Les Cahiers du GERAD ISSN: 0711–2440

Data mining-driven shift enumeration for accelerating the solution of large-scale personnel scheduling problems

F. Rastgar-Amini, D. Aloise, C. Contardo, G. Desaulniers

G–2023–61

December 2023

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Data mining-driven shift enumeration for accelerating the solution of large-scale personnel scheduling problems

Farin Rastgar-Amini [a, b]

Daniel Aloise [c, b]

Claudio Contardo [d, b]

Guy Desaulniers [a, b]

[a] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Qc), Canada, H3C 3A7*

[b] *GERAD, Montréal (Qc), Canada, H3T 1J4*

[c] *Department of Computer Engineering and Software Engineering, Polytechnique Montréal, Montréal (Qc), Canada, H3C 3A7*

[d] *Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Montréal (Qc), Canada, H3G 2W1*

farin.rastgar-amini@polymtl.ca
daniel-2.aloise@polymtl.ca
claudio.contardo@concordia.ca
guy.desaulniers@gerad.ca

December 2023
Les Cahiers du GERAD
G–2023–61

Abstract : This study addresses large-scale personnel scheduling problems in the service industry by combining mathematical programming with data mining techniques to enhance efficiency. The studied problem aims at efficiently scheduling skilled employees over a one-week planning horizon, minimizing costs while meeting diverse job demands. In service industries, shift planning is intricately tied to customer presence, leading to a multitude of potential shifts and to a difficult optimization problem that cannot be easily solved using a commercial mixed-integer programming solver. Nevertheless, these problems are categorized as recurrent problems, where distinct instances share common characteristics and solution structures that differ only in a few parameters over time. Consequently, we propose to use a data mining technique, namely, the k-nearest neighbors algorithm, to expedite the solution process while upholding solution quality. In particular, we suggest using schedules of past solutions to reduce the problem size. Specifically, for an upcoming instance, we identify similar historical instances and streamline the enumeration of shifts to align with the comparable historical instances' schedules. This approach allows us to effectively address the problem using a commercial solver within a reasonable timeframe while preserving solution quality. Significantly, our methodology offers decision-makers the flexibility to determine the extent to which they wish to scale down the problem. Our experiments, conducted on a total of 80 instances with up to 12 jobs and 190 employees, yield an average removal of 85.5% of decision variables. This resulted in a noteworthy average speedup factor of 15.5, with a marginal average cost increase of 1.2%.

Keywords : Personnel scheduling, heuristic, k-nearest neighbors

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., Rastgar et al., 2022). 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 an 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 would help 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., Özder et al., 2020; Van den Bergh et al., 2013). 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 Dantzig (1954), 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. We identify similar instances in historical data while identifying outliers. Consequently, we streamline the enumeration of shifts for upcoming instances to align with the schedules of comparable historical 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 (kNN) algorithm to find similar data points. The proposed data mining-driven optimization algorithms are 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 2 includes a literature review of the solution methodologies for personnel scheduling problems focusing on artificial intelligence techniques. In Section 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 4. Section 5 presents the experimental results. Finally, Section 6 concludes with a brief summary and discusses potential extensions.

2 Related works

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 Van den Bergh et al. (2013). For example, decomposition methods have been developed for large-scale personnel scheduling problems (see, e.g., Attia et al., 2019; Rekik et al., 2004; Restrepo et al., 2018). 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.

In recent years, the mathematical optimization community has been increasingly interested in using artificial intelligence techniques to improve MILP solver performance. Bengio et al. (2021) provides a survey of the contribution of machine learning (ML) to combinatorial optimization problems. ML methodologies prove effective in addressing extensive continuous and combinatorial problems by acquiring and leveraging problem structures (see, e.g., Shi et al., 2023), predicting optimal solutions through the analysis of data extracted from previously solved instances (see, e.g., Xavier et al., 2020; Lodi et al., 2020), reducing problem sizes (see, e.g., Xu et al., 2016; Rastgar et al., 2022), and enhancing the performance of established optimization algorithms (see, e.g., Xavier et al., 2020; Gasse et al., 2019; Lodi and Zarpellon, 2017; Morabit et al., 2021; Tang et al., 2020).

Data mining and knowledge extraction offer 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 an optimization problem, leading to more efficient search strategies and improved decision-making. However, the use of data mining and knowledge extraction in personnel scheduling problems, specifically in the context of the retail industry, is limited.

Li et al. (2012) 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 research aims 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.

Guastalla et al. (2022) 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 regu-

lations. 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 Guastalla et al. (2022) and Li et al. (2012) 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, Rastgar et al. (2022) 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 below does not treat new instances as entirely distinct scenarios. Instead of predicting future shifts, the goal is to identify similar shifts within historical data.

