the smallest average computational time (174 seconds) and an average cost increase of 1.28%. On the other hand, LPF-UNet* produces the smallest average cost increase (0.37%) in an average computational time of 674 seconds. All other heuristics are dominated by these two algorithms. Therefore, if short computational times are sought, the heuristics MILP-UNet* and LPF-UNet* proposed in this paper and relying on an ML model to reduce the number of variables seem to be the most appropriate.

Table 4.10 Computational times (in seconds)

Inst.	Exact	Tol-1%	Tol-5%	LPF	PRR	MILP-UNet	MILP-UNet*	LPF-UNet*
1	86	33	33	3	31	9	9	11
2	316	103	103	8	15	14	18	19
3	119	48	40	3	48	10	11	22
4	13,349	851	217	11	117	107	97	98
5	335	121	121	5	74	21	24	38
6	5,575	581	204	11	166	60	39	107
7	20,000	3,416	2,047	19	134	690	372	418
8	20,000	18,170	376	24	330	240	233	554
9	2,147	295	295	12	138	46	63	82
10	3,112	701	704	52	179	144	108	285
11	4,296	3,134	517	131	109	269	91	94
12	20,000	12,442	1,873	268	242	927	280	1,283
13	20,000	6,713	6,175	3,431	272	852	907	6,109
14	6,129	3,643	2,480	11	828	273	180	322
Avg.	8,247	3,589	1,084	285	192	262	174	674

4.5.4 Out of distribution results

So far, we have performed computational experiments on instances that share the same distribution with the training dataset. The motivation behind investigating out-of-distribution datasets is to account for scenarios that companies might encounter when the distribution of upcoming instances differs substantially from the training dataset which usually involves only instances from the same company. As the proposed method would not be trained continuously over time but rather only when regularization adjustments become necessary, it is important to evaluate the resilience of this method when facing a significant parameter distribution change. We have generated a new set of instances (10 instances for each of the 14 original instances) using the procedure described in Section 4.4.1 except for the following three aspects:

1. Instead of perturbing the demand on only 10% to 20% of the periods, demand noises are applied to all periods (when the job is active);
2. A larger variance is selected for the Gaussian noise, i.e., $\sigma = 0.4 \times d_p^j$ instead of $\sigma = 0.2 \times d_p^j$;
3. A positive noise is always added as follows: $d_p^j \leftarrow d_p^j + \lceil \mathbf{x} \rceil$, where $\mathbf{x}$ is sampled from a random distribution $\mathcal{N}(0, \sigma^2)$. This yields an increased demand;
4. To perturb the employee availability, we add $\lceil 40\% \times |E| \rceil$ employees to the available employees.

Table 4.11 MILP model size

Inst.	Exact/Tol-1%/Tol-5%		Percentage of removed variables (in %)			
	Constraints	Variables	LPF	MILP-UNet	MILP-UNet*	LPF-UNet*
1	3,028	200,587	86.5	66.6	70.7	67.8
2	3,228	132,519	79.4	64.9	65.5	65.4
3	3,368	222,366	87.6	63.9	68.1	68.1
4	3,768	238,333	88.3	66.9	69.2	69.1
5	6,056	444,290	87.8	65.7	69.9	69.9
6	6,316	270,375	79.8	63.3	66.5	62.2
7	6,456	475,964	88.5	70.3	71.9	71.8
8	7,256	487,176	88.6	65.8	69.1	69.0
9	7,220	1,043,674	94.7	78.6	80.4	80.4
10	7,720	2,032,079	97.2	80.1	82.5	82.5
11	10,148	1,363,589	94.0	79.7	82.2	82.2
12	10,888	2,646,682	96.9	82.1	84.7	84.7
13	12,632	1,577,112	93.0	68.4	74.6	74.5
14	14,440	3,375,548	96.7	81.1	84.2	84.2
Avg.			89.9	71.2	74.3	73.7

Additionally, we have constructed two sets of large instances by combining the instances described above. These instances include 13 and 14 jobs performed by 167 and 202 employees.

The mean and standard deviation of the demand per job and period (excluding the overnight periods when a job is inactive) in this new dataset and the training dataset is (3.12, 2.92) and (2.37, 2.07), respectively. To determine whether these means are equal or significantly different, we implemented a two-sample Student's t -test, where the null hypothesis was "the two means are equal" and the alternative hypothesis was "the two means are different". The p -value obtained from this test is $1.4e-15$. Assuming a significance level of 0.01, this p -value indicates that the means are statistically different. Therefore, we conclude that the demand distribution of the new dataset is significantly different from that of the training dataset.

These 160 out-of-distribution test instances are grouped into 16 sets of 10 instances and are denoted O1, O2, ..., O16. Each of these instances was solved using three algorithms: Exact, MILP-UNet*, and LPF-UNet*. The statistics of the instances and the results of conducted experiments are reported in Table 4.12. Each line indicates the average results obtained for an instance set, namely, the average time (in seconds) required by each algorithm as well as the average increase (in percentage) of the solution cost obtained by the two heuristics compared to the optimal value computed by the Exact algorithm. Additionally, each row provides insights into the number of decision variables of the Exact method and the percentage of variables removed (as shown in the table as "Rem. Vars.") by the two heuristics. The last

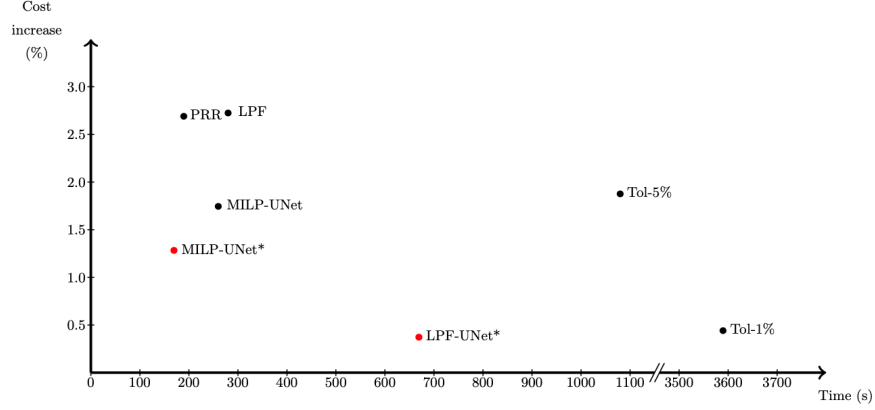


Figure 4.5 Heuristic performance summary

row reports averages over all instances.

From these results, we first observe that these instances include a range of small, medium, and large-sized problems. The Exact algorithm can solve 13 of them within the 20,000-second time limit whereas this limit is reached for 3 instances resulting in an average computational time of 4,949 seconds. Next, we observe that both MILP-UNet* and LPF-UNet* heuristics significantly reduce the number of decision variables that remain in the optimization model, with an average reduction of 78.3% and 76.1%, respectively. MILP-UNet* and LPF-UNet* drastically reduce the computational time, by speed factors of 11.5 and 8.7, respectively. Like with the original instances, LPF-UNet* filters out slightly fewer variables and requires larger computational times than MILP-UNet* but produces better quality solutions. Overall, we can say that the performance of these two heuristics is maintained on these out-of-distribution instances. The slight deterioration of the solution quality indicates, however, that training does not need to be performed continuously, but only occasionally when there is a substantial change in employee availability/skills or in the job demand patterns.

4.6 Conclusion

In this work, we have studied a flexible personnel scheduling problem that involves a large number of potential work shifts. To efficiently solve this problem, we have proposed three ML models to predict the starting and ending times of the shifts and use these predictions to filter out from a MILP model shift variables associated with shifts that have a low probability of being selected. Different heuristics based on this approach were devised and two of them provided Pareto-optimal results on 14 real-world test instances. Indeed, the MILP-UNet*

Table 4.12 Results on the out-of-distribution instances

Datasets			Exact			MILP-UNet*			LPF-UNet*		
Instant	Jobs	Employees	Variables	Constraints	Time (s)	Rem. Vars. (%)	Time (s)	Cost increase (%)	Rem. Vars. (%)	Time (s)	Cost increase (%)
O1	2	24	308,340	3,168	145	75.0	30	3.4	70.3	18	0.9
O2	2	38	316,781	3,448	110	73.6	19	1.2	69.2	29	1.5
O3	2	48	462,368	3,648	189	73.3	23	2.2	70.3	30	0.8
O4	2	76	603,688	4,208	240	71.2	91	1.2	69.0	52	1.2
O5	4	48	886,236	6,336	906	75.4	59	1.5	72.2	72	1.0
O6	4	66	919,306	6,696	556	74.6	52	1.8	71.7	91	1.0
O7	4	76	1,187,424	6,896	900	74.8	76	1.5	72.5	129	1.1
O8	4	132	1,799,265	8,016	3,591	72.5	190	1.5	71.1	409	1.2
O9	5	35	862,201	7,420	199	83.6	21	1.4	83.6	29	0.1
O10	5	70	1,748,778	8,120	887	84.8	60	1.1	84.8	72	0.4
O11	7	52	1,815,608	10,448	947	85.3	65	0.9	83.0	89	1.3
O12	7	104	3,544,740	11,468	6,909	85.6	281	1.3	85.6	331	0.6
O13	8	132	3,992,496	13,392	20,000	75.8	2,563	0.7	74.4	2,978	0.5
O14	10	70	4,093,804	14,840	3,590	85.7	151	0.9	82.6	332	0.4
O15	13	167	6,686,251	20,792	20,000	79.4	2,038	0.0	76.3	2,529	-0.3
O16	14	202	8,967,208	22,836	20,000	82.5	1,190	0.7	80.4	1,879	0.0
Avg.					4,949	78.3	432	1.33	76.1	566	0.73

heuristic was the fastest with an average speedup factor of 47 compared to an Exact algorithm but yielded an average cost increase of 1.28%. Even when compared to the state-of-the-art neighborhood-based heuristic PRR, MILP-UNet* shows a much more robust behavior being capable of finding better quality solutions in similar computing times. On the other hand, the LPF-UNet* heuristic was able to compute high-quality solutions with an average cost increase of 0.37% but was slower with a speedup factor of 12. These results clearly show the potential to design efficient solution approaches by combining state-of-the-art techniques from ML and mathematical programming.

For future research, we envision using ML to select from the history (previous weeks/months) a small subset of shifts or even complete schedules for some employees (e.g., the regular employees) or some jobs and to restrict the optimization model to the selected shifts/schedules for these employees and jobs. In fact, for certain jobs, the demand curves are highly similar from one week to another and they are often covered by the same employees. Identifying recurrent demand patterns from historical data might be exploited to identify promising candidate shifts for some jobs and employees. This approach holds promise for expediting the solution process while maintaining solution quality. Furthermore, we anticipate extending these investigations to encompass scenarios involving multi-job shifts and breaks.

CHAPTER 5 ARTICLE 2: DATA MINING-DRIVEN SHIFT ENUMERATION FOR ACCELERATING THE SOLUTION OF LARGE-SCALE PERSONNEL SCHEDULING PROBLEMS

F. Rastgar-Amini, D. Aloise, C. Contardo, and G. Desaulniers, “Data mining-driven shift enumeration for accelerating the solution of large-scale personnel scheduling problems”, This paper was submitted on December 1, 2023, and published on December 9, 2024, *ACM Transactions on Evolutionary Learning*, vol. 5, no. 4, pp. 1–24, 2025.

5.1 Introduction

The focus of our research is flexible personnel scheduling problems in the service industry, particularly large retail stores. A personnel scheduling procedure determines the least-cost work schedule for employees who are required to cover the demand of multiple jobs during a planning horizon [e.g., 96]. The demand is expressed for each time period and determines how many employees are needed to fulfill a job according to the presence of customers. Unlike fixed-shift scheduling in other environments such as manufacturers, hospitality, and healthcare, the service industry requires greater flexibility due to varying customer arrivals throughout the day. In hospitals, for instance, employees typically adhere to 8-hour or 12-hour shifts, with predetermined and fixed start times during the week (e.g., 7 a.m., 3 p.m., and 11 p.m.). In contrast, within the service sector, shifts may start and end at various times throughout the day, aiming to create cost-effective schedules that require the fewest possible employees at any given moment. This means that instead of assigning only a few shift options to employees, such as 3 per day, as seen in nurse scheduling, the variety of potential shifts must be significantly high. For instance, there could be over 900 possible shifts in a single day if the scheduling window spans from 7 a.m. to 10 p.m., with shifts starting and ending every quarter of an hour and lasting anywhere from 4 to 9 hours. Furthermore, the demand for staffing is not defined on a per-shift basis but rather for each time period, often in 15-minute intervals. This high degree of flexibility results in a enormous number of potential shift combinations, rendering the scheduling problem exceptionally complex.

In the context of a retail store, the number and type of employees required to perform various jobs in the store tend to follow a similar or repetitive pattern over time. For example, many retail stores experience a consistent pattern of lower customer arrivals on Mondays compared to other days of the week. On the other hand, Saturdays are typically one of the busiest days for retail stores, as many people have the weekend off and are more likely to go shopping.

This creates a consistent pattern of high customer demand on Saturdays. In practice, some employees are assigned to similar schedules most of the time. Therefore, it might be possible to allocate consistent or even fixed shifts to a group of employees. This helps to reduce the problem size by limiting the number of employees/jobs to be scheduled. In light of these insights, we were motivated to initiate a study exploring the utilization of historical scheduling data for future planning horizons.

The field of personnel scheduling research encompasses a range of research methodologies, incorporating distinct analytical approaches coupled with solution or evaluation techniques. Among them, methods based on mixed-integer linear programming (MILP) stand out as the most prevalent approaches in the scientific literature [see, e.g., 2, 11]. In this paper, we focus on a personnel scheduling problem variant that can be formulated as a set-covering model with explicitly defined shift variables. First introduced by the study in [5], these models use binary variables to represent potential shifts, allowing for a more efficient representation of the scheduling problem. When using commercial solvers to solve these problems, they struggle to handle the intricacies of flexible personnel scheduling in the service industry due to the large number of potential shifts originating from the dynamic nature of demand. Heuristics can, therefore, be useful to reduce the model size, such as the proposed approach to fix the schedules of a group of employees.

In this study, we contribute to accelerating the solution of flexible personnel scheduling problems by using data mining methods in a pre-processing step to extract information from previously solved problem instances and use this information to reduce the problem size. In particular, we identify similar instances in historical data while identifying outliers. Thus, we streamline the enumeration of shifts to align with the patterns of similar past solved instances. While the integration of data mining and knowledge extraction in optimization is promising, ensuring the quality and relevance of mined data, as well as managing the computational complexity of knowledge extraction and utilization, are challenging. Therefore, we implement essential data pre-processing methods and utilize the simple yet effective k -Nearest Neighbors (k NN) algorithm to find similar data points. The proposed data mining-driven optimization algorithm is tested on a series of artificially generated instances from a set of eight real-world instances provided by our industrial partner UKG, which commercializes personnel scheduling optimization software.

The remainder of this paper is structured as follows. Section 5.2 discusses related works. In Section 5.3, the problem under study is defined, and the MILP model employed in the optimization phase is introduced. The methodology and approach considered in this study are outlined in Section 5.4. Section 5.5 presents the experimental results. Finally, Section 5.6

concludes with a brief summary and discusses potential extensions.

5.2 Related works

This literature review is divided in two parts: learning-based optimization algorithms and personnel scheduling. In the latter part, we mainly focus on works involving a data mining/machine learning (ML) component.

5.2.1 Learning-based optimization algorithms

Data mining and knowledge extraction have the potential to enhance optimization techniques by leveraging historical data, patterns, and insights. Researchers have applied data mining techniques such as clustering and classification to discover patterns and relationships within the input data of optimization problems, leading to more efficient search strategies and improved decision-making [65].

Such ideas have been largely explored, specially in the metaheuristic community aiming to create new solutions from good fragments (or blocks) of high quality solutions previously found and stored during the search. In fact, recognizing recurring patterns that often emerge in a collection of solutions can be effectively cast as a frequent pattern mining task [66]. In the paper [67], frequent patterns are mined by the FPmax* algorithm of the study in [68] and used for solution re-initialization within the hybrid genetic algorithm of the study in [69], thus guiding the search through more promising regions in the solution space. In paper [70], the FPmax* algorithm is also used to mine patterns from the population of a genetic-based search, but is used to guide, in this case, the offspring solution construction. By integrating mathematical optimization with an evolutionary genetic approach, the study in [71] introduce a "reduce-solve-combine" memetic search method. In each generation, a common pattern mined from two parent solutions is first used to reduce the original instance. The reduced instance is then solved, and an offspring solution is finally obtained by combining the mined common pattern with the solution of the reduced instance.

The author of the study [72] provides a comprehensive taxonomy of how data mining/ML can be integrated into metaheuristics to improve optimization outcomes. Notably, the survey highlights the use of data mining/ML techniques for problem decomposition, which is especially relevant for tackling complex and large-scale problems. The author discusses various supported approaches for breaking down optimization problems into smaller, more manageable subproblems, which can be solved more efficiently. Three primary decomposition strategies are commonly employed: (1) *data space decomposition*, where input data is parti-

tioned (e.g., using clustering techniques) to form smaller subproblems which are then solved and their solutions combined to form a final complete solution [see, e.g., 73–75]; (2) *decision space decomposition*, which divides decision variables into subsets, reducing interdependencies and enabling focused problem-solving; this approach relates with the frequent pattern mining methods previously discussed; and (3) *objective space decomposition*, particularly useful in multi-objective optimization, where data mining/ML reduces or clusters objectives, thus increasing the efficiency of computing dominance relations, and guiding the search process [see, e.g., 76–78].

In recent years, the mathematical optimization community has been increasingly interested, as well, in using data mining and ML techniques to improve the performance of mathematical programming solvers. The study in [55] provide a survey of the potential contribution of ML to the combinatorial optimization community. ML methodologies prove effective in addressing extensive continuous and combinatorial problems by acquiring and leveraging problem structures [see, e.g., 57], predicting optimal solutions through the analysis of data extracted from previously solved instances [see, e.g., 58, 63], reducing problem sizes [see, e.g., 62, 79–81], and enhancing the performance of established optimization algorithms [see, e.g., 59–61, 82]. Recent advances have also focused on using machine learning to reduce the complexity of optimization problems by predicting and fixing certain decision variables. For example, the study in [83] propose a learning-based approach for repeatedly solving routing problems that are similar (e.g., only customer demands may vary from one problem to another), particularly the capacitated vehicle routing problem (CVRP). Their method builds a classifier using historical solutions to predict and fix the values (0 or 1) of variables that are likely to take the same value in the reference solution and in an optimal solution for the current problem to solve, thereby reducing the search space and accelerating the problem-solving process. This approach shows significant computational savings while maintaining solution quality, especially for scenarios where problem data vary slightly but retain a common structure. The methodology that we propose can be seen as similar to that of the study in [83] as it fixes the values of some variables to reduce the search space. On the other hand, we only fix variables to 0 and we address a problem that is very different from a vehicle routing problem.

5.2.2 Personnel scheduling

Addressing personnel scheduling problems often requires navigating complex, high-dimensional search spaces, where traditional methods may prove insufficient to find optimal solutions. To overcome this scalability issue, a wide range of mathematical and learning-based optimization algorithms as well as metaheuristics have been developed as highlighted by the study

in [11]. For example, decomposition methods have been developed for large-scale personnel scheduling problems [see, e.g., 26, 33, 38]. Nevertheless, when dealing with practical personnel scheduling issues, numerous constraints related to shift feasibility and various complex cost structures pose challenges. Consequently, designing pricing problems for generating shifts dynamically within a column generation algorithm becomes a challenging task in such problems.

The use of data mining and knowledge extraction in personnel scheduling problems, specifically in the context of the retail industry, has been also quite limited in the literature. The study in [97] introduce a pattern recognition-based approach for solving assignment problems focusing on nurse scheduling and educational timetabling. The nurse rostering problem is a staff assignment problem involving the allocation of shifts to nurses while considering various constraints. A timetabling problem in education can be described as the process of assigning events such as exams into a limited number of periods under certain constraints. In these problems, some constraints are mandatory due to resource limitations and legal regulations, while others are preferences used for assessing schedule quality. Calculating objective values for highly constrained problems can be costly, and feasible solution spaces are limited. The study in [97] aim to predict solution quality without the need for actual objective value calculations, making the heuristic search process more efficient. This is achieved through the application of neural networks to expedite the evaluation of solutions using pattern recognition.

The study in [98] propose a decision support system for healthcare organizations that combines optimization and process mining approaches. The system utilizes event logs from the information system to simulate work schedules, considering operative constraints, personal preferences, and regulations. The authors emphasize the importance of recognizing patterns in realized rostering plans as a means to understand personnel needs and habits.

The methodologies presented by the studies in [98] and [97] are designed to address complexities arising from numerous and complex constraints rather than focusing on flexibility in shift enumeration. These challenges emerge particularly in scenarios where staff preferences play a crucial role, yet the available working shifts are constrained to a few fixed-shift schedules.

To address scalability concerns in personnel scheduling problems, driven by a large number of variables, the study in [96] introduce a deep learning-based approach. This approach predicts and eliminates variables unlikely to contribute to an optimal solution. The method predicts the start and end times of shifts, using these predictions to exclude shift variables associated with shifts that are not likely to be selected in the optimal solution of a MILP model. Notably, this method restricts the enumeration of potential shifts to those starting

and ending at the predicted times. This approach could be adapted to environments where repeating historical schedules is desirable. The key distinction with our work lies in the fact that the method proposed here does not treat new instances as entirely distinct scenarios. Instead of predicting future shifts, the goal is to identify similar shifts within historical data.

5.3 The flexible personnel scheduling problem in retail

In this section, first, the description of the studied flexible personnel scheduling problem is provided (Section 5.3.1). Following that, Section 5.3.2 introduces the notation utilized in the formulated mathematical model, which is subsequently detailed in Section 5.3.3.

5.3.1 Problem statement

This study revisits a problem previously addressed by the studies in [84] and [96]. This problem encompasses several variants, including the consideration of multiple jobs, the use of flexible shifts, and a one-week scheduling horizon, in addition to requiring employees to have at least n^O days off per week. It is assumed that shifts are mono-jobs, i.e., employees are assigned to the same job throughout the shift, but an employee can be assigned to different jobs in different shifts as long as they are qualified for those jobs. It is common practice to adopt mono-job shifts as a way to simplify operational processes, especially in personnel management, and eliminate the need for short breaks when changing tasks. Although our simplified approach makes our approach clearer, we recognize that multi-job scenarios may require more complex decision-making processes in the future.

We do not incorporate breaks into our problem formulation, reflecting real-world scenarios in the retail industry in which breaks are imposed instantly by immediate work demands. The forecasted demand is normally adjusted during certain intervals to compensate for the breaks added during operations. By introducing new decision criteria and considering break placement regulatory constraints, our proposed heuristic can be adapted where this no-break assumption does not hold.

We follow the set-covering model used by the studies in [84] and [96] to formulate the corresponding flexible personnel scheduling problem with side constraints as follows: there is a set of jobs denoted by J , and a set of skilled employees denoted by E , who are assigned to work on these jobs. Only a subset of employees E^j is qualified to perform job $j \in J$. There is a set of mono-job shifts S , where a shift $s \in S$ is defined by a pair of valid start and end times a_s and b_s (for example, within a quarter hour) yielding a valid duration g_s (for instance 4 to 9 hours).

The planning horizon is partitioned into a set P consisting of non-overlapping periods, typically lasting 15 minutes each. Projected demands, denoted as d_p^j for each job $j \in J$ during period $p \in P$, are forecasted based on expected sales or transactions two or three weeks prior to the planning horizon. The planning horizon $H = \{1, 2, \dots, 7\}$ is further segmented into seven days. Due to variations in job demands, such as differing start times, the set of potential shifts S^j for a job $j \in J$ may vary. Moreover, adhering to real-world scenarios, we assume no isolated demand spans less than the minimum shift duration. Thus, a shift is considered a *candidate* for a job if it covers periods with non-zero demand for that job, ensuring feasibility. Employee assignments to a subset of shifts S^e are contingent on their skills and availability throughout the planning horizon.

The scheduling problem seeks to allocate employees to shifts and jobs, ensuring the number of assigned employees matches the job's demand at each period while respecting specific working rules. These rules include (1) enforcing a minimum rest time of n^R periods between two consecutive shifts for an employee and (2) requiring a minimum number of days off, denoted as n^O , for each employee. However, constraints related to employee qualifications and time availability may pose challenges in adequately covering all demands. Consequently, shifts with elevated costs can be scheduled anonymously and later assigned to temporary workers to meet demand. Furthermore, due to the minimum shift duration requirement, there might be periods without demand that require coverage, or more employees might be assigned than demanded for a specific period, resulting in over-coverings. A penalty is imposed on Over-covered periods because employees are considered unproductive during over-covering time periods. To ensure an equitable distribution of over-coverings across periods, a non-decreasing step-wise function is used to calculate the over-covering penalty.

The objective function includes labor costs. The remuneration of the employees is based on their hours worked but it is not employee-dependent like in practice (otherwise, senior employees would work less than junior ones which are usually paid at a lesser rate). The employees are divided into different levels based on the hours they work, and each level has a different hourly rate. As an illustration, consider a scenario where every 8 hours of work corresponds to a payment level. In this case, an employee working for 20 hours would receive compensation at three levels. Specifically, they would be paid c_1^E per period for the initial 8 hours, c_2^E per period (where $c_2^E > c_1^E$) for the subsequent 8 hours, and c_3^E per period (where $c_3^E > c_2^E$) for the remaining 4 hours. This step-wise structure is considered to favor a balanced distribution of the work hours between the employees.

The model being proposed in Section 5.3.3 is said to be explicit because it requires listing all possible shifts, which can be very numerous due to the shifts' flexible start time, end time,

and duration. To illustrate, if shifts can begin at any 15-minute interval and last anywhere between 3 to 8 hours in 15-minute increments and if the demand of a job for a day spans from 06:00 to 22:00, we get a total of around 1100 possible shifts for this job and day.

5.3.2 Notation

Tables 5.1 to 5.3 in this section present the notation used for sets, subsets, parameters, and decision variables in the mathematical model.

Table 5.1 Sets and subsets

Set	Description
S	Set of enumerated shifts
P	Set of time periods in the planning horizon
E	Set of employees
J	Set of jobs
H	Set of total days in the planning horizon
S^j	Subset of shifts that are candidates for job $j \in J$
S^p	Subset of shifts that cover period $p \in P$
S^h	Subset of shifts that start on day $h \in H$
S^e	Subset of shifts that can be assigned to employee $e \in E$
E^j	Subset of employees that are qualified for job $j \in J$
J^e	Subset of jobs that employee $e \in E$ is qualified for
E_p^j	Subset of employees that are qualified for job j and available at period p
F	Set of steps for the over-covering penalty step-wise function
Q	Set of steps for the employee remuneration step-wise function

Table 5.2 Parameters

Parameter	Description
d_p^j	Number of employees needed for job $j \in J$ at time period $p \in P$
a_s, b_s	Starting and ending periods of shift $s \in S$
g_s	Length (in number of periods) of shift $s \in S$
n_e^O	minimum number of days off for employee $e \in E$
n^R	Minimum rest periods between two consecutive shifts
c_s^A	Cost of an anonymous shift $s \in S$ per period
c_q^E	Employee remuneration per period on step $q \in Q$
m_q	Maximum number of periods on step $q \in Q$
c_k^V	Cost per over-covering on step $k \in K$
m_f	Maximum number of over-coverings on step $f \in F$

Table 5.3 Decision Variables

Variable	Description
$z_{s,e,j}$	Binary variable equal to 1 if shift $s \in S^j \cap S^e$ is assigned to employee $e \in E$ for job $j \in J$
$u_{s,j}$	Number of anonymous shifts $s \in S$ assigned for job $j \in J$
$v_{p,j,f}$	Number of over-coverings for job $j \in J$ at period $p \in P$ and on step $f \in F$
$w_{e,q}$	Number of periods worked by employee $e \in E$ on step $q \in Q$
$o_{e,h}$	Binary variable equal to 1 if employee $e \in E$ is off on day $h \in H$

5.3.3 Mathematical model

The mathematical model for the flexible personnel scheduling problem in this research is formulated similarly to the study in [96] as the following integer program:

$$\min \quad \sum_{j \in J} \sum_{s \in S^j} c_s^A u_{s,j} + \sum_{e \in E} \sum_{q \in Q} c_q^E w_{e,q} + \sum_{j \in J} \sum_{p \in P} \sum_{k \in K} c_k^V v_{p,j,f} \quad (5.1)$$

$$s.t. : \quad \sum_{s \in S^p \cap S^j} u_{s,j} + \sum_{e \in E_p^j} \sum_{s \in S^e \cap S^p \cap S^j} z_{s,e,j} - \sum_{f \in F} v_{p,j,f} = d_p^j \quad \forall p \in P, j \in J \quad (5.2)$$

$$\sum_{j \in J^e} \sum_{s \in S^e \cap S^j \cap S^h} z_{s,e,j} + o_{e,h} = 1 \quad \forall e \in E, h \in H \quad (5.3)$$

$$\sum_{j \in J^e} \sum_{s \in S^e \cap S^j} g_s z_{s,e,j} - \sum_{q \in Q} w_{e,q} = 0 \quad \forall e \in E \quad (5.4)$$

$$\sum_{h \in H} o_{e,h} \geq n_e^O \quad \forall e \in E \quad (5.5)$$

$$\begin{aligned} & \sum_{j \in J^e} \sum_{s \in S^e \cap S^j \cap S^{h+1}} a_s z_{s,e,j} + M o_{e,h+1} \\ & - \sum_{j \in J^e} \sum_{s \in S^e \cap S^j \cap S^h} (b_s + 1 + n^R) z_{s,e,j} \geq 0 \quad \forall e \in E, h \in H \setminus \{7\} \end{aligned} \quad (5.6)$$

$$z_{s,e,j} \in \{0, 1\} \quad \forall j \in J, e \in E^j, s \in S^j \cap S^e \quad (5.7)$$

$$u_{s,j} \geq 0, \text{ integer} \quad \forall j \in J, s \in S^j \quad (5.8)$$

$$v_{p,j,f} \in [0, m_f] \quad \forall p \in P, j \in J, f \in F \quad (5.9)$$

$$w_{e,q} \in [0, m_q] \quad \forall e \in E, q \in Q \quad (5.10)$$

$$o_{e,h} \in \{0, 1\} \quad \forall e \in E, h \in H \quad (5.11)$$

The objective function (5.1) is designed to minimize the total costs encompassing personnel and anonymous shift costs, as well as over-covering penalties. Constraints (5.2) guarantee that job demands at each period are met through personalized or anonymous shifts, facilitating the computation for over-coverings. Constraints (5.3) mandate that each employee is assigned either to a shift or a day off during the planning horizon. Each employee's working hours in each time step $q \in Q$ are computed through constraints (5.4). Additionally, each employee e is granted a minimum of n_e^O days off, as stipulated by constraints (5.5). The big-M constraints (5.6) mandate a minimum rest time of n^R periods between two consecutive shifts assigned to the same employee on subsequent days, with $M = \max_{s \in S} \{b_s + 1 + n^R\}$. Lastly, constraints (5.7)-(5.11) identify the variables' acceptable domains.

5.4 Methodology

Our research involves an in-depth analysis of historical data related to job demand patterns and previously assigned schedules. The primary objective is to find recurrent patterns within

the historical data that bear similarity to the new problem we aim to solve. This enables us to restrict the enumeration of potential shifts, focusing on solutions derived from similar historical instances.

Our method effectively aligns with the ‘learning to prune’ framework, where ML is utilized to narrow the search space rather than directly solve the problem for its entire variable space. After the search space is reduced, it addresses the flexible personnel scheduling problem using a MILP solver.

Our proposed method have some characteristics and respects some constraints that we outline below:

- One primary constraint we face is our partner’s reliance on a MILP-based methodology and their unwillingness to adopt other search frameworks that might alter their existing workflow. This limitation restricts the reference algorithms compared in Section 5.5, although we recognize the connections between our proposed method and other frequent pattern mining techniques discussed in Section 5.2.1.
- Our pattern-mining task relies on historical data from previously solved personnel scheduling problems, rather than on patterns discovered during the search process.
- We opted for an ML model that requires minimal feature engineering, thereby allowing seamless integration with the optimization solutions currently in use. The k NN algorithm, being similarity-based, relies solely on the distances between the data entities under investigation.
- Although we have evaluated our approach using historical data consisting of exact solutions for personnel scheduling instances to better benchmark our proposed method, there is no reason why historical heuristic solutions cannot be used as well.

In this section, we start by describing a data instance of our problem (Section 5.4.1). Next, we explain how we measure the dissimilarity between data instances (Section 5.4.2). In Section 5.4.3, we describe the utilization of the k NN algorithm to find similar instances and outliers. Lastly, we describe the optimization heuristic pipeline (Section 5.4.4).

5.4.1 Instance representation

As detailed in Section 5.3, in order to address the personnel scheduling problem, the input data includes the demand for each job, as well as the number of available and skilled employees

for each time period within a day. For a one-week planning horizon, there are 672 distinct 15-minute time periods in P ($7 \text{ days} \times 24 \text{ hours/day} \times 4 \text{ periods/hour} = 672 \text{ periods}$). In Figure 5.1, we illustrate the demand curves of two different jobs, each with distinct patterns. Job 1 exhibits a steady demand pattern, possibly reflecting the need for consistent supervision in a specific section, requiring one employee per day. Conversely, Job 2 displays a more dynamic pattern with spikes, particularly on Saturdays, indicating a higher demand for employees on these occasions. However, a recurrent pattern in demand curves is evident across weekdays. In more extensive scenarios with multiple jobs, this similarity extends not only within a single job but also between different job curves.

We assume that we have access to historical instances and their solutions for a set N of previous weeks. A demand curve for each job over the course of a day in the historical data corresponds to a time series of 96 data points ($4 \text{ periods/hour} \times 24 \text{ hours}$). Let us denote $\mathcal{X}$ the set of these demand curves, where each curve $X^{(j,h,n)}$, represents the demand of employees for a particular job j on day h of week n . Thus, the total number of historical data points is equal to $|J| \times |H| \times |N|$. A function $Y : \mathcal{X} \rightarrow 2^{|S|}$ associates each $X^{(j,h,n)} \in \mathcal{X}$ with the subset of shifts used in the optimal solution of (5.1)–(5.11) to cover job j on day h during week n .

Figure 5.2 depicts the optimal schedule of 17 employees assigned to perform the two jobs of the example presented in Figure 5.1. For job 1 and job 2, the numbers of selected shifts are 21 and 47, respectively. For the demand curve $X^{(1,1,w)}$ associated with job 1 and day 1 in the week w , the set of shifts $Y(X^{(1,1,w)}) = (20, 35), (36, 56), (57, 84)$, where the shifts s are identified by their start and end periods (a_s, b_s) .

5.4.2 Dissimilarity Measure

To select potential shifts from the history, we need to determine if the demand curve for a job and a day is similar to one or several that have been seen in the historical dataset. A methodical way of quantifying similarity between demand curves is, thus, required.

Given two demand curves $X, X' \in \mathcal{X}$, their shape-based similarity can be determined by comparing local point distances. The most common distance function for quantitative data is the Euclidean distance [99]. Knowing that X and X' are of the same length (i.e., 96), and that X_p refers to the job demand at hour p , the Euclidean distance between X and X' is defined as

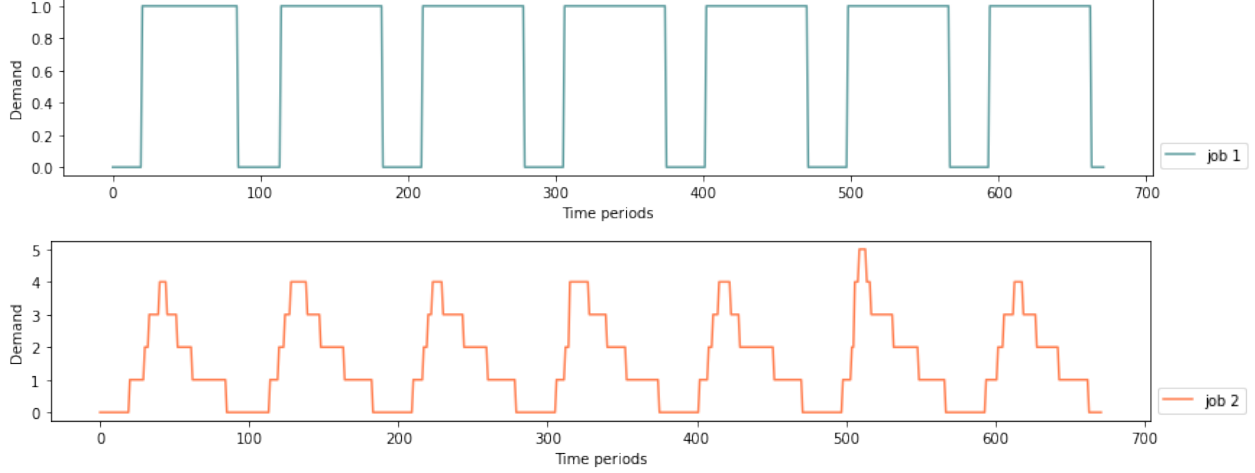


Figure 5.1 Demand curves of two example jobs

$$Dist(X, X') = \sqrt{\sum_{p=1}^{96} (X_p - X'_p)^2} \quad (5.12)$$

Our dataset composed by job demand curves presents two noteworthy aspects. Firstly, the data points exhibit varying scales, particularly between different job categories, and the Euclidean distance metric is sensitive to these scale differences. If one feature possesses a significantly larger scale than another, it can exert a disproportionate influence on distance calculations, potentially overshadowing its actual importance. Consequently, we apply data scaling to rectify this issue, ensuring that each feature contributes equally to the distance metric. In this regard, we use Min-Max normalization, which scales the data to a specific range, such as $[0, 1]$.

Secondly, the temporal nature of our data points highlights the significance of variations within each data point. Notably, any increase or decrease in demand within specific time periods signifies a crucial role in pattern extraction. For instance, we observe that shifts often start or end at a period when the demand experiences an incremental or decreasing trend, respectively. To address this issue, we implement a *Wavelet transformation* as another step of data preparation [99].

Wavelets are a widely recognized method for breaking down and summarizing time-series data into a multidimensional representation at various levels of granularity. The *Haar wavelet*

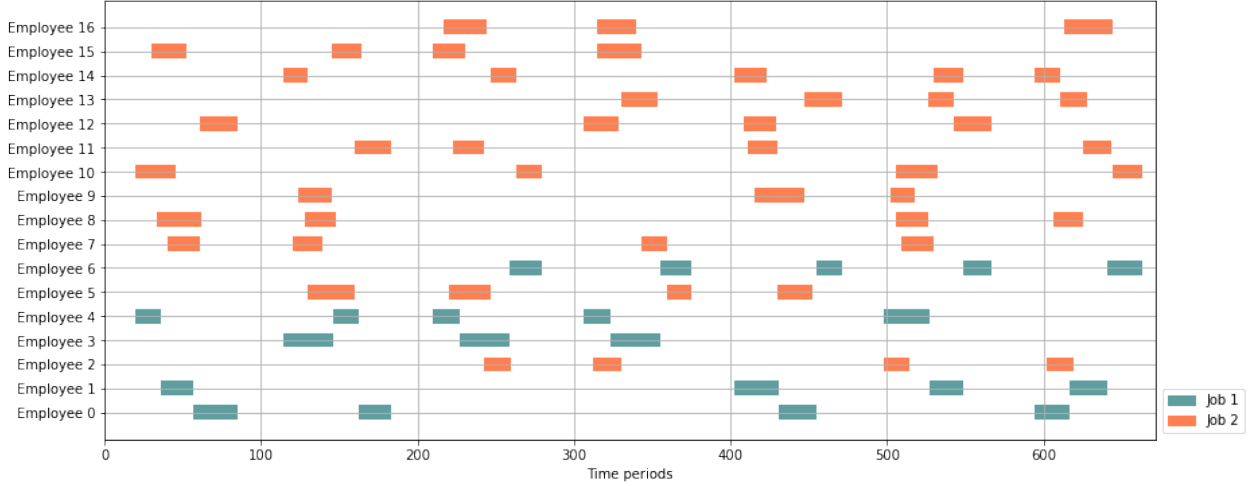


Figure 5.2 Employees' schedule assigned to jobs of example 5.1

stands out as a favored choice for this purpose due to its straightforward, intuitive characteristics and ease of application [100]. The Haar wavelet transformation works by successively dividing the time series into segments and computing the average and difference between adjacent data points within each segment. This process is repeated iteratively, creating a wavelet decomposition of the time series into approximation and detail coefficients at multiple scales. These coefficients can then be used for feature extraction and reveal underlying temporal patterns in the data, which may not be apparent in the raw time series. Euclidean distance can then be applied to the transformed data, potentially capturing a more meaningful dissimilarity between data points. Furthermore, by highlighting important temporal features, the wavelet transformation can improve the discriminative power of Euclidean distance. This can be especially valuable when we want to find nearest neighbors based on relevant temporal patterns rather than raw values.

5.4.3 K-Nearest Neighbors Algorithm

In this section, we present our approach to find similar instances using the k NN algorithm, which is a widely used supervised ML algorithm for classification and regression tasks [101]. However, it can also be adapted for similarity search, making it a powerful tool for time series analysis. The k NN algorithm is a non-parametric and instance-based learning method. Given a dataset and a new data point i , k NN identifies the k nearest neighbors to i in the dataset based on a chosen distance metric. Additionally, k NN can identify data points that are significantly dissimilar from the dataset, effectively detecting outliers. The data scaling and transformation techniques applied during our data pre-processing render the Euclidean

distance metric a well-suited choice for utilization in the k NN algorithm.

Let $\hat{n} \notin N$ be the index of the upcoming week for which we want to solve a new personnel scheduling problem. To find the demand curves in $\mathcal{X}$ that are similar to the demand curves arising from this new problem instance, we apply the following steps:

1. Test data points: We define the set of test data points for this instance as explained in Section 5.4.1, generating $|J| \times |H|$ points from the decomposition of the problem into jobs and days of the planning horizon. These data points $\hat{X}^{(j,h,\hat{n})} \in \hat{\mathcal{X}}$ are then transformed and pre-processed in the same manner as the set $\mathcal{X}$.
2. Nearest neighbor search: For each test data point $\hat{X}^{(j,h,\hat{n})} \in \hat{\mathcal{X}}$, we calculate its distance to all points in $\mathcal{X}$, using the chosen distance metric, to identify the set $\mathcal{K}^{(j,h,\hat{n})}$ containing its k nearest data points in $\mathcal{X}$.
3. Outlier detection: For each data point in $\hat{\mathcal{X}}$, we analyze the distribution of its k nearest neighbors to determine whether it is an outlier. Intuitively, a data point could be considered an outlier if the average distance from its k nearest neighbors is much larger than the average distance of the other data points with respect to their k nearest neighbors [102].

5.4.4 Optimization Heuristic

The k NN algorithm provides a set $\mathcal{K}^{(j,h,\hat{n})}$ of similar data points for each test data point $\hat{X}^{(j,h,\hat{n})} \in \hat{\mathcal{X}}$. Then, for each data point $\hat{X}^{(j,h,\hat{n})}$ that is not detected as an outlier, we can form the set of potential shifts, denoted by $\hat{S}^{(h,j)}$, for job j and day h . This set is defined as $\hat{S}^{(h,j)} = \bigcup_{X \in \mathcal{K}^{(j,h,\hat{n})}} Y(X)$ and collects the shifts used in the sub-solutions associated with the similar data points. As observed from preliminary experiments, the restriction of potential shifts to neighbors' historic shifts for an outlier data point can substantially degrade solution quality because its demand curve is too different from previous ones. Therefore, we opt to consider all possible shifts for outlier data points at the expense of more computational effort.

To solve a new personnel scheduling problem over week $\hat{n}$, we propose the following optimization heuristic which is referred to as MILP- k NN:

1. For each pair of job $j \in J$ and day $h \in H$ that is not detected as an outlier, compute $\hat{S}^{(h,j)}$ using the k NN algorithm;
2. Build the MILP model (5.1)–(5.11) incorporating all possible shifts in $S^j \cap S^h$ for outlier data points in $\hat{\mathcal{X}}$, and only the shifts in $\hat{S}^{(h,j)} \cap S^j \cap S^h$ for the other data points $\hat{X}^{(j,h,\hat{n})} \in \hat{\mathcal{X}}$ (this restriction does not apply to the anonymous shifts);

3. Solve the resulting MILP model using a general-purpose MILP solver.

5.5 Experimental results

The proposed MILP- k NN heuristic is evaluated in different situations by means of computational experiments. It was programmed in Python 3 with libraries from scikit-learn [103] for data scaling and k NN. Similarly, the code for loading the input data, running the k NN, as well as creating the MILP models was also written in Python 3. All MILP problems were solved using the IBM ILOG CPLEX 22.1.1.0 MILP solver on a Linux machine powered by an 8-core (2 threads per core) Intel(R) Xeon(R) CPU E5-2637 v2 clocked at 3.5 GHz.

In this section, we start by describing how we generated a dataset for our experiments (Section 5.5.1) before explaining how we can tune the value of k for MILP- k NN (Section 5.5.2). Then, we report in Section 5.5.3 the results obtained by MILP- k NN.

5.5.1 Data generation

Our similarity analysis is based on the principle that the industry, including our partner, maintains and stores historical data such as input data (e.g., demand, employees' data) and outputs (e.g., schedules from existing solvers and shifts). However, we only had access to a few real-life instances with almost no historical data for testing our methodology. The size of these instances was also limited (less than 8 jobs and 90 employees). In this context, we have decided to create 8 datasets inspired by the demand curves of these real-life scenarios. For each dataset, we generate a large number of instances with the same number of jobs and employees as described in Table 5.4. For each dataset, the instances differ by the demand curves of the jobs but the skills and availability of the employees remain the same to mimic a stability of the operations in a retail store.

Table 5.4 Number of jobs and employees in the datasets

Dataset	1	2	3	4	5	6	7	8
Jobs	5	6	7	8	8	9	10	12
Employees	85	95	128	95	119	133	165	190

Let us summarize the process of creating the instances for one dataset. In order to construct the demand curves, we consider the following general assumptions:

- Each job's working hours are distributed continuously throughout the day.

- Each demand curve exhibits a bell-shaped pattern similar to Figure 5.1.
- The number of customers visiting the store increases on weekends, leading to a noticeable rise in the demand curve, especially during peak hours.
- Some jobs require a constant or minimally fluctuating number of employees throughout the working hours.
- The demand of each job at each period follows a normal distribution.

The generation process of an instance proceeds as follows. For each job, we randomly select from the real-life scenarios a job curve, which provides for each time period the expected demand of a random Poisson variable. Next, we draw from these random variables the demand to consider at the corresponding period for the corresponding job. Once the demand curves are established, we proceed to add employees recursively to ensure that an adequate number of qualified employees are available in accordance with the demand for each job during each time period. The contract types of employees are randomly assigned as either full-time or part-time. The availability of part-time employees is determined randomly, with their working days selected from a uniform distribution ranging from 2 to 4 days.

Once we have one initial instance for each dataset, we run a Monte Carlo simulation, as proposed by the study in [104], to generate 55 additional historical instances for each dataset. This simulation only perturbs the demand curves using a coefficient of variation of 10%. Out of the 56 historical instances generated for each dataset, 6 are also used for validation and parameter tuning in Section 5.5.2.