3 The flexible personnel scheduling problem in retail

In this section, first, the description of the studied flexible personnel scheduling problem is provided (Section 3.1). Following that, Section 3.2 introduces the notation utilized in the formulated mathematical model, which is subsequently detailed in Section 3.3.

3.1 Problem statement

This study revisits a problem previously addressed by Hassani et al. (2023) and Rastgar et al. (2022). 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 Hassani et al. (2023) and Rastgar et al. (2022) 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 $H = \{1, 2, \ldots, 7\}$ 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 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.

3.2 Notation

Tables 1 to ?? in this section present the notation used for sets, subsets, parameters, and decision variables in the mathematical model.

Table 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 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/inE
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 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 $h \in H$

3.3 Mathematical model

The mathematical model for the flexible personnel scheduling problem in this research is formulated similarly to the work of Rastgar et al. (2022) as the following integer program:

$$\min \quad \sum_{j\in J}\sum_{s\in S^j} c_s^A u_{s,j} + \sum_{e\in E}\sum_{q\in Q} c_q^E w_{e,q} + \sum_{j\in J}\sum_{p\in P}\sum_{k\in K} c_k^V v_{p,j,f} \tag{1a}$$

$$\text{s.t.} \quad \sum_{s\in S^p\cap S^j} u_{s,j} + \sum_{e\in E_p^j}\sum_{s\in S^e\cap S^p\cap S^j} z_{s,e,j} - \sum_{f\in F} v_{p,j,f} = d_p^j \qquad \forall p\in P, j\in J \tag{1b}$$

$$\sum_{j\in J^e}\sum_{s\in S^e\cap S^j\cap S^h} z_{s,e,j} + o_{e,h} = 1 \qquad \forall e\in E, h\in H \tag{1c}$$

$$\sum_{j\in J^e}\sum_{s\in S^e\cap S^j} g_s z_{s,e,j} - \sum_{q\in Q} w_{e,q} = 0 \qquad \forall e\in E \tag{1d}$$

$$\sum_{h\in H} o_{e,h} \geq n_e^O \qquad \forall e\in E \tag{1e}$$

$$\sum_{j\in J^e}\sum_{s\in S^e\cap S^j\cap S^{h+1}} a_s z_{s,e,j} + M o_{e,h+1} - \sum_{j\in J^e}\sum_{s\in S^e\cap S^j\cap S^h} (b_s + 1 + n^R) z_{s,e,j} \geq 0 \qquad \forall e\in E, h\in H\setminus\{7\} \tag{1f}$$

$$z_{s,e,j} \in \{0,1\} \qquad \forall j\in J, e\in E^j, s\in S^j\cap S^e \tag{1g}$$

$$u_{s,j} \geq 0, \text{ integer} \qquad \forall j\in J, s\in S^j \tag{1h}$$

$$v_{p,j,f} \in [0, m_f] \qquad \forall p\in P, j\in J, f\in F \tag{1i}$$

$$w_{e,q} \in [0, m_q] \qquad \forall e\in E, q\in Q \tag{1j}$$

$$o_{e,h} \in \{0,1\} \qquad \forall e\in E, h\in H \tag{1k}$$

The objective function (1a) is designed to minimize the total costs encompassing personnel and anonymous shift costs, as well as over-covering penalties. Constraints (1b) guarantee that job demands at each period are met through personalized or anonymous shifts, facilitating the computation for over-coverings. Constraints (1c) 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 (1d). Additionally, each employee e is granted a minimum of n_e^O days off, as stipulated by constraints (1e). The big-M constraints (1f) 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 (1g)–(1k) identify the variables' acceptable domains.

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.

In this section, we start by presenting an instance and its features to be considered for similarity analysis (Section 4.1). Next, we explain how we measure the similarity of the data points (Section 4.2). In Section 4.3, we describe the utilization of the kNN algorithm to find similar instances and outliers. Lastly, we describe the optimization heuristic pipeline (Section 4.4).

4.1 Instance representation

As detailed in Section 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 days $\times$ 24 hours/day $\times$ 4 periods/hour = 672 periods). In Figure 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.

For the similarity analysis elaborated in the following section, we assume that we have access to historical problem instances and their solutions for a set N of previous weeks and we construct a historical dataset $\mathcal{D}$ composed of the demand curves for each job over the course of each day of the history. Thus, the total number of historical data points is equal to $|J| \times |H| \times |N|$. Each data point $i = (j, h, n) \in \mathcal{D}$ corresponds to a particular job j and day h of a week n characterized by a set of features, namely, the demands of job j on day h in week n. To elaborate, let $\tilde{p}(h)$ represent the index of the first period in day h which is divided into L time periods. The features of a data point $i = (j, h, n)$ are given by a vector

$$X^{(i)} = \left(d^j_{\tilde{p}(h)}, d^j_{\tilde{p}(h)+1}, \ldots, d^j_{\tilde{p}(h)+L-1}\right),$$

specifying the demand for job j during each time period of day h in week n.