In addition to generating input data instances, we obtain solution schedules for each instance using a MILP solver. For each generated instance, we construct the corresponding MILP model (5.1)–(5.11) incorporating all shift variables $z_{s,e,j}$ (called the z -variables hereafter), and solve it using a general-purpose MILP solver. As detailed in Section 5.4.1, we decompose each historical instance and its solution per job and day to define data points $X \in \mathcal{X}$ and their sets of selected shifts $Y(X)$. Finally, we further generate 10 additional test instances for evaluation purposes. For these instances, we also record their data points $\hat{X} \in \hat{\mathcal{X}}$. Although solving such large datasets can be time-consuming, especially for cases with many jobs and employees, it is important to note that in the worst-case scenario, we need to solve these instances only once. For future instances, we will use our proposed method, which aims to reduce both the problem size and the solution time.

5.5.2 Tuning parameter k

The k parameter in k NN represents the number of nearest neighbors to consider for each data point. Its value has a substantial impact on the speedup that we can obtain and on the solution quality. Indeed, a larger k entails the inclusion of more historical shift variables ($\hat{S}^{(h,j)} = \cup_{X \in \mathcal{K}(j,h,\hat{n})} Y(X)$) and is anticipated to yield more robust solutions at the cost of larger computational times. To illustrate this, we performed the following experiments on the 6 validation instances used for each dataset. First, we solved these instances with an Exact algorithm that solves the MILP model (5.1)–(5.11), incorporating all possible shifts and a relative optimality gap tolerance of 0.01%. Then, we solve these instances once again with the MILP- k NN heuristic (considering only the other 50 historical instances) and different values of k ranging from 1 (representing the nearest neighbor) to $k_{max} = |J| \times 7 \times 50$ (encompassing all shifts selected in the historical instances if there are no outliers). For each validation instance and value of k , we record the problem size reduction, the computational time, and the *error* which is the deviation between the solution cost and that obtained by the Exact method.

We illustrate our findings using the first two datasets as examples to avoid being exhaustive. Similar trends are observed across the remaining datasets. Figure 5.3 shows the results obtained for Dataset 1 on the left and Dataset 2 on the right. In all sub-figures, the x -axis indicates the value of k , ranging from 1 to k_{max} . Sub-figures 5.3a and 5.3b demonstrate the impact of varying the value of k on the problem size, with the percentage of removed z -variables reported on the y -axis. Smaller values of k yield fewer decision variables, being the percentage of removed variables decreased as the value of k increases, as anticipated. In simple terms, as k increases, the problem size is reduced less because more z -variables are included. Conversely, with smaller k values, the problem is more effectively reduced due to the inclusion of fewer z -variables.

Sub-figures 5.3c and 5.3d illustrate the impact of varying the value of k in the MILP- k NN algorithm on the computational time. The y -axis depicts the corresponding time in seconds, with each point providing the average computational time over the validation instances. As k increases, the plots show how this time is influenced, offering insights into the impact of parameter k . Note that the average computational times of the Exact algorithm for the two instance sets are 4276 and 5470 seconds, respectively.

Finally, Sub-figures 5.3e and 5.3f indicate the quality of the solutions produced by MILP- k NN by displaying the error in percentage on the y -axis. Notably, when $k = 1$, a large error occurs; however, increasing the value of k results in a decreased error. In fact, it seems that, for each dataset, a small error can be achieved with a relatively small value of k that yields

a substantial speedup. Similar results have been observed for the other datasets.

Assuming that each dataset corresponds to a different retail store with different variability from one week to another, we propose to adjust the value of k for each dataset using their validation instances. To do so, we apply a dichotomic search method on each validation instance that we denote with index v . This method works as follows. Let $e_v(k')$ be the error obtained by MILP- k NN for instance v when $k = k'$ and let τ be an acceptable error ($\tau = 1\%$ for our tests). As we observed in our tests, we assume that $e_v(1) > \tau$ and $e_v(k_{max}^v) \leq \tau$, where $k_{max}^v = |J| \times 7 \times 50$ is the maximal k value to consider for instance v . The algorithm starts by setting the search interval $[\underline{k}, \bar{k}] = [1, k_{max}^v]$ and iteratively halves it until $\bar{k} = \underline{k} + 1$, ensuring that $e_v(\underline{k}) > \tau$ and $e_v(\bar{k}) \leq \tau$ throughout the search. Each iteration proceeds as follows:

1. Set $k = \lceil ac(\bar{k} - \underline{k})2 \rceil$.
2. If $e_v(k) > \tau$, then set $\underline{k} = k$; otherwise set $\bar{k} = k$.

When $\bar{k} = \underline{k} + 1$, the algorithm stops and we define the optimal value of k for instance v as $k_v^* = \bar{k}$. After all optimal values k_v^* have been computed for the validation instances of a dataset, we set $k = \max_v k_v^*$ when applying MILP- k NN for the test instances associated with this dataset.

Figure 5.4 shows the search space in the interval $(1, 166]$ for one validation instance of Dataset 2 with 6 jobs and 95 employees. In this interval, we explored 8 values of k until we found the smallest value of k producing an error of at most 1%. In this case, $k_v^* = 25$ and MILP- k NN requires 1176 seconds to yield a solution with an error of 0.94%.

5.5.3 Performance of MILP- k NN heuristic

The performance of the proposed MILP- k NN heuristic is evaluated in this section on 80 test problems (10 per dataset) by comparing to the following three algorithms:

1. Exact: Solves the MILP model (5.1)–(5.11) considering all possible shifts, with an optimality gap tolerance of 0.01% and a time limit of 20,000 seconds;
2. Tol-5%: Similar to Exact, but with an optimality gap tolerance set to 5% and a time limit of 3,600 seconds;
3. MILP-UNet*: A heuristic proposed by the study in [96] that applies a pre-processing procedure to reduce the size of the MILP model (5.1)–(5.11).

If Exact and Tol-5% are unable to solve the problem to optimality within the specified time limits, the best integer solution found during the enumeration process is returned.

MILP-UNet* differs from MILP- k NN in several aspects. First, it predicts the likelihood that each period corresponds to a shift starting or ending time period and uses those predictions to reduce the set of shifts to consider in the model. Second, it relies on a deep-learning methodology called a U-shaped encoder-decoder convolutional neural network to make those predictions. Third, in addition to job demands, it considers as features the employee information and the optimal solution of the linear relaxation of model (5.1)–(5.11). In the MILP-UNet* heuristic, a parameter α defines a threshold on the precision and recall metrics of the predictor on the validation instances. If one of those metrics for a given time period is less than α , the predictor is deemed unreliable for this period which is then considered as a valid starting and ending period. Therefore, a lower α value stands for a more aggressive decision-making and leads to more variable removals.

To construct a robust training dataset for MILP-UNet*, we utilized all 8 historical datasets, comprising 448 instances decomposed by jobs and days. The deep neural network of MILP-UNet* is trained once, a process taking approximately 30 minutes, and is then used to make predictions for all test instances. Unlike MILP-UNet*, the datasets are not combined in MILP- k NN and the algorithm is executed independently for each dataset. Since the training for MILP-UNet* is performed offline and the times required to make predictions in MILP-UNet* and to run k NN in MILP- k NN are negligible, we compare these algorithms based only on the computational time of the MILP solver. For both MILP-UNet* and MILP- k NN, the relative gap tolerance of the MILP solver is set to 1%. A time limit of 20,000 seconds is also imposed for the Exact, MILP-UNet*, and MILP- k NN algorithms.

The results of our experiments are summarized in Tables 5.5 to 5.7 reflecting the average over 10 test problem instances for each dataset. Tables A.1 to A.8 in Appendix A report detailed results.

Table 5.5 reports statistics on the size of the MILP model (5.1)–(5.11) solved by the four algorithms (Exact, Tol-5%, MILP-UNet*, and MILP- k NN). For the Exact and Tol-5% algorithms which solve the MILP model with all variables, this table reports the total numbers of constraints, of z -variables, and of the other variables for each dataset. For MILP- k NN and MILP-UNet*, the table indicates in the column *z -Removed* the percentage of the z -variables that were removed from the full model by the proposed pre-processing as well as the value of the corresponding parameter k or α used for each dataset. Recall that these parameter values were adjusted per dataset using the validation instances (see Section 5.5.2 for parameter k).

From the results in Table 5.5, we make the following observations. As expected, the problem

size increases with the number of jobs and employees (i.e., with the dataset id). Notably, the number of z -variables constitutes a very large proportion of all variables (over 90%) which emphasizes the key characteristic of our approach that targets the z -variables for problem size reduction. MILP- k NN significantly reduces the number of z -variables in each dataset, with removal percentages ranging from 77.7% to 96.8%. A larger percentage of removed variables indicates a more effective reduction, making the problem faster to solve. This highlights the effectiveness of the MILP- k NN method in simplifying the MILP model. While MILP-UNet* generally removes fewer z -variables than MILP- k NN, it still achieves substantial reductions with removal percentages ranging from 66.5% to 89.3%. On average, MILP- k NN and MILP-UNet* remove 85.5% and 78.6% of z -variables, respectively, showing a consistent and significant reduction across these datasets.

Table 5.6 focuses on the solution quality obtained by MILP- k NN, MILP-UNet*, and Tol-5% on the test instances. For each method and dataset, Table 5.6 presents the average error, measured as the relative average difference between the costs of the solutions obtained by the method and those provided by the Exact algorithm, over 10 test instances, as well as its standard deviation, and its minimum and maximum values (all expressed in percentage). The best results are highlighted in bold. Note that a negative error means that the heuristic found a better solution than the Exact algorithm for at least one test instance. This is possible when the Exact algorithm reached the time limit and did not find an optimal solution. We also remark that Tol-5% could not find any integer solution for the instances in Datasets 7 and 8 within the given time limit. On average, MILP-UNet* yields a lower error (0.7%) compared to MILP- k NN (1.2%). This indicates that MILP-UNet* performs better on average across the datasets, but we observe that most of the difference is due to the very good performance of MILP-UNet* on the two largest Datasets 7 and 8. For these datasets, the pre-processing of MILP-UNet* removed on average between 6.5% and 7.5% less variables than that of MILP- k NN, minimizing the risk of using bad predictions. On the other hand, MILP- k NN exhibits less variability in the solution quality with a lower average standard deviation (0.3%) compared to MILP-UNet* (0.7%). This is reflected, in general, with larger average minimum errors and smaller average maximum errors. In summary, while both MILP- k NN and MILP-UNet* show competitive performance, MILP- k NN demonstrates more consistent results, particularly in terms of average standard deviation and maximum errors across different datasets. The choice between the two methods may depend on other criteria such as the average computational time, or the available offline training time, which is much larger for the deep learning approach.

Table 5.7 is similar to Table 5.6 but concerns the computational time. For each heuristic, it gives the average, standard deviation, minimum and maximum computational time in sec-

Table 5.5 MILP model size

Dataset	Exact / Tol-5%			MILP- k NN		MILP-UNet*	
	Constraints	z -Variables	Other Variables	k	z -Removed (%)	α (%)	z -Removed (%)
1	8,420	1,127,934	65,426	57	77.7	85	72.3
2	9,964	1,253,580	79,841	28	82.8	85	66.5
3	11,964	1,747,850	92,345	70	82.6	85	84.6
4	12,652	1,289,998	103,476	22	80.8	80	71.6
5	13,132	1,669,661	105,010	59	86.7	80	78.0
6	14,756	1,924,712	117,999	27	79.1	70	78.3
7	16,740	2,323,550	131,375	5	94.8	60	88.3
8	19,928	2,622,678	157,546	3	96.8	60	89.3
Avg.					85.5		78.6

Table 5.6 Error (in %)

Dataset	Average			Standard deviation			Minimum			Maximum		
	MILP- k NN	MILP-UNet*	Tol-5%	MILP- k NN	MILP-UNet*	Tol-5%	MILP- k NN	MILP-UNet*	Tol-5%	MILP- k NN	MILP-UNet*	Tol-5%
1	1.3	2.5	2.0	0.2	1.2	0.8	0.9	0.6	0.5	1.6	4.6	3.4
2	1.3	1.2	1.0	0.2	0.7	0.6	0.8	0.5	0.2	1.8	4.5	2.6
3	1.2	1.7	32.2	0.1	1.4	29.7	-0.1	-0.2	0.2	1.2	8.6	64.0
4	1.1	1.6	2.0	0.3	0.4	1.3	0.4	0.8	0.0	1.9	2.4	4.2
5	1.3	1.0	2.1	0.2	0.4	0.8	0.9	0.3	0.9	1.5	2.4	4.7
6	1.2	1.9	9.2	0.2	0.4	13.0	0.9	1.2	0.3	1.7	3.0	74.4
7	0.9	-1.5	*	0.4	0.6	*	-0.2	-2.7	*	1.6	-0.5	*
8	1.4	-3.0	*	0.5	0.4	*	0.4	-3.6	*	2.8	-2.2	*
Avg.	1.2	0.7	*	0.3	0.7	*						

onds for each dataset. It also provides in column *Exact* the average time required by the Exact algorithm that serves as a reference for comparison. From these results, we observe that MILP- k NN and MILP-UNet* exhibit significantly lower average computational times compared to the Exact algorithm and the Tol-5% algorithms, demonstrating the efficiency of these two heuristics. On average, MILP- k NN has a lower computational time (620 seconds) compared to MILP-UNet* (967 seconds), which is due to a larger average number of z -variables removed (see Table 5.5). This suggests that MILP- k NN tends to be more computationally efficient on average across the datasets. Furthermore, MILP- k NN exhibits a lower average standard deviation (155 seconds) compared to MILP-UNet* (449 seconds), indicating more consistency. Given that the average computation time is also less for MILP- k NN, this heuristic entails, in general, smaller minimum and maximum computational times. In summary, this table highlights the efficiency of MILP- k NN and MILP-UNet* in solving test instances within a reasonable computational time. MILP- k NN, on average, demonstrates a more efficient and stable performance compared to MILP-UNet* which requires a large training dataset for effective learning, i.e., for training a deep neural network. This complexity and the lack of adequate training dataset may contribute to potential challenges when generalizing to diverse datasets as observed in the MILP-UNet*'s maximum error of 8.6% for Dataset 3 (see Table 5.6).

Overall, MILP- k NN consistently outperforms MILP-UNet* in terms of computational effi-

ciency. Although MILP-UNet* generally yields better average solution quality, as indicated by lower average and minimum errors across datasets, MILP- k NN shows more stability with lower standard deviations and maximum errors. Thus, the choice between MILP- k NN and MILP-UNet* involves a trade-off between model complexity and performance. MILP-UNet* may compute better solutions but at the cost of increased computational complexity and the need for a substantial training dataset.

5.5.4 Robustness analysis of MILP- k NN heuristic

This section presents a robustness analysis in order to assess the impact of the data generation process in the results obtained by the proposed MILP- k NN heuristic. This method relies on the identification of historical job demand patterns and previously assigned schedules, under the assumption that these patterns are recurrent. To simulate this, we introduced a 10% coefficient of variation in the demand data generation process (see Section 5.5.1). To evaluate its robustness, we generated additional test instances with varying distributions from the historical datasets by applying larger coefficients of variation in the demand data generation process. Specifically, for each dataset, we created two new sets of test instances: one with a 20% coefficient of variation and another with a 30% coefficient of variation. The corresponding sets of qualified employees and their available times were generated following the methodology described in Section 5.5.1, but tailored to the newly generated demand data. We assessed the performance of MILP- k NN on these test instances by comparing it to the Exact method. Tables 5.8 and 5.9 summarize the results obtained.

On average, MILP- k NN removed 46.7% and 22.8% of the z -variables for instances generated with 20% and 30% coefficients of variation, respectively. Notably, the amount of removed z -variables decreases by increasing the variability of test instances compared to the training instances. This is caused by the outlier detection phase of our heuristic. As the job demand curves are more perturbed in the test instances, more outlier data points are detected leading to the inclusion of all their possible shifts in the MILP- k NN.

Yet, for the 20% coefficient of variation, MILP- k NN demonstrates a substantial reduction in computational time, averaging 1,814 seconds, which is roughly 69% faster than the Exact method. This efficiency is achieved while maintaining a low average error rate of 1.6%. For the 30% coefficient of variation, despite the increased variability, MILP- k NN continues to outperform the Exact method in terms of computational time, with an average of 1,936 seconds, which is about 44% faster. The method also maintains a low average error rate of 1.0%, indicating robustness in solution quality even with more challenging datasets.

These results demonstrate that MILP- k NN is sensitive to the historical database. Indeed,

Table 5.7 Computational time (in seconds)

Dataset	Exact	Average			Standard deviation			Minimum			Maximum		
		MILP- <i>k</i> NN	MILP-UNet*	Tol-5%	MILP- <i>k</i> NN	MILP-UNet*	Tol-5%	MILP- <i>k</i> NN	MILP-UNet*	Tol-5%	MILP- <i>k</i> NN	MILP-UNet*	Tol-5%
1	1,867	214	272	564	39	113	132	141	171	294	319	817	795
2	4,197	267	498	1,132	48	171	137	150	230	761	408	941	1,385
3	17,641	1,221	865	3,233	7	414	444	1,221	367	1,789	1,234	1,982	3,600
4	1,858	228	238	882	50	50	209	143	162	590	319	380	1,698
5	4,429	133	342	1,127	25	57	196	90	105	878	198	333	1,438
6	6,675	473	342	2,815	99	58	598	327	150	1,752	709	478	3,600
7	20,000	922	2,830	3,600	376	1,472	3,600	352	1,070	3,600	2,038	6,521	3,600
8	20,000	1,503	2,347	3,600	598	1,258	3,600	303	654	3,600	2,660	6,743	3,600
Avg.	9,584	620	967	2,119	155	449	1,114						

the heuristic is highly effective if the job demand patterns are recurrent and similar between them. As the data variability increases, the method’s pruning capability decreases, but it still manages to maintain low error rates and computational efficiency under mild data variations.

Table 5.8 Experiments with a 20% coefficient of variation

Dataset	Jobs	Employees	Exact				MILP- <i>k</i> NN		
			Constraints	<i>z</i> -Variables	Other Variables	Time (seconds)	Time (seconds)	<i>z</i> -Removed (%)	Error (in %)
1	5	91	8,540	1,384,129	65,843	1,709	382	33.3	0.8
2	6	129	10,644	2,102,308	80,578	2,610	1007	45.2	1.4
3	7	155	12,508	2,419,245	92,991	5,708	345	71.2	1.1
4	8	111	12,972	1,713,670	103,671	821	1,193	55.5	2.4
5	8	168	14,112	2,635,480	105,942	7,486	338	54.3	1.5
6	9	142	14,936	2,050,846	118,142	1,005	988	55.1	2.1
7	10	199	17,420	3,098,985	132,112	14,245	3,528	32.5	2.1
8	12	252	21,168	3,963,331	159,009	13,612	6,734	26.2	1.2
Average						5,899	1,814	46.7	1.6

Table 5.9 Experiments with a 30% coefficient of variation

Dataset	Jobs	Employees	Exact				MILP- <i>k</i> NN		
			Constraints	<i>z</i> -Variables	Other Variables	Time (seconds)	Time (seconds)	<i>z</i> -Removed (%)	Error (in %)
1	5	99	8,700	1,405,176	65,867	573	472	22.5	1.3
2	6	137	10,804	2,033,140	80,682	1,705	565	41.3	1.1
3	7	192	13,248	2,992,273	93,458	2,616	1,475	16.4	0.6
4	8	122	13,192	1,846,482	103,800	587	378	41.5	1.1
5	8	185	14,452	3,062,643	106,121	6,793	1,681	32.2	1.5
6	9	152	15,136	2,339,417	118,233	760	852	19.2	0.4
7	10	227	17,980	3,526,118	132,415	6,414	2,613	3.7	0.6
8	12	270	21,528	4,169,816	159,225	8,143	7,455	5.6	1.0
Average						3,445	1,936	22.8	1.0

5.6 Conclusion

Our research endeavors to address the complex challenges associated with personnel scheduling by employing a novel methodology that integrates historical data analysis, similarity measures, and mixed integer linear programming. The in-depth analysis of historical data patterns and the identification of recurrent trends play a pivotal role in our approach. By leveraging the *k*NN algorithm, we dynamically adapt to historical instances, ensuring a focused search space for potential solutions. Our optimization heuristic, MILP-*k*NN, efficiently utilizes historical shifts to generate near-optimal personnel schedules.

The experimental results showcase the adaptability and effectiveness of the MILP- k NN heuristic across various datasets. By systematically exploring the impact of the k NN parameter k , we strike a balance between solution quality and computational efficiency. The trade-off analysis demonstrates that, in many instances, a relatively small k can yield solutions with negligible error while significantly reducing solution times. The simplicity of the k NN approach contributes to the efficiency and stability of MILP- k NN which relies on a straightforward mechanism for finding similar instances and does not involve complex training processes.

In the quest for near-optimal personnel schedules, the proposed methodology offers a promising avenue for organizations seeking efficient and tailored solutions. As personnel scheduling remains a critical aspect of workforce management, our approach aligns with the evolving landscape of optimization techniques and data-driven decision-making. Nevertheless, as future research avenues, we see two opportunities. First, instead of using a fixed value of the parameter k in the k NN algorithm, it might be possible to adjust this value for each datapoint depending on the computed average distance of each point. Second, the proposed method could be generalized to the case where employees work on multi-job shifts, i.e., when they can switch from one job to another in the middle of a shift.

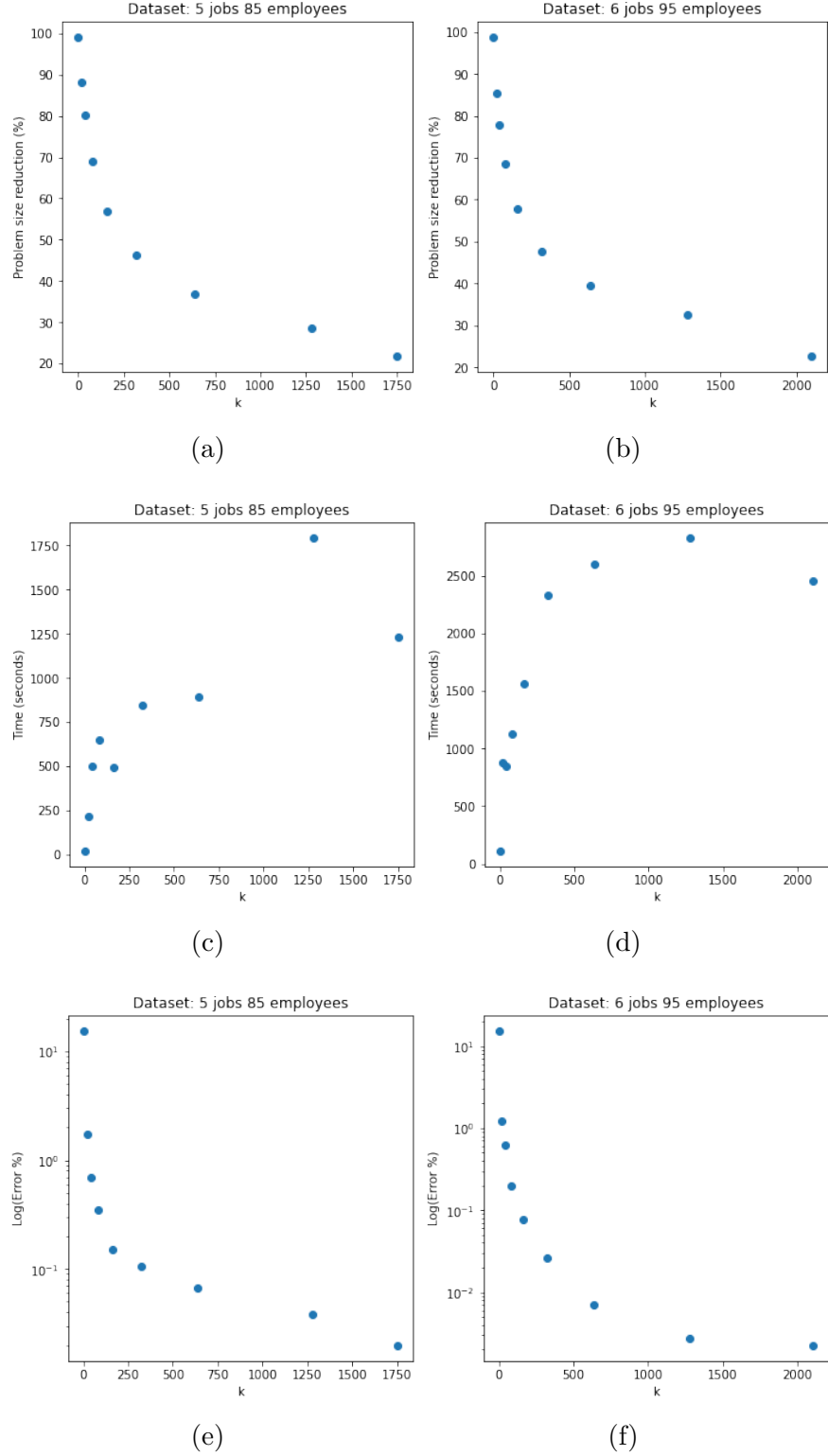


Figure 5.3 Impact of k on problem size, computational time, and solution quality for the instances in Dataset 1 (in the left) and in Dataset 2 (in the right).

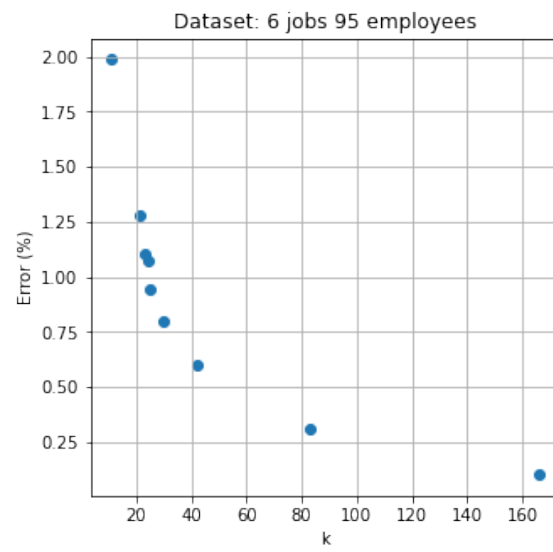


Figure 5.4 Search for the value of k for one validation instance in Dataset 2

CHAPTER 6 ARTICLE 3: A DATA-DRIVEN METAHEURISTIC FOR PERSONNEL SCHEDULING IN THE RETAIL SECTOR CONSIDERING MULTI-JOB SHIFTS

F. Rastgar-Amini, C. Contardo, G. Desaulniers, “Flexible Personnel Scheduling Problems in the Retail Sector Considering Multi-Job Shifts”, This paper was submitted on March 16, 2026, *INFOR: Information Systems and Operational Research*, 2026

6.1 Introduction

Personnel scheduling in the retail sector is a complex optimization challenge, driven by the need to align employee work schedules with highly variable customer demand while minimizing operational costs. The dynamic nature of job demands, coupled with the requirement for workforce flexibility, necessitates innovative scheduling approaches that go beyond traditional methods.

Personnel scheduling problems (PSPs) arise in a wide range of application areas, including healthcare, transportation, manufacturing, call centers, and retail, each with distinct operational characteristics and regulatory requirements. In healthcare, for example, continuous coverage and strict skill-mix regulations dominate scheduling decisions, while in transportation, routing and duty-time constraints play a central role. Manufacturing environments often emphasize production continuity and machine utilization, whereas call centers must respond to rapidly fluctuating call volumes with fine-grained staffing adjustments. These differences lead to diverse problem structures, constraint sets, and solution approaches across sectors. In this paper, the focus is placed on retail environments, where demand varies significantly over short time intervals, and skilled employees are often required to perform multiple types of work within flexible shifts, making scheduling particularly challenging.

In retail environments, PSPs are typically characterized by several common dimensions, including the definition of feasible shifts, the set of jobs or activities to be performed, the set of available employees, and the resulting schedules that assign employees to shifts and jobs over the planning horizon (typically, one week). An employee schedule is composed of a sequence of shifts and days off, where the days off are determined when solving the problem. Shifts specify when employees start and end work and may include breaks. A shift is said to be mono-job if the employee performs a single job throughout the entire shift, whereas it is deemed multi-job if it is composed of multiple segments during which the

employee performs different jobs. Jobs define the types of work that must be covered, such as cashiering or stocking. Employees differ in their availability, skill eligibility, and contractual conditions, all of which must be respected in the scheduling process. The primary objective is usually to meet time-varying staffing requirements at minimum cost, accounting for wages, overtime, and penalties associated with undercoverage or overcoverage. To ensure feasibility and quality, the selected schedules must satisfy a wide range of constraints, including labor regulations, rest requirements, skill compatibility, and workload limits.

The motivation for this research arises from the pressing need to improve operational efficiency and cost-effectiveness in retail personnel scheduling. In retail settings, customer demand fluctuates significantly throughout the day and week, often requiring distinct job roles—such as cashiers, sales associates, or stock clerks—at different times. The demand for a job is, typically, expressed as a demand curve that indicates the number of employees required in every (e.g., 15-minute) period of the planning horizon. In order to closely match fluctuating customer demand and reduce the costs of both understaffing and overstaffing, retail scheduling often allows highly flexible shift start times and durations. However, organizational rules such as minimum and maximum shift durations, rest requirements, and contractual constraints typically limit full flexibility. As a result, traditional mono-job shift scheduling may lead to overstaffing during low-demand periods or understaffing during peak times, which in turn increases labor costs or degrades service quality. By allowing employees to switch between jobs within a single shift, multi-job shifts enable more adaptive scheduling and a closer alignment between workforce allocation and demand patterns. This approach can reduce the total number of employees required, limit overcoverage penalties, and improve employee utilization, ultimately leading to significant cost savings and improved customer satisfaction.

Initially introduced by [5], mathematical programming models and more specifically mixed-integer linear programming (MILP) formulations constitute one of the most widely adopted approaches for solving PSPs, both in the academic literature and in industrial practice. Dantzig formulated an employee-to-toll-booth assignment problem using a set-covering model. In this formulation, all feasible working patterns are enumerated explicitly for all possible start times, end times, and breaks. Then, one integer variable is associated with each feasible work pattern. This explicit representation has made set-covering formulations especially appealing in both industrial and academic contexts, since they offer enhanced modeling flexibility [e.g., 22, 23, 25–27, 37, 38, 85]. In practice, personnel scheduling must account for a wide variety of organizational rules that reflect the priorities of different companies, such as minimum and maximum working hours per shift, limits on the number of shifts per week, employee preferences, job priorities, and rest requirements. From the perspective of software

providers that deliver scheduling solutions to multiple clients, it is often preferable to rely on explicit set-covering formulations, in which shifts are defined explicitly and assigned to employees and jobs. While this modeling approach offers flexibility and ease of customization across diverse operational contexts, it requires the enumeration of all feasible shifts, which can lead to a very large number of decision variables and significant computational challenges. To avoid complete shift enumeration, several works [e.g., 37, 38, 44] have proposed column-generation-based algorithms that generate only a subset of the shifts by solving pricing subproblems. These sophisticated techniques may be difficult to implement in practice when complex shift feasibility rules and costs must be taken into account in practice.

In contrast to explicit methods, implicit formulations are frequently employed to eliminate the need for exhaustive enumeration of shifts by imposing rules directly within the constraints [e.g., 28, 29, 33, 34]. However, they are difficult to deploy in practice when a wide range of heterogeneous and sometimes contradictory organizational rules must be respected. In many real-world settings, employment costs, contractual conditions, skill restrictions, and company-specific policies are complex and difficult to represent accurately within purely implicit models [3]. In multi-job settings, a shift is not simply a time interval but a structured sequence of job assignments over time. Such complexity makes it challenging—and in many cases impractical—to represent feasible schedules implicitly. As argued in [4], implicit formulations struggle to capture the combinatorial structure of multi-job shifts, particularly when job switching, activity-level demand, and fine-grained time discretization are required. Consequently, explicit shift enumeration remains the appropriate modeling approach in this study.

The computational challenges induced by large-scale personnel scheduling models have motivated the development of a wide range of solution approaches in both the research literature and practice. These include exact optimization techniques (e.g., branch-and-price), problem-specific heuristics, metaheuristics, and, more recently, artificial intelligence-based methods. Heuristic and metaheuristic approaches aim to construct high-quality solutions within reasonable computing times by exploiting problem structure or guided search strategies, while data-driven and learning-based methods leverage historical information to support or enhance decision-making. Each of these tools contributes differently to improving tractability, solution quality, or robustness in complex scheduling environments.

This research project was conducted in collaboration with our industrial partner UKG that markets an optimization software for solving PSPs. Given that this software is used by a large number of companies with a wide range of operational rules that are not always easy to model, UKG relies on a set-covering formulation, where feasible shifts are explicitly

enumerated. In this context, a central difficulty lies in the sheer size of the resulting models if shift enumeration is not limited. Consequently, considerable attention has been devoted to approaches that focus on reducing the effective problem size, either by limiting the number of shifts considered, guiding the selection of promising candidates, or approximating parts of the decision process. Such strategies aim to preserve the modeling flexibility and practical appeal of set-covering formulations while alleviating their computational burden. This perspective motivates the methods developed in this work, which concentrate on problem size reduction as a key lever for improving the efficiency of large-scale personnel scheduling models.

Within this context, this study builds on the research of [96], which addressed the challenge of problem size reduction in set-covering formulations through data-driven shift enumeration for mono-job settings. That work demonstrated that learning-based methods can effectively identify high-quality shift candidates and substantially reduce the number of variables required in large-scale models. The current research extends this line of work by introducing multi-job shifts, allowing employees to perform different jobs within a single shift while preserving the core methodological principles developed for the mono-job case. By leveraging and adapting the mono-job learning frameworks, this research broadens their applicability to more realistic retail environments and addresses the additional complexity induced by job switching within shifts. Note also that the learning technique could also be exploited to reduce the size of the pricing subproblems in a column generation/branch-and-price algorithm.

This paper is organized as follows: In Section 6.2, we review the relevant literature. Section 6.3 formally defines the PSP under study and presents the MILP model that we propose for this problem. Section 6.4 details the methodology, including the ML models and heuristic pipeline. Experimental results are discussed in Section 6.5. Finally, Section 6.6 provides conclusions and future research directions.

6.2 Literature Review

PSPs were introduced by Edie [1] to decrease the vehicles' waiting time for service at toll booths with the minimum number of toll collectors. These problems have been studied extensively since then and have evolved alongside the increasing complexity of workforce management in sectors such as healthcare, transportation, manufacturing, call centers, and retail. PSPs comprise a broad class of optimization problems concerned with assigning employees to work periods, shifts, and jobs over a planning horizon while satisfying operational demand, labor regulations, and organizational objectives. This paper studies a PSP in the retail industry that involves multi-job shifts. Therefore, we omit from our review most works that consider only mono-job shifts like that of Hassani et al. [84]. For extensive surveys on

PSPs, the reader is referred to [13], [11], and [2].

Different PSP variants involving multi-job shifts have been studied in the literature. We classify them into three categories: anonymous multi-job shift scheduling problem (AMJSSP) that aims at defining anonymous shifts over one or several days; personalized multi-job shift scheduling problem (PMJSSP) that builds personalized shifts over a single day; and multi-job tour scheduling problem (MJTSP) that computes personalized schedules for a whole week. Below, we summarize the main works for each category.

In the AMJSSP, mono-job and multi-job shifts respecting feasibility rules are built to cover at minimum cost the demand of multiple jobs (often called activities) over a day or multiple days. They are not assigned to specific employees. Demasse et al. [105] were among the first to study this problem over a single day. They devised a column generation algorithm where the shift generation subproblem is modeled as an automaton and solved by constraint programming. Quimper and Rousseau [4] proposed two ways to model complex shift construction rules, namely, an expanded automaton and context-free grammar, and derived specialized graph structures that are exploited in an efficient large neighborhood search heuristic. Côté et al. [36] proposed an implicit MILP model that is based on a context-free grammar graph to encode shift feasibility rules. Restrepo et al. [106] developed a column generation heuristic where the pricing subproblems are modeled as shortest path problem with resource constraints and solved by a pulse algorithm. They tested their method on instances defined over a single day or over multiple days when the demand is continuous and shifts can overlap over two days. In the latter case, the shifts are not linked to form employee schedules. To overcome this drawback, Restrepo, et al. [38] proposed a Benders decomposition algorithm for a AMJSSP defined over a one-week horizon. The master problem is solved by column generation to create anonymous employee schedules where the shifts are not filled with jobs. The Benders' subproblems fill those shifts with jobs and are formulated as a MILP model based on a context-free grammar graph like in [36]. This method succeeded to solve to optimality 7-day AMJSSP instances with up to 10 jobs and 42 employees in up to 4,112 seconds on average. It could also solve single-day instances with up to 30 jobs in about the same time. Recently, Römer [15] introduced novel MILP formulations using block-based state-expanded networks to encode shift feasibility rules, with techniques like aggregate side constraints and decoupling of shift composition from job assignments to significantly reduce model size and symmetry. Through computational experiments on two literature benchmark sets involving up to 10 jobs, the approach solves all single-day instances—including over 70 previously open ones—to optimality, with most requiring less than 30 minutes.

Some works have tackled the PMJSSP which aims at computing personalized shifts for a sin-

gle day, considering heterogeneous employees with different skills and availability. Côté et al. [44] presented an exact branch-and-price algorithm where the pricing subproblems generate schedules, are modeled using context-free grammars, and solved by dynamic programming. This algorithm was, however, only tested on the AMJSSP over a single day and on an activity assignment problem over multiple days where the shifts for each employee are pre-fixed. Boyer et al. [37] extended this algorithm for the PMJSSP with interruptible jobs (like in the previous works) and uninterruptible tasks with fixed durations that are subject to precedence constraints. More recently, Hernández-Leandro et al. [18] proposed a matheuristic for the PMJSSP, where Lagrangian relaxation identifies promising shifts modeled via context-free grammars, and a set-covering model restricted to the shifts found throughout the Lagrangian relaxation iterations is solved to obtain a feasible solution. Tested on the same instances used by Côté et al. [44], this approach yielded high-quality results.

The MJTSP extends the PMJSSP to multiple days. It aims at computing one-week schedules for personalized employees (with different skills and availability), including the positioning of the days off. For a restricted problem version where days off are given as inputs, Dahmen and Rekik [17] proposed a MILP formulation involving shift variables and also developed a hybrid heuristic combining tabu search with branch-and-bound. For the MJTSP, Gérard et al. [107] developed a MILP model based on schedule variables and four solution methods, including two column generation heuristics and a greedy heuristic. The column generation pricing subproblems are solved by a nested dynamic programming heuristic (schedules are built from shifts, which are built from shift segments), which is also used in the greedy heuristic. Their computational results on real and synthetic instances with up to 5 jobs and 45 employees indicate that the column generation heuristic using a diving procedure to find an integer solution finds high-quality solutions if sufficient time is provided, while the greedy heuristic can provide solutions of varied quality in a few of seconds. Restrepo et al. [108] introduced two MILP formulations for the MJTSP: one with shift variables and another with schedule variables. For each model, they devised an exact branch-and-price algorithm and a column generation heuristic where columns are generated at the root node and the final restricted master problem is solved as a mixed-integer problem. For the shift-based model, the pricing subproblems are modeled using a context-free grammar graph and solved by the dynamic programming algorithm of Quimper and Rousseau [4]. For the schedule-based model, the subproblems are solved in two phases, where the first phase uses context-free grammars to generate shifts and the second phase applies a label-setting algorithm to find schedules from the generated shifts. Computational results on randomly generated instances with up to 5 jobs and 40 employees indicate that the shift-based column generation heuristic outperforms its schedule-based counterpart. At the opposite, exploiting the schedule-based model in

a branch-and-price algorithm yields better results than relying on the shift-based model. Recently, Shariat and Huang [109] conceived a large neighborhood search algorithm for the MJTSP which re-optimizes at each iteration the schedule of an employee using context-free grammars for generating new potential shifts for this employee and a shortest path problem with resource constraints for generating a new schedule. Experiments on real and randomly generated instances involving up to 19 jobs and 102 employees show that, except for the smallest instances, the proposed heuristic clearly outperforms solving a MILP formulation using the CPLEX MILP solver subject to a 1-hour time limit.

Recent advances for solving combinatorial optimization problems leverage ML and data mining to enhance MILP efficiency, focusing on variable reduction, warm-starting, and bound prediction rather than replacing MILP with pure heuristics [55]. For large-scale MJTSPs considering mono-job shifts only, Rastgar-Amini et al. [96] studied ML methods to accelerate the solution of a set-covering model with explicit shift enumeration. They proposed deep learning heuristics that predict promising shift start and end time points from training data to filter out non-promising shifts and reduce problem size. The reduced-sized model is then solved using a commercial solver. Computational results on real-life instances with up to 8 activities and 94 employees show that this approach significantly speeds up solution times with acceptable optimality gaps. Rastgar-Amini et al. [110] tackled the same problem with k-nearest neighbors data mining to identify similar historical instances and restrict shift enumeration to those matching past optimal schedules, thereby reducing problem complexity. Experiments on instances derived from real data, involving up to 12 jobs and 252 employees, achieve an average 85.5% reduction in decision variables, resulting in up to 15.5 times faster computing times with about 1.2% higher costs, compared to solving the problem without variable reduction.

The flexibility and strong performance of the ML-driven shift enumeration approach proposed in [96] make it particularly suitable for extension to more complex scheduling settings. These concepts motivate the present study to extend its mathematical framework to accommodate multi-job shifts in the MJTSP. To achieve this, we extend the learning-based prediction models to identify not only shift boundaries but also the start and end times of individual job segments within shifts. This extension preserves the scalability benefits of ML-driven shift enumeration while enabling its application to more realistic retail scheduling environments where job switching within shifts is essential.

6.3 A Multi-Job Tour Scheduling Problem

This section begins by providing a description of the MJTSP under study and presenting the notation that will be used in the remainder of the paper. It then introduces a MILP formulation of this problem.

6.3.1 Problem Description

The MJTSP with multi-job shifts that we consider is an extension of the mono-job shift variant studied by Rastgar-Amini et al. [96]. The problem involves assigning a set of skilled employees E to a set of potential shifts S over a one-week planning horizon divided into a set of days $H = \{1, 2, \dots, 7\}$ and a set of 15-minute time periods P . For each job $j \in J$ and period $p \in P$, demand d_p^j specifies the number of required employees. Each shift $s \in S$ is defined by a start period a_s , an end period b_s , and a sequence of segments, where each segment corresponds to a time interval and a job $j \in J$. A shift s is feasible if its duration $k_s = b_s - a_s + 1$ (in periods) respects a minimum (n_{min}^S) and a maximum duration (n_{max}^S) and if each of its segments also respects a minimum duration (n_{min}^K). A cost of c^M is associated with the transition between two segments of a shift. The assignment of an employee to a shift needs to satisfy two feasibility requirements: the employee is (1) available throughout the shift, and (2) qualified to perform all the jobs associated with that shift. Additionally, a feasible schedule (the sequence of shifts assigned to an employee e) must include a minimum number of days off (n_e^O) and a minimum rest duration between two consecutive shifts (n^R).

In the retail industry, the perfect match of demand and scheduled employees is difficult to achieve due to high demand fluctuations and respecting the minimum shift/segment duration requirements, leading to overcovering some periods with excessive employees. Also, the insufficient skills and availabilities of employees may cause the undercoverage of some jobs at some periods, which we handle by *anonymous* shifts at a higher cost to guarantee feasibility. An anonymous shift contains a single segment and is, thus, associated with a single job. It can be selected more than once and will be offered to one or several temporary employees (not in E) if selected.

The goal of our MJTSP is to minimize total costs, including employee remuneration, anonymous shift costs, overcoverage penalties, and transition cost while satisfying job demand d_p^j for each job $j \in J$ at each period $p \in P$. Overcoverage penalties and employee remuneration follow non-decreasing stepwise functions, consistent with practice [see 96]. For overcoverage, it allows to better spread it over different periods. For remuneration, it allows to better balance working time among the employees. Note that the remuneration costs are not real ones.

Otherwise, the more experienced employees which are usually paid more would be awarded less working hours.

Note that we assume that no breaks need to be assigned to the shifts. In fact, in the retail industry, breaks are often assigned during the operations when observing the real demand. To account for the addition of these breaks when solving the MJTSP, the demands are slightly increased at strategic time periods of the horizon.

6.3.2 Notation

The notation for sets, subsets, decision variables, and parameters is presented in Tables 6.1 to 6.3. It is tailored to accommodate multi-job shifts while maintaining consistency with the mono-job model.

This notation depends on the following assumptions.

1. A shift $s \in S$ cannot contain two segments associated with the same job, i.e., $g_s = |J^s|$;
2. The starting and ending periods of a shift or a segment are included in this shift or segment;
3. The segments in a shift $s \in S$ are non-overlapping, i.e.,

$$[\alpha_{s,j_1}, \beta_{s,j_1}] \cap [\alpha_{s,j_2}, \beta_{s,j_2}] = \emptyset, \quad \forall j_1, j_2 \in J^s, j_1 \neq j_2;$$

4. The segments in a shift $s \in S$ fully cover the shift, i.e.,

$$\bigcup_{j \in J^s} [\alpha_{s,j}, \beta_{s,j}] = [a_s, b_s].$$

Only the first of these assumptions is over-restrictive. The proposed approach can easily be adapted if it does not hold, but additional notation would be required.

Table 6.1 Sets and Subsets

Set	Description
H	Set of days in the planning horizon, $H = \{1, 2, \dots, 7\}$
P	Set of all 15-minute time periods in the planning horizon
E	Set of employees
H^e	Subset of available days for employee $e \in E$
J	Set of jobs
S	Set of all shifts
S^j	Subset of shifts that include job $j \in J$ in at least one segment
J^s	Subset of jobs included in shift $s \in S$
S_j^p	Subset of shifts $s \in S$ that include job $j \in J$ and cover period $p \in P$, i.e., $p \in [\alpha_{s,j}, \beta_{s,j}]$ for some segment in s
S^h	Subset of shifts that start on day $h \in H$
S^e	Subset of shifts $s \in S$ assignable to employee $e \in E$, where e is qualified for all jobs $j \in J^s$ and available during $[a_s, b_s]$
E_p^j	Subset of employees qualified for job $j \in J$ and available at period $p \in P$
F	Set of steps for the overcoverage penalty stepwise function
Q	Set of steps for the employee remuneration stepwise function

Table 6.2 Decision Variables

Variable	Description
$z_{s,e}$	Binary variable equal to 1 if shift $s \in S^e$ is assigned to employee $e \in E$, 0 otherwise
$u_{s,j}$	Number of anonymous shifts $s \in S^j$ assigned for job $j \in J$
$v_{p,j,f}$	Number of overcoverages for job $j \in J$ at period $p \in P$ on step $f \in F$
$w_{e,q}$	Number of periods worked by employee $e \in E$ on remuneration step $q \in Q$
$o_{e,h}$	Binary variable equal to 1 if employee $e \in E$ is off on day $h \in H^e$, 0 otherwise

Table 6.3 Parameters

Parameter	Description
d_p^j	Number of employees required for job $j \in J$ at period $p \in P$
g_s	Number of segments in shift $s \in S$
a_s, b_s	Start and end periods of shift $s \in S$
$\alpha_{s,j}, \beta_{s,j}$	Start and end periods of the segment for job $j \in J^s$ in shift $s \in S$
k_s	Duration of shift $s \in S$ in periods, $k_s = b_s - a_s + 1$
n_e^O	Minimum number of days off for employee $e \in E$
n^R	Minimum number of rest periods between consecutive shifts
c_s^A	Cost of anonymous shift $s \in S$ per period
c_q^E	Employee remuneration per period on step $q \in Q$
m_q	Maximum number of periods on remuneration step $q \in Q$
c_f^V	Cost per overcoverage on step $f \in F$
m_f	Maximum number of overcoverages on step $f \in F$
c^M	Cost of a transition between two segments of a shift
n_{min}^S, n_{max}^S	Minimum and maximum duration (in periods) of a shift
n_{min}^K	Minimum duration (in periods) of a segment in a shift

6.3.3 Mathematical Model

The MJTSP is formulated as a MILP model, adapting the set-covering model from [96, 110].