With each data point $i = (j, h, n) \in \mathcal{D}$, we also associate a set of shifts $Y^{(i)}$ that were used for job j on day h in the solution of week n. To be more specific, let $z^{*(i)}_{s,e,j}$ denote the value of the corresponding $z_{s,e,j}$ variable within the optimal solution of the problem containing sub-instance i. Then, we define $Y^{(i)} = \{s \in S^j \cap S^h \,|\, \exists\, e \in E^j \text{ such that } z^{*(i)}_{s,e,j} = 1\}$. Figure 2 depicts the optimal schedule of 17 employees assigned to perform the 2 jobs of the example presented in Figure 1. For job 1 and job 2, the numbers of selected shifts are 21 and 47, respectively. For the data point i associated with job 1 and day 1 in the corresponding week, the set of shifts $Y^{(i)} = \{(20, 35), (36, 56), (57, 84)\}$, where the shifts s are identified by their start and end periods (a_s, b_s).

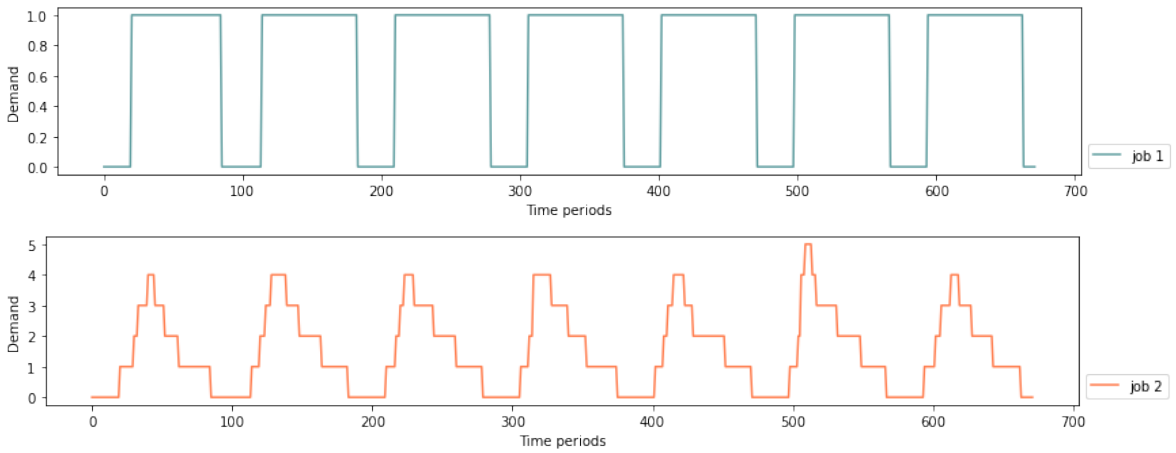

Figure 1: Demand curves of 2 example jobs

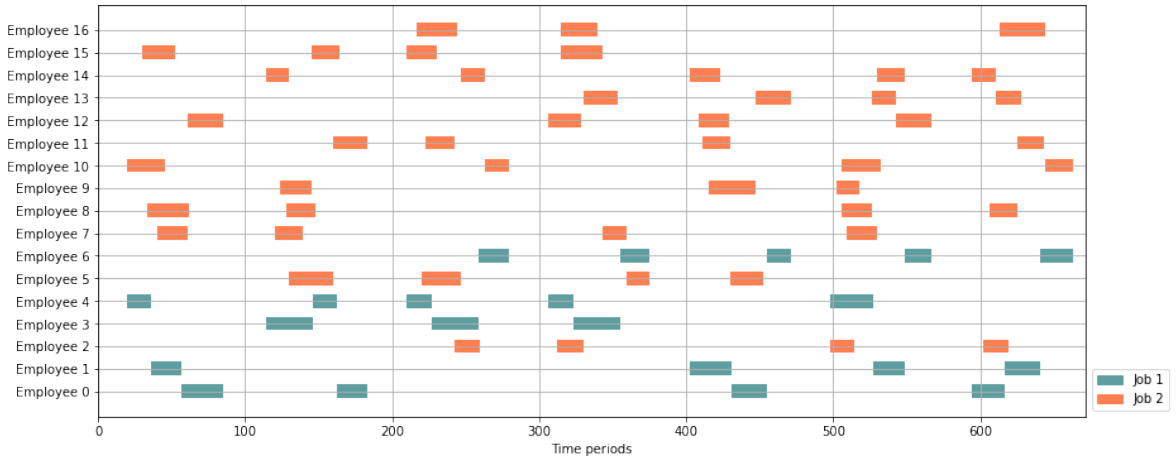

Figure 2: Employees' schedule assigned to jobs of example 1

4.2 Similarity 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 $\mathcal{D}$. A methodical way of quantifying similarity between data points is, thus, required.