Assuming that each shift $s \in S$ is explicitly enumerated, this model is as follows:

$$\begin{aligned}
\min \quad & \sum_{j \in J} \sum_{s \in S^j} c_s^A u_{s,j} + \sum_{e \in E} \sum_{q \in Q} c_q^E w_{e,q} \\
& + \sum_{j \in J} \sum_{p \in P} \sum_{f \in F} c_f^V v_{p,j,f} + \sum_{e \in E} \sum_{s \in S^e} c^M (g_s - 1) z_{s,e}
\end{aligned} \tag{6.1}$$

$$\text{s.t.} \quad \sum_{e \in E_p^j} \sum_{s \in S^e \cap S_j^p} z_{s,e} + \sum_{s \in S_j^p} u_{s,j} - \sum_{f \in F} v_{p,j,f} = d_p^j, \quad \forall p \in P, j \in J \tag{6.2}$$

$$\sum_{s \in S^e \cap S^h} z_{s,e} + o_{e,h} = 1, \quad \forall e \in E, h \in H^e \quad (6.3)$$

$$\sum_{s \in S^e} k_s z_{s,e} - \sum_{q \in Q} w_{e,q} = 0, \quad \forall e \in E \quad (6.4)$$

$$\sum_{h \in H} o_{e,h} \geq n_e^O, \quad \forall e \in E \quad (6.5)$$

$$\sum_{s \in S^e \cap S^{h+1}} a_s z_{s,e} \geq \sum_{s \in S^e \cap S^h} (b_s + 1 + n^R) z_{s,e} - M o_{e,h+1},$$

$$\forall e \in E, h \in H^e \setminus \{7\} \text{ such that } h+1 \in H^e \quad (6.6)$$

$$z_{s,e} \in \{0, 1\}, \quad \forall e \in E, s \in S^e \quad (6.7)$$

$$u_{s,j} \geq 0, \text{ integer}, \quad \forall j \in J, s \in S^j \quad (6.8)$$

$$v_{p,j,f} \in [0, m_f], \quad \forall p \in P, j \in J, f \in F \quad (6.9)$$

$$w_{e,q} \in [0, m_q], \quad \forall e \in E, q \in Q \quad (6.10)$$

$$o_{e,h} \in \{0, 1\}, \quad \forall e \in E, h \in H^e \quad (6.11)$$

The objective function (6.1) accounts for anonymous-shift usage, stepwise remuneration, stepwise overcoverage penalties, and the cost of transition between jobs. Constraint (6.2) enforces job-by-period coverage by personalized and anonymous shifts, net of overcoverage. Constraint (6.3) encodes exactly one shift or a day off per employee and day that they are available. Constraints (6.4) aggregate worked periods onto remuneration steps for each employee. Constraints (6.5) impose the minimum number of days off for each employee and each day that they work. Constraints (6.6) ensure minimum rest time between consecutive days using $M = \max_{s \in S} \{b_s + 1 + n^R\}$, assuming $a_s \geq 0$ for all $s \in S$. Finally, variable domains are given in (6.7)–(6.11).

6.4 Methodology

The primary challenge in solving the MJTSP lies in the combinatorial explosion of possible shift configurations due to flexible start periods, end periods, durations, and the inclusion of multiple jobs per shift. Enumerating all possible multi-job shifts results in a prohibitively large set of variables, rendering exact MILP solvers computationally intractable for large-scale retail instances. The goal of this paper is to develop an efficient approach to enumerate a reduced set of promising shifts, which are likely to contribute to near-optimal schedules, while maintaining solution quality and computational tractability.

Inspired by the deep learning-based heuristic proposed by [96], we devise a learning-enhanced methodology in two phases: (1) a machine learning phase to predict shift start and end peri-

ods of job segments and narrows down the set of candidate shifts to only the most promising ones, and (2) an optimization phase to solve the MILP model (6.1)–(6.11) with a reduced shift set. This approach leverages training data to guide shift enumeration, significantly reducing the number of variables while preserving the ability to meet job demands, respect employee qualifications, and minimize costs. The methodology is outlined in the following steps, detailed in subsequent subsections.

6.4.1 Step 1: Shift Enumeration

We begin by enumerating all feasible shifts, considering the flexible scheduling parameters and problem constraints. Each shift $s \in S$ is defined by:

- A start time a_s and end time b_s , where $a_s, b_s \in P$ and the shift duration $k_s = b_s - a_s + 1$ lies within the predefined range $[n_{min}^S, n_{max}^S]$.
- A set of g_s segments, where each segment is associated with an interval $[\alpha_{s,j}, \beta_{s,j}]$ that respects the minimum segment duration n_{min}^K .
- Employee qualification constraints, where a shift s is only assignable to employees $e \in E$ who are qualified for all jobs $j \in J^s$ and available during $[a_s, b_s]$.

To enumerate shifts, we generate all possible combinations of start times, end times, and job assignments, subject to the following rules:

- **Mono-job Shifts:** For $g_s = 1$, a shift involves a single job $j \in J$ over $[a_s, b_s]$.
- **Multi-job Shifts:** For $g_s > 1$, a shift involves g_s segments, with $g_s - 1$ transition points $t_s \in [a_s + n_{min}^K, b_s - n_{min}^K + 1]$ ensuring all segments meet the minimum duration requirement.
- **Feasibility Checks:** Shifts are filtered to ensure compatibility with employee availability, qualifications, and operational constraints (e.g., shifts start and end within the store’s operating hours, typically 06:00 to 22:00).

This enumeration results in a large set S , potentially containing millions of shifts for a typical retail instance (e.g., 10 jobs, 64 periods per day, and flexible durations). To make the problem tractable, we proceed to reduce this set using machine learning.

6.4.2 Step 2: Data Generation

Training relies on solved historical instances (real and synthetic). Each data point corresponds to a job–day pair (j, h) and provides features over the day's n^P periods, together with binary labels for segment boundaries. For the features, we consider two sets that yield two different ML models. In the first set, we only use the information stemming from the demands for job j on day h , and from the qualifications and the availability of the employees. For the second set, we also consider information obtained from the optimal solution of the linear relaxation of model (6.1)–(6.11). Let $\tilde{p}(h)$ index the first period of day h and $W_h = \{\tilde{p}(h), \dots, \tilde{p}(h) + n^P - 1\}$. For instance i , job j , day h , we define

$$\begin{aligned} X_1^{(i)} &= (d_{\tilde{p}(h)}^j, d_{\tilde{p}(h)+1}^j, \dots, d_{\tilde{p}(h)+n^P-1}^j), \\ X_2^{(i)} &= (|E_{\tilde{p}(h)}^j|, |E_{\tilde{p}(h)+1}^j|, \dots, |E_{\tilde{p}(h)+n^P-1}^j|), \end{aligned}$$

and we add relaxation-derived signals from the model's linear relaxation $R(i)$ (when solvable within memory/time)

$$\begin{aligned} X_3^{(i)} &= (\nu_{\tilde{p}(h)}^j, \dots, \nu_{\tilde{p}(h)+n^P-1}^j), \\ X_4^{(i)} &= (\mu_{\tilde{p}(h)}^j, \dots, \mu_{\tilde{p}(h)+n^P-1}^j), \end{aligned}$$

where

$$\nu_{\tilde{p}(h)+\ell-1}^j = \max_{e \in E^j, s \in S^j} \{z_{s,e}^{R(i)} \mid \alpha_{s,j} = \tilde{p}(h) + \ell - 1\}, \quad \ell = 1, \dots, n^P, \quad (6.12)$$

$$\mu_{\tilde{p}(h)+\ell-1}^j = \max_{e \in E^j, s \in S^j} \{z_{s,e}^{R(i)} \mid \beta_{s,j} = \tilde{p}(h) + \ell - 1\}, \quad \ell = 1, \dots, n^P, \quad (6.13)$$

and $z_{s,e}^{R(i)} \in \{0, 1\}$ denotes the value of the corresponding $z_{s,e}$ variable in the optimal solution of $R(i)$. The vectors $X_3^{(i)}$ and $X_4^{(i)}$ indicate for each period of day h if this period is used as a segment starting and segment ending period for job j in $R(i)$'s optimal solution, respectively.

For each job $j \in J$ and period $p \in P$, the model outputs a probability score indicating the likelihood that p is a start period ($\alpha_{s,j}$) or end period ($\beta_{s,j}$) for a segment of job j in an optimal shift. For our supervised learning models, the labels are generated from the optimal or near-optimal solutions of the training instances, where each shift s in the solution specifies its job segments $(j, [\alpha_{s,j}, \beta_{s,j}])$ and, for multi-job shifts, the transition point. For each job $j \in J$ and period $p \in P$, we define two binary labels:

- **Start Label:** $y_{p,j}^{\text{start}} = 1$ if p is the start period $\alpha_{s,j}$ of a job segment for job j in any optimal shift $s \in S$ (i.e., $z_{s,e} = 1$ for some $e \in E$ or $u_{s,j} > 0$), and 0 otherwise.

- **End Label:** $y_{p,j}^{\text{end}} = 1$ if p is the end period $\beta_{s,j}$ of a job segment for job j in any optimal shift s , and 0 otherwise.

This feature and label design accounts for the multi-job shift structure, where each shift s encapsulates multiple job segments without requiring a job index in $z_{s,e}$, and incorporates the minimum segment duration constraint (n_{min}^K) from the mathematical model. The inclusion of transition point labels and job transition feasibility features specifically addresses the complexity of multi-job shifts, enabling the model to predict viable shift configurations that respect employee qualifications and operational constraints.

6.4.3 Step 3: Machine Learning Model Development

Like in [96], we use two variants of a machine learning model that differ only by their inputs. Both are 1-dimensional temporal U-shaped convolutional neural network (denoted UNet and UNet*), with encoder-decoder paths, skip connections, and channel-wise inputs (X_1, X_2 for UNet; $X_1, \dots, X_4$ for UNet*). These models are designed for precise temporal predictions and trained on the generated dataset using a weighted binary cross-entropy loss to deal with class imbalance, with labels derived from the solved instances. The reader is referred to [96] for the detailed architecture of our ML models.

After training, each model outputs, for each job $j \in J$, each period $p \in P$ and each start/end status, a prediction indicating whether p should be considered as a potential start/end period of a job segment. These outputs are converted into binary decisions (1 for candidate start/end, 0 otherwise) using the standard classification rule. A predicted value of 1 means that the period is retained as a potential boundary in the shift enumeration phase. A predicted value of 0 means that the period is recommended for exclusion; however, this exclusion is enforced only if the model's precision and recall for that period, evaluated on the validation set, both exceed a reliability threshold ζ . Otherwise, the period remains eligible for shift enumeration despite the negative prediction. The procedure used to compute ζ is detailed in Section 6.5.4.

6.4.4 Step 4: Restricted Shift Enumeration

Using the trained UNet and UNet* models, we restrict the shift enumeration to produce a reduced set $S' \subseteq S$ of promising shifts:

- **Prediction Interpretation:** For a given MJTSP instance, the UNet and UNet* models provide binary predictions for each period $p \in P$ and job $j \in J$, indicating whether p is recommended as potential start or end segment periods.

- **Period Selection Rule:** A period predicted as 1 is always retained as a candidate boundary. A period predicted as 0 is excluded from enumeration only if its precision and recall, evaluated on the validation set, both exceed the reliability threshold ζ . Otherwise, the period remains eligible for shift construction.
- **Shift Construction:** Shifts are then constructed using the retained candidate periods. For each job j and feasible pair of start and end periods $(\alpha_{s,j}, \beta_{s,j})$, mono-job shifts are generated. For multi-job shifts, we enumerate sequences of compatible contiguous job segments (i.e., a pair of consecutive segments spans $[\alpha_{s,j_1}, \beta_{s,j_1}]$ and $[\alpha_{s,j_2}, \beta_{s,j_2}]$ with $\alpha_{s,j_2} = \beta_{s,j_1} + 1$ and $j_1 \neq j_2$), ensuring that $\beta_{s,j} - \alpha_{s,j} + 1 \geq n_{min}^K$ for each segment.
- **Feasibility Check:** The constructed shifts are validated against employee qualifications, availability, and minimum segment duration constraints, forming the reduced set S' .

This conservative filtering mechanism significantly reduces the number of enumerated shifts while controlling the risk of eliminating shift components that may be important for high-quality solutions.

6.4.5 Step 5: Optimization with Reduced Shift Set

The reduced shift set S' is incorporated into model (6.1)–(6.11), replacing the full set S . The reduced model is then solved using a commercial solver (e.g., CPLEX) to obtain near-optimal employee schedules, minimizing the total cost while satisfying demand, rest, and qualification constraints. The objective function and constraints remain unchanged, but the reduced variable set decreases the model’s complexity, enabling faster solution times. In the following, the MILP-based solution approaches using UNet and UNet* to reduce the set of shifts are denoted MILP-UNet and MILP-UNet*, respectively.

6.5 Experimental Analysis

In this section, we present the numerical experiments conducted to evaluate the performance of our proposed method. First, we describe the experimental setup (Section 6.5.1), the datasets and instances used (Section 6.5.2), list the solution methods benchmarked and the evaluation metrics considered (Section 6.5.3), and calibrate the heuristics (Section 6.5.4), before analyzing the results obtained (Section 6.5.5).

6.5.1 Experimental Setup

The proposed heuristics are evaluated in different situations by means of computational experiments. They are programmed in Python 3 with libraries from scikit-learn [103] for data scaling, shift enumeration, and ML modeling. Similarly, the code for loading the input data, running the UNet and UNet* models, as well as creating the MILP models was also written in Python 3. All MILP problems were solved using the IBM ILOG CPLEX 22.1.1.0 MILP solver on a Linux machine powered by an 8-core (2 threads per core) Intel(R) Xeon(R) CPU E5-2637 v2, clocked at 3.5 GHz.

6.5.2 Datasets and Instances

Our goal is to construct a single training dataset of solved historical instances from which we can train a model to predict the starting and ending periods of shift segments for all test instances. This implies solving the full model (6.1)–(6.11) for these instances to provide start and end labels as discussed in Section 6.4.2. However, for practical MJTSP instances with several jobs, the number of variables in this model explodes so much (up to 20,000,000) that even the solution of linear relaxation is out of the scope of our system memory. Therefore, we opted to construct 5 smaller datasets, involving 2 or 3 jobs, extracted from the real datasets used by [110]. These datasets are denoted S1 to S5. Within each new dataset, we examine different types of jobs with varying demand distributions. For each job, there are multiple qualified employees ensuring demand coverage over the planning horizon. Table 6.4 provides the numbers of jobs and employees in each dataset.

Table 6.4 Numbers of jobs and employees in the training datasets

Dataset	S1	S2	S3	S4	S5
Jobs	2	2	2	3	3
Employees	14	29	31	25	36

For each dataset, we use the data generation procedure of [110] to create a total of 66 instances, with 6 designated for validation and 10 set aside for testing. This procedure keeps the skills and availability of the employees constant for the 66 instances, but randomly perturbs the demand curve of each job for each instance. The test instances do not contribute to the training process, and we refer to them as in-distribution instances in the remainder of the paper. Given that each instance is broken down into jobs and days for the construction of the training dataset, both UNet and UNet* models are built using 4704 data points each (from 56 instances $\times$ 7 days $\times$ (2+2+2+3+3) jobs).

[96] showed using an out-of-distribution analysis that mono-job MILP-UNet and MILP-UNet* models are capable of processing datasets that significantly differ from the training datasets. Consequently, to evaluate the robustness of the methods concerning multi-job shifts, we also test our approach using 8 larger datasets, denoted L1 to L8, and detailed in [110], which encompass more extensive real-world challenges. Table 6.5 presents the numbers of jobs and employees in these datasets. For each dataset, 10 test instances have been randomly generated using the procedure of [110].

Table 6.5 Number of jobs and employees in the out-of-distribution datasets

Dataset	L1	L2	L3	L4	L5	L6	L7	L8
Jobs	5	6	7	8	8	9	10	12
Employees	85	95	128	95	119	133	165	190

For each instance of each dataset, shift enumeration respects the following constraints for each shift $s \in S$.

- The start time a_s and end time b_s can be at any period $p \in P$ during which the demand of the corresponding job is strictly positive at a_s and b_s .
- The shift duration $k_s = b_s - a_s + 1$ lies within 4 to 10 hours (or 16 to 40 periods).
- The number of segments $g_s \in \{1, 2\}$.
- If $g_s = 2$, each segment lasts at least 2 hours (8 periods).
- If $g_s = 2$, there exists at least one available employee who is skilled for both jobs $j \in J^s$.
- If s is an anonymous shift, it is a mono-job shift, i.e., $g_s = 1$.

To illustrate, if there are two jobs and the demand of each job spans from 06:00 to 22:00 for a given day, we get a total of 2,450 mono-job shifts and of 21,450 multi-job shifts for this day.

6.5.3 Solution Methods and Evaluation Metrics

To solve the MJTSP, we propose in Section 6.4 the MILP-UNet and MILP-UNet* heuristics. The performance of these two heuristics will be assessed with respect to the following two baseline solution methods.

1. **Exact-Mono**: Exact algorithm where model (6.1)–(6.11) is solved considering all mono-job shifts but no multi-job shifts, a tolerance on the optimality gap of 0.01%, and a time limit of 20,000 seconds;
2. **Exact-Multi**: Exact algorithm where model (6.1)–(6.11) is solved considering all mono-job and multi-job shifts, a tolerance on the optimality gap of 0.01%, and a time limit of 20,000 seconds.

The proposed heuristics (MILP-UNet and MILP-UNet*) are evaluated using the following metrics.

- **Constraints**: Number of constraints which are equal for all methods.
- **VBP** and **VAP**: Number of variables for Exact-Mono before and after CPLEX’s preprocessing, respectively.
- **VBP-Multi** and **VAP-Multi**: Number of variables for the methods with multi-job shifts before and after CPLEX’s preprocessing, respectively.
- **VRBP** and **VRAP**: Ratio of the number of variables for a method with multi-job shifts to the number of variables for Exact-Mono before and after CPLEX’s preprocessing, respectively, computed as:

$$\text{VRBP} = \frac{\text{VBP-Multi}}{\text{VBP}} \quad \text{and} \quad \text{VRAP} = \frac{\text{VAP-Multi}}{\text{VAP}}.$$

- **CD-Mono** and **CD-Multi**: Relative cost deviation of a method m to Exact-Mono and to Exact-Multi, respectively, computed as:

$$\text{CD-Mono} = \frac{C_m - C_{\text{Exact-Mono}}}{C_{\text{Exact-Mono}}} \quad \text{and} \quad \text{CD-Multi} = \frac{C_m - C_{\text{Exact-Multi}}}{C_{\text{Exact-Multi}}},$$

where C_i , $i \in \{\text{Exact-Mono}, \text{Exact-Multi}, \text{MILP-UNet}, \text{MILP-UNet}^*\}$, is the cost of the solution obtained by method i .

- **Time**: Total runtime in seconds.

6.5.4 Calibration of Parameter ζ

The training instances are solved by Exact-Multi using CPLEX. Next, the required features and labels are extracted from the training instances and their solutions, including both linear

relaxation and optimal solutions for the MILP-UNet and MILP-UNet*. Once the MILP-UNet and MILP-UNet* models are built and trained, we evaluate their performance on the validation instances to identify the intensity level of these models on removing potential shifts in such a way that is acceptable to our computing time and quality preferences. Here, the goal is to minimize the computing time or maximize the variable reduction while preserving solution quality in an acceptable range.

As mentioned in Sections 6.4.3 and 6.4.4, we use a parameter ζ to control how much we trust the model's exclusions at the period level. For each period, we compute precision and recall on the validation set at the end of the training. Then at inference, a period predicted as 'not a start/end' by the ML model is excluded from enumeration only if both its precision and recall exceed ζ ; otherwise, the period remains eligible for enumeration to avoid mistakenly discarding useful shift components.

Assuming that each dataset corresponds to a different retail store with different variability from one week to another, we propose to adjust the value of ζ for each dataset using their validation instances. To do so, we apply a dichotomic search method on each validation instance that we denote with index v . This method works as follows. Let $e_v(\zeta')$ be the CD-Multi value of the predictor for instance v when $\zeta = \zeta'$ and let τ be an acceptable error ($\tau = 1\%$ for our tests). As we observed in our tests, we assume that $e_v(\zeta_{min}) > \tau$ and $e_v(\zeta_{max}) \leq \tau$, where $\zeta_{max} = 100\%$ is the maximal ζ value to consider for instance v . The algorithm starts by setting the search interval $[\underline{\zeta}, \bar{\zeta}] = [50, 100]$ and iteratively halves it until $\bar{\zeta} = \underline{\zeta} + 1$, ensuring that $e_v(\underline{\zeta}) > \tau$ and $e_v(\bar{\zeta}) \leq \tau$ throughout the search. Each iteration proceeds as follows:

1. Set $\zeta = \lceil ac(\bar{\zeta} - \underline{\zeta})2 \rceil$.
2. If $e_v(\zeta) > \tau$, then set $\underline{\zeta} = \zeta$; otherwise set $\bar{\zeta} = \zeta$.

When $\bar{\zeta} = \underline{\zeta} + 1$, the algorithm stops and we define the optimal value of ζ for instance v as $\zeta_v^* = \bar{\zeta}$. When all optimal values ζ_v^* have been computed for the validation instances of a dataset, we set $\zeta = \max_v \zeta_v^*$ when applying MILP-UNet or MILP-UNet* for the test instances associated with this dataset.

To summarize, if the UNet/UNet* model predicts 0 for a given period p , indicating that p is not recommended as a start or end time, this exclusion is enforced only if both the recall and precision metrics associated with period p exceed the threshold ζ on the validation set. If this reliability condition is not satisfied, period p remains eligible for shift enumeration despite the negative prediction. This strategy is used to account for potential instability

in the prediction performance across periods and ensures that periods are removed from consideration only when there is sufficient validation evidence supporting their exclusion.

6.5.5 Evaluation of the Heuristics

In this section, we present the results of the experiments carried out on the in-distribution test instances derived from the datasets listed in Table 6.4. The aim is to evaluate how the proposed heuristics MILP-UNet and MILP-UNet* perform on these instances, in comparison to baseline approaches Exact-Mono and Exact-Multi. For each method, 250 experiments are conducted, consisting of solving 10 test instances for each of the 5 datasets, repeated 5 times to reduce undesirable computing time sensitivity to variable computer load. The average results per dataset are reported in Tables 6.6 to 6.8. Detailed results can be found in Appendix B, in Tables B.1 to B.5.

The summary of problem size is presented in Table 6.6 in terms of constraint and variable numbers. Following the first column, which lists the in-distribution datasets, the second column presents the numbers of constraints, which are identical across all methods. Unlike the constraints, the number of variables may vary across the 10 test instances in each dataset; therefore, the following columns report, for each metric, the average taken over these 10 instances. The comparison between VBP and VAP for Exact-Mono reveals a slight average size reduction varying from 8.25% to 2.94% for the smallest to the largest problems. As indicated in the fifth column, Exact-Multi has, on average 16.68 times more variables before preprocessing than Exact-Mono. After the preprocessing, this ratio drops to 3.54, but still impacts performance. The heuristics reduce VRBP to 5.56 (UNet) and 5.06 (UNet*) on average, and VRAP to 0.44 and 0.32 on average, meaning fewer variables than Exact-Mono after preprocessing in some cases. This demonstrates the effectiveness of ML-driven pruning of non-promising shifts, with UNet* achieving slightly better reduction due to relaxation features.

Table 6.6 Number of constraints and variable ratios before and after presolve.

Dataset	Exact-Mono			Exact-Multi		MILP-UNet		MILP-UNet*	
	Constraints	VBP	VAP	VRBP	VRAP	VRBP	VRAP	VRBP	VRAP
S1	2,968	172,946	158,686	12.2	2.7	2.3	0.6	1.5	0.3
S2	3,268	343,884	335,198	17.0	3.2	8.3	0.5	6.9	0.4
S3	3,308	360,617	347,751	14.0	3.2	3.9	0.3	4.1	0.3
S4	4,532	373,321	352,636	18.0	4.0	3.5	0.2	4.2	0.2
S5	4,752	536,037	520,259	22.2	4.6	9.8	0.6	8.6	0.4
Average				16.68	3.54	5.56	0.44	5.06	0.32

Table 6.7 presents the computing times of the methods. Columns 2 to 5 report the average

computing times in seconds of all methods over the 10 test instances of each dataset. The results reveal that Exact-Multi requires substantially more time than Exact-Mono (average 7249s vs. 380s), reflecting the increased model complexity from multi-job enumeration. For datasets S3 and S5, the solver hit the 20,000-second time limit for 2 instances and 7 instances, respectively. The heuristics significantly mitigate this, with MILP-UNet averaging 565s and MILP-UNet* 886s. This speedup is particularly evident in larger datasets like S3 and S5, where Exact-Multi exceeds the average of 10,000s, while heuristics remain under 3,000s.

Table 6.7 Average computing times (seconds) of Exact-Mono, Exact-Multi, and heuristics.

Dataset	Exact-Mono	Exact-Multi	MILP-UNet	MILP-UNet*
S1	32	191	18	32
S2	317	2,815	164	341
S3	808	10,543	1,399	2,862
S4	78	3,795	123	112
S5	667	18,900	1,122	1,085
Average	380	7,249	565	886

Table 6.8 presents the performance of our methods in terms of solution quality. First, we compare the performance of Exact-Multi with Exact-Mono as indicated in the second column. Although we observe that allowing multi-job shifts in the Exact-Multi method reduces the total cost by an average of 0.64% compared to the Exact-Mono baseline, the average cost of dataset S5 has increased by 11.8%. The higher cost arises because, for most instances, the solution process terminates with a poor-quality solution when the time limit of 20,000 seconds is reached. The overall improvement obtained by Exact-Multi demonstrates the value of flexibility in job assignments, enabling better alignment with fluctuating demands and reducing overcoverage or remuneration costs in most cases.

Table 6.8 Cost deviations (%) for Exact-Multi and heuristics relative to Exact-Mono and Exact-Multi.

Dataset	Exact-Multi	MILP-UNet		MILP-UNet*	
	CD-Mono	CD-Mono	CD-Multi	CD-Mono	CD-Multi
S1	-2.8	-2.5	0.3	-2.8	0.0
S2	-3.7	-3.5	0.2	-3.7	0.0
S3	-1.0	-1.0	0.1	-1.0	0.0
S4	-7.5	-6.6	1.0	-7.5	0.1
S5	11.8	-5.9	-10.8	-6.2	-11.1
Average	-0.64	-3.90	-1.84	-4.24	-2.20

The MILP-UNet and MILP-UNet* heuristics achieve even greater average cost reductions relative to Exact-Mono (-3.90% and -4.24%, respectively), indicating their effectiveness in identifying high-quality solutions. Relative to Exact-Multi, the heuristics show average CD-Multi of -1.84% and -2.20%, with negative values implying that they find better solutions than Exact-Multi for dataset S5. This occurs because Exact-Multi may not always reach optimality within the 20,000-second time limit for large instances, whereas the heuristics, by reducing the problem size, explore the solution space more efficiently, highlighting their robustness in challenging cases where the full enumeration struggles. For the other four datasets, MILP-UNet* achieves solution costs that are essentially identical to those obtained by Exact-Multi (within 0.1%), while MILP-UNet produces solutions with slightly higher costs (up to 1% on average) but with significantly shorter computing times.

6.5.6 Robustness Analysis

To assess generalizability, we test the MILP-UNet heuristic on out-of-distribution instances from datasets L1 to L8 (see Table 6.5) with more jobs and different demand patterns and employee qualifications. Exact-Multi fails to solve the linear relaxation of these large instances, which prevents us from extracting the linear relaxation-derived features and testing MILP-UNet*. Therefore, for these experiments, we proceed only with MILP-UNet, using the UNet model trained on the small instances.

For these large instances (with up to 12 jobs and 190 employees), the average results for each dataset are reported in Table 6.9. Detailed results for the individual instances can be found in Appendix B, in Tables B.6 to B.13. The results show that MILP-UNet maintains a VRBP of 5.14 (similar to training datasets) and VRAP of 0.21, indicating effective variable reduction even when out of distribution. The Exact-Mono approach requires, on average 9,583 seconds to solve these large instances, whereas MILP-UNet reduces the average computing time to 2,619 seconds, corresponding to a speed-up factor of approximately 3.7. Although the average computing times are relatively large for some instances (particularly for datasets L7 and L8, which correspond to cases where Exact-Mono hits the time limit), the results demonstrate that the proposed approach remains practical for large-scale applications. An average CD-Mono of -3.66% shows consistent cost improvements over mono-job baselines. The average variations in CD-Mono (-1.3% to -5.6%) suggest robustness across diverse demand patterns, though longer computing times in datasets L3, L7 and L8 correlate with more jobs, highlighting scalability limits that future work could address.

Table 6.9 Performance metrics on out-of-distribution datasets.

Dataset	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
L1	1,193,360	1,159,209	1,867	6.9	0.4	569	-1.3
L2	1,333,421	1,281,532	4,197	8.0	0.4	1,373	-4.1
L3	1,840,195	1,789,951	17,641	6.1	0.2	3,914	-3.1
L4	1,393,474	1,300,789	1,858	2.5	0.1	192	-5.3
L5	1,774,671	1,710,613	4,429	5.9	0.2	1,780	-3.2
L6	2,042,711	1,945,212	6,675	3.6	0.1	1,838	-3.3
L7	2,454,925	2,337,631	20,000	3.8	0.1	6,025	-3.4
L8	2,780,224	2,647,628	20,000	4.3	0.2	5,263	-5.6
Average	1,851,623	1,771,571	9,583	5.14	0.21	2,619	-3.66

6.6 Conclusion

This study extended the flexible personnel scheduling framework from previous studies to incorporate multi-job shifts. By proposing ML-driven heuristics to restrict shift enumeration in a shift-based set-covering MILP model of the MJTSP, we achieved significant reductions in computing time while maintaining or improving solution quality. Experimental results demonstrated average cost savings of up to 4.22% over mono-job baselines and speedups over full multi-job enumeration, with strong robustness on out-of-distribution data. Insights from the analysis reveal that multi-job flexibility generally reduces costs by better matching demands, but may increase them in specialized scenarios, underscoring the need for instance-specific evaluation. The heuristics' ability to outperform Exact-Multi in some cases highlights the value of size reduction in navigating large search spaces.

Potential extensions include allowing more than two jobs per shift, incorporating break placements, extending to longer horizons with complex days-off patterns, or integrating uncertainty in demands via stochastic models. These advancements could further bridge the gap between academic research and industrial applications in personnel scheduling.

CHAPTER 7 GENERAL DISCUSSION

This thesis presents new methods to address the complexity of large-scale FPSPs in the retail sector. In retail environments, customer arrival rates vary significantly throughout the day and week, which creates highly variable staffing requirements. As a result, flexibility in employees' work schedules is typically prioritized over individual scheduling preferences. When such problems are tackled using mathematical programming approaches that explicitly model a decision variable for each possible assignment of a shift to an employee and a job, the primary source of computational complexity stems from the size of the problem and the number of decision variables rather than from the mathematical formulation itself.

In explicit set-covering formulations commonly used in retail workforce management systems, each feasible shift must be enumerated beforehand. However, the number of feasible shifts grows rapidly when shifts can start and end at many time points and when flexible durations and job assignments are allowed. This combinatorial explosion makes the direct solution of large instances extremely challenging, even when using modern MILP solvers. To address this difficulty, this thesis proposes the use of ML and DM techniques to guide the shift enumeration process and significantly reduce the number of decision variables while maintaining high-quality solutions.

The first contribution of this thesis investigates how ML can assist the enumeration of shifts in large-scale personnel scheduling problems with flexible start and end times. In this work, deep learning models are trained using previously solved FPSPs to identify promising time periods for shift starting and ending times. By learning patterns present in training data, these models allow the shift enumeration process to focus only on the most promising candidates. The resulting reduced set of shifts can then be used within a MILP formulation to obtain high-quality schedules with substantially lower computational effort. Computational experiments show that this learning-assisted approach can significantly reduce the number of decision variables while maintaining near-optimal solution quality.

The second contribution of this thesis further explores the use of data-driven techniques for reducing the search space of personnel scheduling models. Instead of relying solely on ML predictions, this work investigates the use of DM techniques to exploit similarities between scheduling instances. In particular, a k -nearest neighbors approach is used to identify previously solved instances that exhibit similar demand patterns. The shifts appearing in the optimal schedules of these similar instances are then used to guide the shift enumeration process for new problems. This strategy allows the model to restrict the enumeration of

shifts to those that are more likely to appear in good solutions. The computational results demonstrate that this approach can substantially reduce the problem size while preserving solution quality.

The third contribution of this thesis extends the first approach to the more complex case of multi-job shift scheduling. In many retail environments, employees may perform several different jobs during a single shift, such as cashiering, stocking, or assisting customers. This additional flexibility can improve workforce utilization and reduce labor costs, but it also greatly increases the number of possible shift configurations. To address this challenge, the mathematical model proposed in earlier work is extended to incorporate multi-job shifts, where each shift may consist of multiple job segments. The ML framework is also adapted to predict the start and end times of these job segments. This extension demonstrates that learning-based shift enumeration methods remain effective even in significantly more complex scheduling settings.

From a methodological perspective, the contributions of this thesis highlight the benefits of integrating data-driven techniques with traditional optimization models. Rather than replacing mathematical programming approaches, ML and DM are used here as complementary tools that help reduce the size of the optimization model before it is solved. This hybrid strategy preserves the modeling flexibility and optimality guarantees of MILP formulations while improving their scalability for large real-world instances.

Another important aspect of this work is its relevance for industrial applications. The approaches proposed in this thesis were developed in collaboration with the workforce management company UKG, whose scheduling systems rely on explicit shift-based formulations. UKG is a global provider of human capital management and workforce management software, serving hundreds of organizations across industries. In UKG’s scheduling architecture, all feasible shifts are enumerated explicitly before the MILP optimization model is solved — a design choice that maximizes modeling flexibility and cost expressiveness but creates significant computational challenges as instance size grows. A key constraint of this industrial context is that replacing or fundamentally altering this MILP-based architecture is not feasible, as it is deeply integrated into production systems. The methods developed in this thesis are therefore explicitly designed to be compatible with UKG’s existing solver pipeline: rather than proposing a new solution paradigm, they reduce the number of shifts enumerated before the MILP is solved, thereby reducing problem size without modifying the downstream optimization. In such environments, feasible shifts are typically enumerated before optimization. The methods developed in this thesis provide practical ways to control the size of this enumeration process, making it possible to solve large scheduling problems that would

otherwise be computationally intractable.

More generally, the findings of this thesis demonstrate that available data produced by scheduling methods can be leveraged to enhance future optimization procedures. PSPs are inherently repetitive: similar demand patterns and operational constraints tend to recur over time. By exploiting the previously solved and historical problems through ML and DM techniques, it becomes possible to guide optimization algorithms toward promising regions of the solution space. This perspective opens new opportunities for combining predictive analytics with combinatorial optimization in workforce management.

While ML and DM techniques have been applied to combinatorial optimization broadly — including routing, scheduling, and resource allocation problems — their systematic integration into the shift enumeration step of large-scale FPSPs with flexible start and end times in the retail sector remains largely unexplored. To the best of our knowledge, this thesis is among the first studies to address this specific gap. While previous work in personnel scheduling has primarily focused on developing stronger mathematical formulations or advanced decomposition methods such as column generation, relatively little attention has been given to exploiting historical scheduling data to guide the optimization process itself. The methods proposed in this thesis demonstrate that data-driven techniques can effectively identify promising regions of the solution space and significantly reduce the size of explicit set-covering models without compromising solution quality. By combining predictive models with mathematical programming, this work contributes to exploiting the strengths of ML and combinatorial optimization for better workforce scheduling and provides practical approaches that can be applied to real-world scheduling systems used in the retail industry.

It is natural to ask why the proposed data-driven approach is preferable to a column generation algorithm in this context. Column generation avoids explicit enumeration by generating only promising shifts on the fly via a pricing subproblem. However, in the FPSP setting with complex labor regulations, heterogeneous employees, and fine-grained time discretizations, the pricing subproblem itself is NP-hard and must be solved repeatedly, often dominating total computational effort. Furthermore, as noted in Chapter 2, UKG’s industrial architecture relies on an explicit set-covering formulation, and replacing it with a column generation scheme would require redesigning core components of the production system, a change that falls outside the scope of this research. The ML-based approach offers an alternative that achieves substantial problem size reduction while remaining fully compatible with the existing solver pipeline. Importantly, the design choices within the ML and optimization components — including the choice of multi-label classification, the use of per-period recall and precision thresholds, and the decomposition of instances by job and day — were motivated by the need

to produce predictions that directly map to the structure of the MILP variable set, making the integration with the MILP solver straightforward and computationally efficient.

CHAPTER 8 CONCLUSION

In this thesis, we presented new data-driven approaches to address the computational complexity of large-scale FPSPs arising in the retail sector. These problems are characterized by highly variable demand patterns and by the need to generate flexible shifts that adapt to these fluctuations. When explicit set-covering formulations are used, which is common in industrial workforce management systems, the enumeration of all feasible shifts can lead to extremely large optimization models.

The objective of this thesis was therefore to develop methods capable of reducing the number of candidate shifts considered in the optimization model while maintaining high-quality schedules. To achieve this goal, we investigated the use of ML and DM techniques to guide the shift enumeration process. These techniques exploit information contained in previously solved instances to identify promising scheduling decisions and restrict the size of the resulting optimization models.

In this chapter, we summarize the contributions of this thesis in Section 8.1. In Section 8.2, we discuss the limitations of the proposed approaches and present several directions for future research.

8.1 Summary of Contributions

While Chapter 7 discussed the contributions of this thesis from a methodological and industrial perspective, this section provides a concise, self-contained summary of the specific numerical contributions of each chapter — focusing on the concrete performance gains achieved in the computational experiments.

In Chapter 4, we developed a learning-based approach to identify promising shift start and end periods in large-scale FPSPs with mono-job shifts. The method uses deep neural networks trained on previously solved instances to guide the shift enumeration process and reduce the size of the resulting MILP model. Computational experiments on real retail instances with up to 10 jobs and 50 employees showed that the proposed method can speed up the computation process by an average factor of 47 while maintaining the cost increase to 1.28% on average.

In Chapter 5, we proposed a data mining-based method that exploits similarities between historical scheduling instances to guide the enumeration of shifts. Using the k -nearest neighbors algorithm, the approach identifies past instances with similar demand patterns and restricts the shift enumeration process to patterns observed in their solutions. Experiments

conducted on instances with up to 12 jobs and 252 employees showed that this method removes on average 85.5% of decision variables, leading to an average speed-up factor of 15.5 while increasing costs by only about 1.2% on average.

In Chapter 6, we extended the learning-based framework to the more complex setting of multi-job shift scheduling, where employees may perform multiple jobs within a single shift. We proposed an explicit set-covering formulation including mono-job and multi-job shifts and used a machine learning model to predict promising start and end periods for job segments, thereby reducing the number of variables generated. Computational experiments on real and synthetic datasets showed that the proposed approach maintains the scalability benefits of the previous methods while enabling cost reductions of up to 4.24% compared to schedules restricted to mono-job shifts.

Overall, the contributions of this thesis demonstrate that integrating ML and DM techniques with mathematical programming models provides an effective strategy for solving large-scale FPSPs. Rather than replacing optimization models, these data-driven methods help guide the search toward promising regions of the solution space and significantly reduce the size of the optimization models that must be solved.

8.2 Limitations and Future Research Directions

Although the methods proposed in this thesis provide significant improvements in the tractability of large-scale personnel scheduling problems, several limitations remain and open the door to future research.

A first limitation concerns the reliance on historical data to train ML models. The effectiveness of the proposed methods depends on the availability of previously solved instances that are representative of future scheduling scenarios. In environments where demand patterns change significantly over time, additional mechanisms may be required to continuously update the training data and adapt the models accordingly.

Another limitation relates to the generation of training datasets for the ML models. In this thesis, training instances were derived from smaller scheduling problems to ensure that optimal or near-optimal solutions could be obtained. While this strategy proved effective in practice, future research could investigate techniques for generating training data directly from large-scale real-world instances.

A related and important limitation concerns the quality of the historical solutions used to guide shift enumeration. The present thesis mitigates this risk by constructing training data from near-optimal solutions computed by a MILP solver on smaller instances. However, in

real deployments where only historical heuristic solutions are available, the quality of those solutions directly conditions the quality of the learned shift enumeration strategy. If such suboptimal historical solutions are used to restrict the shift enumeration, the optimization algorithm could be guided toward poor regions of the solution space, systematically excluding high-quality shifts from consideration. In practice, managers review and manually adjust the schedules obtained from the solver before posting them. The consistent monitoring of the solutions would prevent the potential risk of optimization algorithm to be guided toward poor regions of the solution space. Evaluating the robustness of the proposed methods to degraded historical solution quality constitutes an important direction for future research.

A further limitation concerns the modeling assumptions used in the multi-job scheduling framework. In the present work, the structure of multi-job shifts was restricted to a limited number of job segments in order to control the complexity of the enumeration process. Extending the approach to more general shift structures involving multiple job transitions or more complex operational constraints could be explored in future work.

Another promising direction for future research lies in the integration of predictive demand models with the scheduling process. Retail demand forecasts are typically subject to uncertainty, and incorporating stochastic or robust optimization approaches into the proposed framework could improve the resilience of the resulting schedules.

Finally, the integration of ML techniques directly within optimization algorithms represents an exciting avenue for further investigation. For example, reinforcement learning or adaptive learning mechanisms could be used to dynamically guide the enumeration of shifts during the optimization process rather than relying solely on offline training.

In conclusion, this thesis demonstrates that the combination of data-driven techniques with traditional optimization models offers a promising approach for addressing the computational challenges of large-scale FPSPs. By exploiting historical data to guide the shift enumeration process, the proposed methods significantly reduce the size of explicit set-covering models while maintaining high-quality solutions. These results open new perspectives for the integration of ML and combinatorial optimization in workforce scheduling and other large-scale decision-making problems.

REFERENCES

- [1] L. C. Edie, "Traffic delays at toll booths," *Journal of the Operations Research Society of America*, vol. 2, no. 2, pp. 107–138, 1954.
- [2] E. H. Özder, E. Özcan, T. Eren *et al.*, "A systematic literature review for personnel scheduling problems," *International Journal of Information Technology & Decision Making*, vol. 19, no. 6, pp. 1695–1735, 2020.
- [3] G. M. Thompson, "Improved implicit optimal modeling of the labor shift scheduling problem," *Management Science*, vol. 41, no. 4, pp. 595–607, 1995.
- [4] C.-G. Quimper and L.-M. Rousseau, "A large neighbourhood search approach to the multi-activity shift scheduling problem," *Journal of Heuristics*, vol. 16, no. 3, pp. 373–392, 2010.
- [5] G. B. Dantzig, "Letter to the editor—a comment on Edie's "Traffic delays at toll booths"," *Journal of the Operations Research Society of America*, vol. 2, no. 3, pp. 339–341, 1954.
- [6] S. E. Bechtold and M. J. Brusco, "Working set generation methods for labor tour scheduling," *European Journal of Operational Research*, vol. 74, no. 3, pp. 540–551, 1994.
- [7] W. B. Henderson and W. L. Berry, "Heuristic methods for telephone operator shift scheduling: An experimental analysis," *Management Science*, vol. 22, no. 12, p. 1372, 1976.
- [8] A. Mehrotra, K. E. Murphy, and M. A. Trick, "Optimal shift scheduling: A branch-and-price approach," *Naval Research Logistics*, vol. 47, no. 3, pp. 185–200, 2000.
- [9] M. J. Brusco, T. R. Johns, and R. R. Venkataraman, "Lp-based working subsets for personnel scheduling: evaluation and augmentation," *European Journal of Industrial Engineering*, vol. 12, no. 2, pp. 175–198, 2018.
- [10] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, vol. 153, no. 1, pp. 3 – 27, 2004. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S037722170300095X>

- [11] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck, “Personnel scheduling: A literature review,” *European Journal of Operational Research*, vol. 226, no. 3, pp. 367–385, 2013.
- [12] K. R. Baker, “Workforce allocation in cyclical scheduling problems: A survey,” *Journal of the Operational Research Society*, vol. 27, no. 1, pp. 155–167, 1976. [Online]. Available: <https://doi.org/10.1057/jors.1976.30>
- [13] P. Brucker, R. Qu, and E. Burke, “Personnel scheduling: Models and complexity,” *European Journal of Operational Research*, vol. 210, no. 3, pp. 467–473, 2011. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221710007897>
- [14] Y. Qu and T. Curtois, *Solving the Multi-activity Shift Scheduling Problem using Variable Neighbourhood Search*. Scitepress, Feb. 2020, vol. 1, pp. 227–232, international Conference on Operations Research and Enterprise Systems, ICORES ; Conference date: 22-02-2020 Through 24-02-2020. [Online]. Available: <http://www.icores.org/?y=2020>
- [15] M. Römer, “Block-based state-expanded network models for multi-activity shift scheduling,” *J. of Scheduling*, vol. 27, no. 4, p. 341–361, Jul. 2023. [Online]. Available: <https://doi.org/10.1007/s10951-023-00789-3>
- [16] A. Nasirian, L. Zhang, A. M. Costa, and B. Abbasi, “Multiskilled workforce staffing and scheduling: A logic-based benders’ decomposition approach,” *European Journal of Operational Research*, vol. 323, no. 1, pp. 20–33, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221724009020>
- [17] S. Dahmen and M. Rekik, “Solving multi-activity multi-day shift scheduling problems with a hybrid heuristic,” *Journal of Scheduling*, vol. 18, no. 2, pp. 207–223, 2015.
- [18] N. A. Hernández-Leandro, V. Boyer, M. A. Salazar-Aguilar, and L.-M. Rousseau, “A matheuristic based on lagrangian relaxation for the multi-activity shift scheduling problem,” *European Journal of Operational Research*, vol. 272, no. 3, pp. 859–867, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221718306155>
- [19] L. K. Trautsch and B. Kovari, “A simulated annealing approach to the multi-activity multi-day shift scheduling problem.”

- [20] F. Hagemann, T. F. Porrmann, and M. Römer, “Multi-activity shift scheduling under uncertainty: The value of shift flexibility,” *European Journal of Operational Research*, vol. 323, no. 3, pp. 988–998, 2025.
- [21] A. Caprara, M. Monaci, and P. Toth, “Models and algorithms for a staff scheduling problem,” *Mathematical Programming*, vol. 98, no. 1-3, pp. 445–476, 2003.
- [22] J. F. Bard and H. W. Purnomo, “A column generation-based approach to solve the preference scheduling problem for nurses with downgrading,” *Socio-Economic Planning Sciences*, vol. 39, no. 3, pp. 193–213, 2005.
- [23] J. F. Bard and H. W. Purnomo, “Preference scheduling for nurses using column generation,” *European Journal of Operational Research*, vol. 164, no. 2, pp. 510–534, 2005.
- [24] J. Beliën and E. Demeulemeester, “A branch-and-price approach for integrating nurse and surgery scheduling,” *European journal of operational research*, vol. 189, no. 3, pp. 652–668, 2008.
- [25] J. O. Brunner and R. Stolletz, “Stabilized branch and price with dynamic parameter updating for discontinuous tour scheduling,” *Computers & Operations Research*, vol. 44, pp. 137–145, 2014.
- [26] D. Attia, R. Bürgy, G. Desaulniers, and F. Soumis, “A decomposition-based heuristic for large employee scheduling problems with inter-department transfers,” *EURO Journal on Computational Optimization*, vol. 7, no. 4, pp. 325–357, 2019.
- [27] S. Dahmen, M. Rekik, F. Soumis, and G. Desaulniers, “A two-stage solution approach for personalized multi-department multi-day shift scheduling,” *European Journal of Operational Research*, vol. 280, no. 3, pp. 1051–1063, 2020.
- [28] S. L. Moondra, “An lp model for work force scheduling for banks,” *Journal of Bank Research*, vol. 7, no. 4, pp. 299–301, 1976.
- [29] S. E. Bechtold and L. W. Jacobs, “Implicit modeling of flexible break assignments in optimal shift scheduling,” *Management Science*, vol. 36, no. 11, pp. 1339–1351, 1990.
- [30] S. E. Bechtold and L. W. Jacobs, “Improvement of labour utilisation in shift scheduling for services with implicit optimal modelling,” *International Journal of Operations & Production Management*, vol. 41, no. 4, pp. 595–607, 1991.
- [31] T. Aykin, “Optimal shift scheduling with multiple break windows,” *Management Science*, vol. 42, no. 4, pp. 591–602, 1996.

- [32] T. Aykin, “A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows,” *Journal of the Operational Research Society*, vol. 49, no. 6, pp. 603–615, 1998.
- [33] M. Rekik, J.-F. Cordeau, and F. Soumis, “Using Benders decomposition to implicitly model tour scheduling,” *Annals of Operations Research*, vol. 128, no. 1-4, pp. 111–133, 2004.
- [34] M. Rekik, J.-F. Cordeau, and F. Soumis, “Implicit shift scheduling with multiple breaks and work stretch duration restrictions,” *Journal of Scheduling*, vol. 13, no. 1, pp. 49–75, 2010.
- [35] M.-C. Côté, B. Gendron, C.-G. Quimper, and L.-M. Rousseau, “Formal languages for integer programming modeling of shift scheduling problems,” *Constraints*, vol. 16, no. 1, pp. 54–76, 2011.
- [36] M.-C. Côté, B. Gendron, and L.-M. Rousseau, “Grammar-based integer programming models for multiactivity shift scheduling,” *Management Science*, vol. 57, no. 1, pp. 151–163, 2011.
- [37] V. Boyer, B. Gendron, and L.-M. Rousseau, “A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem,” *Journal of Scheduling*, vol. 17, no. 2, pp. 185–197, 2014.
- [38] M. I. Restrepo, B. Gendron, and L.-M. Rousseau, “Combining Benders decomposition and column generation for multi-activity tour scheduling,” *Computers & Operations Research*, vol. 93, pp. 151–165, 2018.
- [39] H. Ni and H. Abeledo, “A branch-and-price approach for large-scale employee tour scheduling problems,” *Annals of Operations Research*, vol. 155, no. 1, pp. 167–176, 2007.
- [40] J. Beliën and E. Demeulemeester, “On the trade-off between staff-decomposed and activity-decomposed column generation for a staff scheduling problem,” *Annals of Operations Research*, vol. 155, no. 1, pp. 143–166, 2007.
- [41] Q. Lequy, M. Bouchard, G. Desaulniers, F. Soumis, and B. Tachefine, “Assigning multiple activities to work shifts,” *Journal of Scheduling*, vol. 15, no. 2, pp. 239–251, 2012.

- [42] Q. Lequy, G. Desaulniers, and M. M. Solomon, “A two-stage heuristic for multi-activity and task assignment to work shifts,” *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 831–841, 2012.
- [43] Q. Lequy, G. Desaulniers, and M. M. Solomon, “Assigning team tasks and multiple activities to fixed work shifts,” *INFOR: Information Systems and Operational Research*, vol. 51, no. 2, pp. 64–75, 2013.
- [44] M.-C. Côté, B. Gendron, and L.-M. Rousseau, “Grammar-based column generation for personalized multi-activity shift scheduling,” *INFORMS Journal on Computing*, vol. 25, no. 3, pp. 461–474, 2013.
- [45] A. Legrain, J. Omer, and S. Rosat, “A rotation-based branch-and-price approach for the nurse scheduling problem,” *Mathematical Programming Computation*, pp. 1–34, 2019.
- [46] M. Defraeye and I. Van Nieuwenhuyse, “A branch-and-bound algorithm for shift scheduling with stochastic nonstationary demand,” *Computers & Operations Research*, vol. 65, pp. 149–162, 2016.
- [47] G. Felici and C. Gentile, “A polyhedral approach for the staff rostering problem,” *Management Science*, vol. 50, no. 3, pp. 381–393, 2004.
- [48] E. K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem, “The state of the art of nurse rostering,” *Journal of Scheduling*, vol. 7, no. 6, pp. 441–499, 2004.
- [49] E. K. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman, “A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem,” *European Journal of Operational Research*, vol. 188, no. 2, pp. 330–341, 2008.
- [50] P. Brucker, E. K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe, “A shift sequence based approach for nurse scheduling and a new benchmark dataset,” *Journal of Heuristics*, vol. 16, no. 4, pp. 559–573, 2010.
- [51] L. Kletzander and N. Musliu, “Solving the general employee scheduling problem,” *Computers & Operations Research*, vol. 113, p. 104794, 2020. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0305054819302369>
- [52] N. Xue, D. Landa-Silva, I. Triguero, and G. P. Figueredo, “A genetic algorithm with composite chromosome for shift assignment of part-time employees,” in *2018 IEEE Congress on Evolutionary Computation*. IEEE, 2018, pp. 1–8.

- [53] S. Pan, M. Akplogan, N. Touati, L. Létocart, R. Wolfler Calvo, and L.-M. Rousseau, “A hybrid heuristic for the multi-activity tour scheduling problem,” *Electronic Notes in Discrete Mathematics*, vol. 69, pp. 333 – 340, 2018, joint EURO/ALIO International Conference 2018 on Applied Combinatorial Optimization. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1571065318301951>
- [54] R. Hassani, G. Desaulniers, and I. Elhallaoui, “Real-time personnel re-scheduling after a minor disruption in the retail industry,” *Computers & Operations Research*, vol. 120, p. 104952, 2020.
- [55] Y. Bengio, A. Lodi, and A. Prouvost, “Machine learning for combinatorial optimization: a methodological tour d’horizon,” *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.
- [56] B. Abbasi, T. Babaei, Z. Hosseini-fard, K. Smith-Miles, and M. Dehghani, “Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management,” *Computers & Operations Research*, vol. 119, p. 104941, 2020.
- [57] M. Kruber, M. E. Lübbecke, and A. Parmentier, “Learning when to use a decomposition,” in *Integration of AI and OR Techniques in Constraint Programming*. Springer International Publishing, 2017, pp. 202–210.
- [58] Á. S. Xavier, F. Qiu, and S. Ahmed, “Learning to solve large-scale security-constrained unit commitment problems,” *INFORMS Journal on Computing*, vol. 33, no. 2, pp. 739–756, 2020.
- [59] A. Lodi and G. Zarpellon, “On learning and branching: a survey,” *TOP*, vol. 25, no. 2, pp. 207–236, 2017.
- [60] M. Morabit, G. Desaulniers, and A. Lodi, “Machine-learning-based column selection for column generation,” *Transportation Science*, vol. 55, no. 4, pp. 815–831, 2021.
- [61] Y. Tang, S. Agrawal, and Y. Faenza, “Reinforcement learning for integer programming: Learning to cut,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 9367–9376.
- [62] H. Xu, C. Caramanis, and S. Mannor, “Statistical optimization in high dimensions,” *Operations Research*, vol. 64, no. 4, pp. 958–979, 2016.

- [63] A. Lodi, L. Mossina, and E. Rachelson, “Learning to handle parameter perturbations in combinatorial optimization: an application to facility location,” *EURO Journal on Transportation and Logistics*, vol. 9, no. 4, p. 100023, 2020.
- [64] M. Fischetti and M. Fraccaro, “Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks,” *Computers & Operations Research*, vol. 106, pp. 289–297, 2019.
- [65] L. Jourdan, C. Dhaenens, and E.-G. Talbi, “Using datamining techniques to help metaheuristics: A short survey,” in *International workshop on hybrid metaheuristics*. Springer, 2006, pp. 57–69.
- [66] O. Jamsheela and G. Raju, “Frequent itemset mining algorithms: a literature survey,” in *2015 IEEE international advance computing conference (IACC)*. IEEE, 2015, pp. 1099–1104.
- [67] M. R. d. H. Maia, A. Plastino, and U. d. S. Souza, “An improved hybrid genetic search with data mining for the cvrp,” *Networks*, vol. 83, no. 4, pp. 692–711, 2024.
- [68] G. Grahne and J. Zhu, “Efficiently using prefix-trees in mining frequent itemsets.” in *FIMI*, vol. 90, 2003, p. 65.
- [69] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, “A hybrid genetic algorithm for multidepot and periodic vehicle routing problems,” *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.
- [70] Y. Zhou, J.-K. Hao, and B. Duval, “Frequent pattern-based search: A case study on the quadratic assignment problem,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 3, pp. 1503–1515, 2020.
- [71] Y. Zhou, J. Li, J.-K. Hao, and F. Glover, “Detecting critical nodes in sparse graphs via “reduce-solve-combine” memetic search,” *INFORMS Journal on Computing*, vol. 36, no. 1, pp. 39–60, 2024.
- [72] E.-G. Talbi, “Machine learning into metaheuristics: A survey and taxonomy,” *ACM Comput. Surv.*, vol. 54, no. 6, Jul. 2021. [Online]. Available: <https://doi.org/10.1145/3459664>
- [73] S. Gao, Y. Wang, J. Cheng, Y. Inazumi, and Z. Tang, “Ant colony optimization with clustering for solving the dynamic location routing problem,” *Applied Mathematics and Computation*, vol. 285, pp. 149–173, 2016.

- [74] S. Erdoğan and E. Miller-Hooks, “A green vehicle routing problem,” *Transportation research part E: logistics and transportation review*, vol. 48, no. 1, pp. 100–114, 2012.
- [75] M. A. Adibi and J. Shahrabi, “A clustering-based modified variable neighborhood search algorithm for a dynamic job shop scheduling problem,” *The International Journal of Advanced Manufacturing Technology*, vol. 70, pp. 1955–1961, 2014.
- [76] M. Wu, K. Li, S. Kwong, Q. Zhang, and J. Zhang, “Learning to decompose: A paradigm for decomposition-based multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 376–390, 2018.
- [77] D. K. Saxena, J. A. Duro, A. Tiwari, K. Deb, and Q. Zhang, “Objective reduction in many-objective optimization: Linear and nonlinear algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 77–99, 2012.
- [78] M. Pal, S. Saha, and S. Bandyopadhyay, “Decor: differential evolution using clustering based objective reduction for many-objective optimization,” *Information Sciences*, vol. 423, pp. 200–218, 2018.
- [79] J. Fitzpatrick, D. Ajwani, and P. Carroll, “Learning to sparsify travelling salesman problem instances,” in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2021, pp. 410–426.
- [80] J. Lauri and S. Dutta, “Fine-grained search space classification for hard enumeration variants of subset problems,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 2314–2321.
- [81] D. Tayebi, S. Ray, and D. Ajwani, “Learning to prune instances of k-median and related problems,” in *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 2022, pp. 184–194.
- [82] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, “Exact combinatorial optimization with graph convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [83] M. Morabit, G. Desaulniers, and A. Lodi, “Learning to repeatedly solve routing problems,” 2022. [Online]. Available: <https://arxiv.org/abs/2212.08101>
- [84] R. Hassani, G. Desaulniers, and I. El Hallaoui, “A parallel ruin and recreate heuristic for personnel scheduling in a flexible working environment,” *Journal of Scheduling*, p. forthcoming, 2023.

- [85] R. Bürgy, H. Michon-Lacaze, and G. Desaulniers, “Employee scheduling with short demand perturbations and extensible shifts,” *Omega*, vol. 89, pp. 177–192, 2019.
- [86] V. A. Mabert and C. A. Watts, “A simulation analysis of tour-shift construction procedures,” *Management Science*, vol. 28, no. 5, pp. 520–532, 1982.
- [87] M. J. Brusco and L. W. Jacobs, “Starting-time decisions in labor tour scheduling: An experimental analysis and case study,” *European Journal of Operational Research*, vol. 131, pp. 459–475, 2001.
- [88] E. K. Burke, J. Li, and R. Qu, “A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems,” *European Journal of Operational Research*, vol. 203, no. 2, pp. 484–493, 2010.
- [89] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, “Learning to branch in mixed integer programming,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
- [90] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, “Binary relevance for multi-label learning: an overview,” *Frontiers of Computer Science*, vol. 12, no. 2, pp. 191–202, 2018.
- [91] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.
- [92] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [93] S. Asgari Taghanaki, K. Abhishek, J. P. Cohen, J. Cohen-Adad, and G. Hamarneh, “Deep semantic segmentation of natural and medical images: a review,” *Artificial Intelligence Review*, vol. 54, no. 1, pp. 137–178, 2021.
- [94] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.
- [95] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

- [96] F. Rastgar-Amini, C. Contardo, G. Desaulniers, and M. Gasse, “Learning to enumerate shifts for large-scale flexible personnel scheduling problems,” *Journal of Scheduling*, vol. 28, no. 4, pp. 425–443, 2025.
- [97] J. Li, E. K. Burke, and R. Qu, “A pattern recognition based intelligent search method and two assignment problem case studies,” *Applied Intelligence*, vol. 36, pp. 442–453, 2012.
- [98] A. Guastalla, E. Sulis, R. Aringhieri, S. Branchi, C. Di Francescomarino, and C. Ghidini, “Workshift scheduling using optimization and process mining techniques: An application in healthcare,” in *2022 Winter Simulation Conference (WSC)*. IEEE, 2022, pp. 1116–1127.
- [99] C. C. Aggarwal, *Data Mining: The Textbook*. Springer Publishing Company, Incorporated, 2015.
- [100] F.-P. Chan, A.-C. Fu, and C. Yu, “Haar wavelets for efficient similarity search of time-series: with and without time warping,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 686–705, 2003.
- [101] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [102] X. Wang, X. Wang, and M. Wilkes, *New Developments in Unsupervised Outlier Detection : Algorithms and Applications*. Springer, 2020.
- [103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [104] A. F. Porto, C. A. Henao, H. López-Ospina, E. R. González, and V. I. González, “Dataset for solving a hybrid flexibility strategy on personnel scheduling problem in the retail industry,” *Data in Brief*, vol. 32, p. 106066, 2020.
- [105] S. Demasse, G. Pesant, and L.-M. Rousseau, “A cost-regular based hybrid column generation approach,” *Constraints*, vol. 11, no. 4, pp. 315–333, 2006.
- [106] M. I. Restrepo, L. Lozano, and A. L. Medaglia, “Constrained network-based column generation for the multi-activity shift scheduling problem,” *International Journal of Production Economics*, vol. 140, no. 1, pp. 466–472, 2012.

- [107] M. Gérard, F. Clautiaux, and R. Sadykov, “Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce,” *European Journal of Operational Research*, vol. 252, no. 3, pp. 1019–1030, 2016.
- [108] M. I. Restrepo, B. Gendron, and L.-M. Rousseau, “Branch-and-price for personalized multiactivity tour scheduling,” *INFORMS Journal on Computing*, vol. 28, no. 2, pp. 334–350, 2016.
- [109] R. Shariat and K. Huang, “A large-scale neighborhood search algorithm for multi-activity tour scheduling problems,” *Journal of Heuristics*, vol. 30, pp. 225–267, 2024.
- [110] F. Rastgar-Amini, D. Aloise, C. Contardo, and G. Desaulniers, “Data mining-driven shift enumeration for accelerating the solution of large-scale personnel scheduling problems,” *ACM Transactions on Evolutionary Learning*, vol. 5, no. 4, pp. 1–24, 2025.

APPENDIX A DETAILED RESULTS OF CHAPTER 5

The results of our experiments are comprehensively presented for the test instances in each dataset, spanning Tables A.1 through A.8. In the *Exact* / *Tol-5%* columns, we indicate the number of z -variables and other variables in the complete MILP model (5.1)–(5.11). The *z -Removed* columns offer a comparative analysis, illustrating the percentage of z -variables removed by MILP- k NN and MILP-UNet*. The *Error* columns specify the error (in percentage) in the cost of the solution produced by each heuristic with respect to that of the Exact method. Finally, the *Time* columns disclose the computational time, in seconds, required by the Exact, MILP- k NN, MILP-UNet*, and Tol-5% algorithms to solve the given instances. Bold results are the best ones. Observe that, in Tables A.7 and A.8, no errors, nor times are provided for the Tol-5% method because no feasible solutions were computed within the time limit.

Table A.1 Dataset 1: 5 jobs, 85 employees

st.st.pt< -st.pt>	Exact / Tol-5%		z -Removed (in %)		Error (in %)			Time (in seconds)			
	z -Variables	Other Variables	MILP- k NN	MILP-UNet*	MILP- k NN	MILP-UNet*	Tol-5%	Exact	MILP- k NN	MILP-UNet*	Tol-5%
1	1,135,697	65,747	75.8	71.1	1.1	1.2	2.0	1010	173	174	795
2	1,133,505	65,668	76.5	72.2	1.6	3.3	3.2	741	228	294	605
3	1,129,170	65,475	77.4	74.1	1.1	4.0	0.5	2462	183	171	294
4	1,120,427	65,076	77.8	72.1	1.4	1.3	3.2	1415	177	187	755
5	1,125,086	65,290	78.0	72.9	1.2	1.2	2.2	868	141	184	672
6	1,123,198	65,222	77.7	73.3	0.9	4.6	1.9	927	264	189	631
7	1,121,226	65,124	78.0	72.4	1.2	1.9	1.0	874	319	817	586
8	1,128,354	65,457	78.8	73.5	1.6	4.0	0.6	1008	203	205	395
9	1,128,083	65,437	78.2	70.8	1.3	0.6	3.4	546	223	229	399
10	1,135,787	65,765	78.3	71.0	1.2	2.7	2.3	1597	230	272	513
Avg.	1,127,934	65,426	77.7	72.3	1.3	2.5	2.0	1867	214	272	564

Table A.2 Dataset 2: 6 jobs, 95 employees

st.st.pt< -st.pt>	Exact / Tol-5%		z -Removed (in %)		Error (in %)			Time (in seconds)			
	z -Variables	Other Variables	MILP- k NN	MILP-UNet*	MILP- k NN	MILP-UNet*	Tol-5%	Exact	MILP- k NN	MILP-UNet*	Tol-5%
1	1,258,007	80,187	82.3	65.8	1.5	0.6	0.5	2,956	210	230	932
2	1,257,270	80,120	82.5	65.9	1.6	0.8	2.0	2,372	150	762	1,070
3	1,255,285	79,939	82.9	65.8	1.2	0.9	0.4	4,609	254	527	1,294
4	1,251,144	79,558	82.7	66.4	1.2	1.1	1.3	5,470	273	941	1,385
5	1,250,145	79,636	83.2	69.3	1.4	4.5	1.0	4,581	408	371	1,081
6	1,251,564	79,654	83.0	66.8	0.8	1.1	2.6	6,591	213	431	1,269
7	1,249,268	79,515	82.9	66.6	1.0	0.7	0.2	1,706	291	315	1,213
8	1,252,600	79,827	82.9	66.5	1.8	0.9	0.4	4,142	281	580	1,166
9	1,253,100	79,835	83.0	66.8	1.4	0.9	0.9	5,468	285	538	1,153
10	1,256,527	80,136	82.8	65.4	1.3	0.5	0.8	3,053	306	288	761
Avg.	1,253,580	79,841	82.8	66.5	1.3	1.2	1.0	4,197	267	498	1,132

Table A.3 Dataset 3: 7 jobs, 128 employees

st.st.st.pt< -st.pt>	Exact / Tol-5%		z-Removed (in %)		Error (in %)			Time (in seconds)			
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Tol-5%	Exact	MILP-kNN	MILP-UNet*	Tol-5%
1	1,755,479	92,691	82.4	84.1	1.7	1.0	59.6	6,419	752	785	3,602
2	1,754,165	92,624	82.4	85.7	1.2	8.6	61.2	20,000	867	568	3,603
3	1,750,828	92,443	82.5	84.6	1.6	0.9	8.9	20,000	1,193	367	3,600
4	1,743,056	92,062	82.5	84.8	-0.1	-0.2	62.4	20,000	2,303	1,809	3,600
5	1,742,457	92,140	82.9	84.7	2.1	1.4	0.8	9,999	1,084	612	1,789
6	1,743,818	92,158	82.7	84.6	1.2	1.7	0.2	19,994	795	877	2,999
7	1,740,630	92,019	82.8	84.5	1.3	0.9	1.4	20,000	502	457	2,690
8	1,746,367	92,331	82.8	84.4	0.8	0.5	62.3	20,000	2,102	729	3,600
9	1,747,081	92,339	82.5	84.4	1.7	1.3	64.0	20,000	1,379	1,982	3,600
10	1,753,440	92,640	82.7	84.2	1.0	0.8	1.6	20,000	1,234	466	3,238
Avg.	1,747,850	92,345	82.6	84.6	1.2	1.7	32.2	17,641	1,221	865	3,233

Table A.4 Dataset 4: 8 jobs, 95 employees

st.st.st.pt< -st.pt>	Exact / Tol-5%		z-Removed (in %)		Error (in %)			Time (in seconds)			
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Tol-5%	Exact	MILP-kNN	MILP-UNet*	Tol-5%
1	1,290,300	103,491	80.3	71.3	0.4	1.3	0.0	1,262	251	319	820
2	1,289,950	103,477	80.7	71.9	1.4	1.2	1.6	1,515	143	256	590
3	1,290,300	103,491	81.0	71.9	0.5	2.4	1.3	1,570	191	223	840
4	1,289,936	103,477	81.4	71.5	1.9	2.1	0.3	1,110	189	236	751
5	1,289,950	103,477	81.3	71.9	1.1	2.3	3.0	3,026	211	162	861
6	1,289,600	103,463	80.7	71.5	1.1	1.6	4.2	2,304	275	213	1,108
7	1,289,936	103,477	81.0	71.7	0.8	0.8	3.9	1,926	317	178	626
8	1,289,572	103,463	81.0	71.7	1.2	1.4	0.5	1,756	185	164	776
9	1,289,936	103,477	80.9	72.0	1.2	1.4	1.8	2,086	319	246	747
10	1,289,600	103,463	80.1	71.0	0.9	1.0	3.6	2,030	197	380	1,698
Avg.	1,289,998	103,476	80.8	71.6	1.1	1.6	2.0	1,858	228	238	882

Table A.5 Dataset 5: 8 jobs, 119 employees

st.st.st.pt< -st.pt>	Exact / Tol-5%		z-Removed (in %)		Error (in %)			Time (in seconds)			
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Tol-5%	Exact	MILP-kNN	MILP-UNet*	Tol-5%
1	1,678,538	105,356	86.4	77.6	0.9	0.4	1.7	6,801	121	109	885
2	1,676,636	105,289	86.6	77.7	1.5	1.3	1.6	3,950	116	136	913
3	1,672,365	105,108	86.5	77.9	1.3	0.3	4.7	3,099	135	159	882
4	1,663,409	104,727	86.7	78.3	1.3	0.8	3.2	3,245	109	105	1,292
5	1,664,266	104,805	87.0	78.4	1.2	1.2	0.9	9,111	90	112	1,290
6	1,664,932	104,823	86.6	78.6	1.5	2.4	1.6	2,330	198	237	1,160
7	1,661,480	104,684	86.8	78.2	1.0	0.9	1.4	4,638	191	121	1,098
8	1,668,465	104,996	87.1	78.0	1.5	1.0	1.6	7,559	115	333	1,438
9	1,676,668	105,004	86.5	77.9	1.4	0.6	1.5	1,315	133	208	878
10	1,668,910	105,305	86.9	77.5	1.1	0.7	2.4	2,473	119	114	1,438
Avg.	1,669,661	105,010	86.7	78.0	1.3	1.0	2.1	4,429	133	163	1,127

Table A.6 Dataset 6: 9 jobs, 133 employees

st.st.st.pt< -st.pt>	Exact / Tol-5%		z-Removed (in %)		Error (in %)			Time (in seconds)			
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Tol-5%	Exact	MILP-kNN	MILP-UNet*	Tol-5%
1	1,924,558	117,992	79.2	78.0	1.3	1.4	2.0	9,512	425	434	1,752
2	1,925,538	118,025	79.0	78.6	0.9	1.7	74.4	4,399	401	316	3,600
3	1,924,368	117,985	79.1	78.2	1.3	2.0	5.2	5,608	618	348	3,472
4	1,924,648	117,995	79.1	78.3	0.9	2.4	0.7	7,075	709	351	2,206
5	1,923,710	117,962	79.0	77.9	1.5	3.0	0.5	11,286	426	278	3,328
6	1,924,578	117,993	79.2	78.2	0.9	1.2	2.3	4,919	327	387	3,507
7	1,925,478	118,024	79.2	78.1	1.2	1.8	3.7	4,439	406	345	2,392
8	1,924,314	117,982	79.2	78.1	0.9	2.0	1.5	3,832	360	332	2,813
9	1,924,970	118,004	79.1	79.9	1.7	2.0	1.8	4,556	510	150	1,922
10	1,925,568	118,025	78.9	77.7	1.4	1.5	0.3	9,411	552	478	3,155
Avg.	1,924,712	117,999	79.1	78.3	1.2	1.9	9.2	6,675	473	342	2,815

Table A.7 Dataset 7: 10 jobs, 165 employees

st.st.st.pt< -st.pt>	Exact / Tol-5%		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP- <i>k</i> NN	MILP-UNet*	MILP- <i>k</i> NN	MILP-UNet*	Exact	MILP- <i>k</i> NN	MILP-UNet*
1	2,327,892	131,721	94.5	88.3	1.2	-2.2	20,000	1,161	2,726
2	2,327,279	131,654	94.6	88.8	0.2	-2.7	20,000	352	6,521
3	2,325,325	131,473	94.8	88.7	1.6	-1.3	20,000	521	2,507
4	2,321,301	131,092	94.9	88.6	1.1	-2.1	20,000	455	4,227
5	2,319,815	131,170	94.8	88.5	0.9	-0.5	20,000	1,081	5,102
6	2,321,668	131,188	94.7	88.7	-0.2	-1.8	20,000	1,052	1,785
7	2,319,018	131,049	94.8	88.6	1.0	-1.3	20,000	2,038	1,599
8	2,322,563	131,361	94.9	88.4	1.6	-1.8	20,000	1,155	1,222
9	2,323,179	131,369	94.9	88.3	0.5	-0.7	20,000	613	1,070
10	2,326,202	131,670	94.8	88.3	0.7	-0.8	20,000	791	1,543
Avg.	2,323,550	131,375	94.8	88.5	0.9	-1.5	20,000	922	2,830

Table A.8 Dataset 8: 12 jobs, 190 employees

Inst.	Exact / Tol-5%		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP- <i>k</i> NN	MILP-UNet*	MILP- <i>k</i> NN	MILP-UNet*	Exact	MILP- <i>k</i> NN	MILP-UNet*
1	2,626,969	158,254	96.7	89.3	0.5	-3.4	20,000	1,809	654
2	2,626,221	158,187	96.8	89.4	2.8	-3.4	20,000	1,022	2,507
3	2,624,212	158,006	96.8	89.3	1.1	-3.1	20,000	1,643	2,492
4	2,620,163	157,625	96.9	89.5	1.4	-2.8	20,000	1,197	1,488
5	2,619,513	157,703	96.8	89.4	2.1	-3.0	20,000	1,421	6,743
6	2,620,746	157,721	96.8	89.2	1.6	-2.6	20,000	580	1,732
7	2,618,547	157,582	96.8	89.3	1.3	-2.2	20,000	2,660	847
8	2,621,906	157,894	96.8	89.0	1.8	-3.6	20,000	1,840	2,523
9	2,622,367	157,902	96.8	88.8	0.4	-2.2	20,000	303	3,761
10	2,625,681	154,588	96.8	89.0	1.0	-3.4	20,000	2,555	721
Avg.	2,622,678	157,546	96.8	89.2	1.4	-3.0	20,000	1,503	2,347

APPENDIX B DETAILED RESULTS OF CHAPTER 6

The detailed results of the computational experiments for the small (in-distribution) datasets are reported in Tables B.1 to B.5. For these datasets, the performance of the two proposed heuristics is compared with that of the Exact-Mono and Exact-Multi methods. The meaning of the columns is provided in Section 6.5.5.

Table B.1 Dataset S1: 2 jobs, 14 employees

Instance	Exact-Mono			Exact-Multi				MILP-UNet					MILP-UNet*				
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi
1	172,946	158,686	41	12.2	2.7	92	-2.4	2.5	0.6	29	-2.4	0.0	1.7	0.3	20	-2.4	0.0
2	172,946	158,686	34	12.2	2.7	209	-2.2	2.4	0.6	45	-2.2	0.0	1.6	0.3	22	-2.2	0.0
3	172,946	158,686	27	12.2	2.7	339	-3.0	2.2	0.5	40	-3.0	0.0	1.5	0.3	25	-3.0	0.0
4	172,946	158,686	28	12.2	2.7	723	-2.8	2.2	0.5	45	-1.2	1.7	1.4	0.3	24	-2.8	0.0
5	172,946	158,686	27	12.2	2.7	134	-3.3	2.2	0.5	29	-3.0	0.3	1.5	0.3	16	-3.3	0.0
6	172,946	158,686	36	12.2	2.7	63	-3.0	2.3	0.6	32	-2.7	0.4	1.5	0.3	10	-3.0	0.0
7	172,946	158,686	34	12.2	2.7	61	-3.4	2.3	0.6	24	-2.4	1.0	1.5	0.3	11	-3.4	0.0
8	172,946	158,686	32	12.2	2.7	88	-3.4	2.4	0.6	29	-3.4	0.0	1.6	0.3	13	-3.4	0.0
9	172,946	158,686	27	12.2	2.7	73	-2.7	2.3	0.6	17	-2.7	0.0	1.5	0.3	17	-2.7	0.0
10	172,946	158,686	32	12.2	2.7	125	-1.8	2.5	0.6	28	-1.8	0.0	1.7	0.3	19	-1.8	0.0
Average	172,946	158,686	32	12.2	2.7	191	-2.8	2.3	0.6	32	-2.5	0.3	1.5	0.3	18	-2.8	0.0

Table B.2 Dataset S2: 2 jobs, 29 employees

Instance	Exact-Mono			Exact-Multi				MILP-UNet					MILP-UNet*				
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi
1	347,753	338,842	168	16.8	3.1	2,177	-3.4	8.2	0.5	129	-2.8	0.6	6.0	0.3	375	-3.4	0.0
2	346,695	337,672	313	16.7	3.1	3,084	-3.7	8.2	0.5	63	-3.5	0.2	7.4	0.4	209	-3.7	0.0
3	344,513	335,769	182	17.0	3.2	1,918	-3.8	8.5	0.5	100	-3.7	0.1	7.3	0.4	324	-3.8	0.0
4	339,929	331,450	619	17.2	3.2	1,727	-5.8	8.5	0.5	196	-5.6	0.2	7.2	0.4	237	-5.8	0.0
5	342,297	333,487	714	17.1	3.2	3,825	-4.0	8.3	0.5	188	-3.9	0.1	7.1	0.4	240	-4.0	0.0
6	341,350	332,820	296	17.1	3.2	4,132	-3.6	8.4	0.5	151	-3.5	0.1	6.3	0.3	1143	-3.5	0.1
7	340,368	331,962	253	17.1	3.2	5,878	-3.0	8.3	0.5	203	-2.9	0.1	6.6	0.4	107	-3.0	0.0
8	344,101	335,492	279	17.0	3.2	1,454	-4.0	8.5	0.5	226	-3.7	0.3	6.4	0.3	275	-3.9	0.1
9	343,944	335,251	159	16.9	3.2	1,563	-3.5	8.2	0.5	270	-3.4	0.1	7.7	0.4	271	-3.5	0.0
10	347,885	339,239	186	16.8	3.2	2,395	-2.3	8.3	0.5	117	-2.2	0.0	7.5	0.4	226	-2.2	0.0
Average	343,883	335,198	317	17.0	3.2	2,815	-3.7	8.3	0.5	164	-3.5	0.2	6.9	0.4	341	-3.7	0.0

Table B.3 Dataset S3: 2 jobs, 31 employees

Instance	Exact-Mono			Exact-Multi				MILP-UNet					MILP-UNet*				
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi
1	360,617	347,751	801	14.0	3.2	10,406	-1.4	4.1	0.4	330	-1.3	0.1	3.9	0.3	470	-1.4	0.0
2	360,617	347,751	1,055	14.0	3.2	5,430	-1.4	4.0	0.4	1429	-1.4	0.0	4.4	0.4	1,008	-1.4	0.0
3	360,617	347,751	1,655	14.0	3.2	9,589	-1.8	3.7	0.3	669	-1.8	0.0	4.2	0.3	1,382	-1.8	0.0
4	360,617	347,751	580	14.0	3.2	16,350	-0.9	3.9	0.3	1465	-0.8	0.1	4.2	0.3	1,016	-0.8	0.1
5	360,617	347,751	489	14.0	3.2	7,822	-0.8	4.0	0.3	218	-0.5	0.3	3.8	0.3	1,798	-0.8	0.0
6	360,617	347,751	407	14.0	3.2	5,668	-0.9	4.1	0.4	417	-0.8	0.0	4.2	0.3	727	-0.8	0.0
7	360,617	347,751	353	14.0	3.2	4,290	-0.5	4.0	0.3	493	-0.5	0.0	4.3	0.4	476	-0.5	0.0
8	360,617	347,751	1,856	14.0	3.2	20,000	-0.8	3.7	0.3	7679	-0.8	0.0	4.3	0.3	20,000	-0.9	-0.1
9	360,617	347,751	516	14.0	3.2	20,000	-1.0	4.0	0.3	222	-1.0	0.0	4.0	0.3	748	-1.0	0.0
10	360,617	347,751	372	14.0	3.2	5,876	-0.9	3.8	0.3	1062	-0.8	0.0	3.9	0.3	994	-0.9	0.0
Average	360,617	347,751	808	14.0	3.2	10,543	-1.0	3.9	0.3	1399	-1.0	0.1	4.1	0.3	2,862	-1.0	0.0

Table B.4 Dataset S4: 2 jobs, 25 employees

Instance	Exact-Mono			Exact-Multi				MILP-UNet					MILP-UNet*				
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi
1	373,321	352,636	78	18.0	4.0	2,323	-8.8	3.7	0.3	137	-8.5	0.3	4.4	0.2	195	-8.8	0.0
2	373,321	352,636	74	18.0	4.0	6,942	-8.9	3.7	0.2	107	-7.9	1.1	4.1	0.2	156	-8.9	0.0
3	373,321	352,636	79	18.0	4.0	3,508	-6.4	3.4	0.2	85	-5.8	0.7	3.5	0.2	133	-6.4	0.0
4	373,321	352,636	78	18.0	4.0	4,860	-5.1	3.3	0.2	116	-4.0	1.1	3.9	0.2	66	-5.1	0.0
5	373,321	352,636	78	18.0	4.0	2,011	-7.1	3.6	0.2	133	-6.1	1.1	4.4	0.2	78	-7.1	0.0
6	373,321	352,636	81	18.0	4.0	2,639	-7.8	3.7	0.2	123	-7.5	0.3	4.4	0.2	98	-7.8	0.0
7	373,321	352,636	82	18.0	4.0	2,790	-7.6	3.6	0.2	88	-6.9	0.7	4.3	0.2	94	-7.4	0.2
8	373,321	352,636	74	18.0	4.0	1,084	-8.7	3.4	0.2	106	-7.6	1.2	4.1	0.2	100	-8.4	0.3
9	373,321	352,636	79	18.0	4.0	5,275	-7.3	3.4	0.2	104	-5.5	1.9	4.2	0.2	96	-7.3	0.0
10	373,321	352,636	77	18.0	4.0	6,515	-7.7	3.4	0.2	228	-6.6	1.2	4.5	0.3	109	-7.7	0.0
Average	373,321	352,636	78	18.0	4.0	3,795	-7.5	3.3	0.2	123	-6.6	1.0	4.2	0.2	112	-7.5	0.1

Table B.5 Dataset S5: 2 jobs, 36 employees

Instance	Exact-Mono			Exact-Multi				MILP-UNet					MILP-UNet*				
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi	VRBP	VRAP	Time(s)	CD-Mono	CD-Multi
1	540,578	524,604	287	22.0	4.6	18,759	-5.9	9.8	0.6	672	-5.3	0.6	8.8	0.4	351	-5.8	0.1
2	539,345	523,294	961	22.0	4.7	20,000	-5.3	9.7	0.5	1,509	-6.2	-0.9	9.0	0.4	1,257	-6.4	-1.2
3	536,721	520,915	579	22.2	4.6	13,681	-6.0	10.0	0.6	234	-5.7	0.3	8.2	0.4	525	-5.9	0.0
4	531,345	515,756	910	22.4	4.6	20,000	64.4	9.9	0.6	1,337	-6.7	-43.2	8.6	0.4	1,070	-7.0	-43.4
5	534,118	518,263	1197	22.3	4.6	20,000	63.4	9.7	0.5	1,177	-5.6	-42.2	8.6	0.4	508	-5.9	-42.4
6	533,140	517,483	691	22.3	4.6	20,000	*	9.9	0.6	2,048	-5.8	*	8.6	0.4	1,529	-5.9	*
7	531,852	516,319	618	22.3	4.6	20,000	-5.2	9.8	0.6	1,291	-5.9	-0.8	9.1	0.4	1,603	-6.2	-1.1
8	536,343	520,607	587	22.2	4.6	20,000	*	10.1	0.6	1,294	-6.3	*	7.7	0.3	2,142	-6.6	*
9	536,110	520,290	427	22.2	4.6	20,000	-5.1	9.7	0.6	1,005	-5.5	-0.4	8.9	0.4	1,267	-5.8	-0.8
10	540,820	525,062	420	22.1	4.7	16,551	-6.2	9.9	0.6	654	-6.2	0.0	8.7	0.4	596	-6.2	0.0
Average	536,037	520,259	668	22.2	4.6	18,900	11.8	9.8	0.6	1,122	-5.9	-10.8	8.6	0.4	1,085	-6.2	-11.1

The detailed results for the large (out-of-distribution) datasets are reported in Tables B.6 to B.13. For these datasets, only the Exact-Mono method is considered as a baseline and compared with MILP-UNet. The meaning of the columns is provided in Section 6.5.5.

Table B.6 Dataset L1: 5 jobs, 85 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	1,201,444	1,166,770	1,010	6.8	0.4	427	-1.0
2	1,199,173	1,164,323	741	6.9	0.4	568	-1.6
3	1,194,645	1,160,456	2,462	6.8	0.4	377	-1.3
4	1,185,503	1,151,820	1,415	6.8	0.4	1067	-1.9
5	1,190,376	1,156,156	868	6.8	0.4	573	-1.3
6	1,188,420	1,154,549	927	6.9	0.4	584	-1.1
7	1,186,350	1,152,718	874	6.9	0.4	504	-0.9
8	1,193,811	1,159,769	1,008	7.0	0.5	694	-1.3
9	1,193,520	1,159,281	546	7.0	0.4	466	-1.3
10	1,201,552	1,167,390	1,597	6.9	0.4	432	-1.5
Average	1,193,360	1,159,209	1,867	6.9	0.4	569	-1.3

Table B.7 Dataset L2: 6 jobs, 95 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	1,338,194	1,286,007	2,956	8.2	0.4	604	-3.4
2	1,337,390	1,284,872	2,372	7.9	0.3	677	-3.9
3	1,335,224	1,283,408	4,609	8.2	0.4	2333	-4.3
4	1,330,702	1,279,447	5470	7.8	0.3	1190	-4.6
5	1,329,781	1,277,686	4,581	7.7	0.3	1772	-4.4
6	1,331,218	1,279,613	6,591	8.1	0.4	939	-3.8
7	1,328,783	1,277,277	1,706	8.0	0.3	1491	-3.9
8	1,332,427	1,280,534	4,142	8.0	0.3	1599	-4.4
9	1,332,935	1,280,939	5,468	8.0	0.3	1627	-4.3
10	1,336,663	1,284,669	3,053	8.4	0.4	1498	-3.5
Average	1,333,421	1,281,532	4,197	8.0	0.4	1373	-4.1

Table B.8 Dataset L3: 7 jobs, 128 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	1,848,170	1,797,594	6,419	6.2	0.3	3,466	-3.2
2	1,846,789	1,795,864	20,000	6.1	0.3	2,131	-3.6
3	1,843,271	1,793,098	20,000	6.0	0.2	5,271	-2.8
4	1,835,118	1,785,526	20,000	6.1	0.2	4,130	-3.4
5	1,834,597	1,784,154	9,999	5.9	0.2	1,269	-2.9
6	1,835,976	1,786,035	19,994	6.1	0.3	1,706	-3.5
7	1,832,649	1,782,817	20,000	6.1	0.2	1,602	-2.3
8	1,838,698	1,788,465	20,000	6.1	0.2	8,245	-3.4
9	1,839,420	1,789,079	20,000	6.0	0.2	8,439	-3.5
10	1,846,080	1,795,721	20,000	6.1	0.2	2,882	-2.2
Average	1,840,195	1,789,951	17,641	6.1	0.2	3,914	-3.1

Table B.9 Dataset L4: 8 jobs, 95 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	1,393,791	1,301,165	1,262	2.7	0.1	234	-6.5
2	1,393,427	1,300,754	1,515	2.4	0.1	176	-5.7
3	1,393,791	1,301,165	1,570	2.5	0.1	247	-4.8
4	1,393,413	1,300,738	1,110	2.5	0.1	216	-4.2
5	1,393,427	1,300,754	3,026	2.5	0.1	181	-4.2
6	1,393,063	1,300,342	2,304	2.5	0.1	83	-6.0
7	1,393,413	1,300,738	1,926	2.5	0.1	191	-5.4
8	1,393,035	1,300,311	1,756	2.5	0.1	81	-5.9
9	1,393,413	1,300,738	2,086	2.5	0.1	244	-4.0
10	1,393,063	1,300,342	2,030	2.7	0.1	270	-6.4
Average	1,393,474	1,300,789	1,858	2.5	0.1	192	-5.3

Table B.10 Dataset L5: 8 jobs, 119 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	1,783,894	1,719,394	6,801	6.1	0.2	1,365	-2.8
2	1,781,925	1,717,013	3,950	5.9	0.2	2,531	-3.4
3	1,777,473	1,713,495	3,099	5.9	0.2	1,076	-3.2
4	1,768,136	1,704,874	3,245	5.9	0.2	1,093	-3.9
5	1,769,071	1,704,816	9,111	5.7	0.2	1,039	-2.6
6	1,769,755	1,706,061	2,330	6.0	0.2	1,777	-3.0
7	1,766,164	1,702,594	4,638	5.9	0.2	2,500	-3.2
8	1,773,461	1,709,435	7,559	6.0	0.2	2,006	-3.4
9	1,781,672	1,717,218	1,315	5.8	0.2	2,999	-3.3
10	1,774,215	1,710,310	2,473	6.0	0.2	1,408	-3.2
Average	1,774,671	1,710,613	4,429	5.9	0.2	1,780	-3.2

Table B.11 Dataset L6: 9 jobs, 133 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	2,042,550	1,945,043	9,512	3.7	0.1	1,557	-4.1
2	2,043,563	1,946,210	4,399	3.7	0.1	1,007	-4.2
3	2,042,353	1,944,784	5,608	3.5	0.1	2,520	-1.9
4	2,042,643	1,945,127	7,075	3.5	0.1	763	-2.3
5	2,041,672	1,944,005	11,286	3.3	0.1	1,039	-1.9
6	2,042,571	1,945,060	4,919	3.6	0.1	1,688	-3.5
7	2,043,502	1,946,150	4,439	3.7	0.1	2,453	-5.1
8	2,042,296	1,944,739	3,832	3.6	0.1	1,325	-3.2
9	2,042,974	1,945,520	4,556	3.5	0.1	2,312	-1.9
10	2,043,593	1,946,055	9,411	3.8	0.2	3,716	-4.6
Average	2,042,711	1,945,212	6,675	3.6	0.1	1,838	-3.3

Table B.12 Dataset L7: 10 jobs, 165 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	2,459,613	2,342,197	20,000	4.0	0.1	1,859	-3.9
2	2,458,933	2,341,793	20,000	3.8	0.1	4,731	-5.8
3	2,456,798	2,339,430	20,000	3.9	0.1	7,731	-3.4
4	2,452,393	2,335,316	20,000	3.8	0.1	12,808	-2.1
5	2,450,985	2,333,738	20,000	3.5	0.1	20,000	0.6
6	2,452,856	2,335,759	20,000	3.9	0.1	4,435	-3.9
7	2,450,067	2,333,346	20,000	3.6	0.1	1,346	-3.8
8	2,453,924	2,336,704	20,000	3.9	0.1	2,827	-3.8
9	2,454,548	2,337,462	20,000	3.7	0.1	2,054	-3.2
10	2,457,872	2,340,561	20,000	4.0	0.1	2,461	-4.3
Average	2,454,925	2,337,631	20,000	3.8	0.1	6,025	-3.4

Table B.13 Dataset L8: 12 jobs, 190 employees

Instance	Exact-Mono			MILP-UNet			
	VBP	VAP	Time(s)	VRBP	VRAP	Time(s)	CD-Mono
1	2,785,223	2,654,317	20,000	4.5	0.1	2,542	-5.7
2	2,784,408	2,651,034	20,000	4.3	0.1	5,438	-6.0
3	2,782,218	2,649,401	20,000	4.3	0.1	3,473	-5.5
4	2,777,788	2,645,458	20,000	4.2	0.1	8,342	-5.0
5	2,777,216	2,644,541	20,000	4.0	0.1	13,143	-4.9
6	2,778,467	2,645,823	20,000	4.3	0.1	1,941	-5.4
7	2,776,129	2,643,328	20,000	4.1	0.1	3,074	-5.9
8	2,779,800	2,647,095	20,000	4.4	0.1	4,965	-6.1
9	2,780,269	2,647,818	20,000	4.1	0.1	1,959	-5.9
10	2,780,269	2,647,460	20,000	4.4	0.1	7,750	-5.9
Average	2,780,224	2,647,628	20,000	4.3	0.1	5,263	-5.6