Given two data points $X^{(i)}$ and $X^{(j)}$, their shape-based similarity can be determined by comparing local point distances. The most common distance function for quantitative data is the Euclidean distance (Aggarwal, 2015). Knowing that $X^{(i)}$ and $X^{(j)}$ are of the same length L, the Euclidean distance between them is defined as

$$Dist(X^{(i)}, X^{(j)}) = \sqrt{\sum_{p=1}^{L} \left(X_p^{(i)} - X_p^{(j)}\right)^2} \tag{2}$$

Our dataset presents two noteworthy aspects. Firstly, the data points exhibit varying scales, particularly between different job categories. 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 (Aggarwal, 2015).

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* stands out as a favored choice for this purpose due to its straightforward, intuitive characteristics and ease of application (Chan et al., 2003). 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.

4.3 K-Nearest Neighbors algorithm

In this section, we present our approach to finding similar instances using the kNN algorithm, which is a widely used supervised machine learning algorithm for classification and regression tasks (Cover and Hart, 1967). However, it can also be adapted for similarity search, making it a powerful tool for time series analysis. The kNN algorithm is a non-parametric and instance-based learning method. Given a dataset and a new data point i, kNN identifies the k nearest neighbors to i in the dataset based on a chosen distance metric. Additionally, kNN can identify data points that are significantly dissimilar from the dataset, effectively detecting outliers. The data scaling and transformation techniques applied during data pre-processing render the Euclidean distance metric a well-suited choice for utilization in the kNN 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 data points in $\mathcal{D}$ that are similar to the data points arising from this new problem instance, we apply the following steps:

1. Test data points: We define the set $\hat{\mathcal{D}}$ of test data points for this instance as explained in Section 4.1, generating $|J| \times |H|$ points from the decomposition of the problem into jobs and days of the planning horizon. These data points $(j, h, \hat{n})$ are then transformed and pre-processed in the same manner as the dataset $\mathcal{D}$.
2. Nearest neighbor search: For each test data point $(j, h, \hat{n}) \in \hat{\mathcal{D}}$, we calculate its distance from all points in $\mathcal{D}$ using the chosen distance metric and identify the set $\hat{\mathcal{K}}_{j,h}$ of its k nearest data points.
3. Outlier detection: For each data point, 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 its average distance from k nearest neighbors is much larger than the average distance of the other data points (Wang et al., 2020). More precisely, we compute the average distance of all data points $(j, h, \hat{n}) \in \hat{\mathcal{D}}$ from their k nearest data points in $\mathcal{D}$ as follows:

$$Dist^{avg}_{(j,h,\hat{n})} = \frac{1}{k}\Big(\sum_{(j',h') \in \hat{\mathcal{K}}_{j,h}} Dist(X^{(j,h,\hat{n})}, X^{(j',h')})\Big) \tag{3}$$

4.4 Optimization heuristic

The kNN algorithm provides a set $\hat{\mathcal{K}}_{j,h}$ of similar data points for each test data point $(j, h, \hat{n}) \in \hat{\mathcal{D}}$. Then, for each data point $(j, h, \hat{n})$ 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 $\hat{S}_{h,j} = \bigcup_{i \in \hat{\mathcal{K}}_{j,h}} Y^{(i)}$ 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-kNN:

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 kNN algorithm;
2. Build the MILP model (1a)–(1k) incorporating all possible shifts in $S^j \cap S^h$ for all outlier data points $(j, h, \hat{n})$ and only the shifts in $\hat{S}_{h,j} \cap S^j \cap S^h$ for the other data points $(j, h, \hat{n})$ (this restriction does not apply to the anonymous shifts);
3. Solve the resulting MILP model using a general-purpose MILP solver.

5 Experimental results

The proposed MILP-kNN heuristic is evaluated in different situations by means of computational experiments. It was programmed in Python 3 with libraries from scikit-learn (Pedregosa et al., 2011) for data scaling and kNN. Similarly, the code for loading the input data, running the kNN, 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 5.1) before explaining how we can tune the value of k for MILP-kNN (Section 5.2). Then, we report in Section 5.3 the results obtained by MILP-kNN.

5.1 Data generation

For testing our methodology, we only had access to a few real-life instances with almost no historical data. 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 from the demand curves of these real-life scenarios[1]. For each dataset, we generate randomly a large number of instances with the same number of jobs and employees (see Table 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 memic a stability of the operations in a retail store.

Table 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:

[1] All problem instances will be made available if the paper is published.

- Each job's working hours are distributed continuously throughout the day.
- Each demand curve exhibits a bell-shaped pattern similar to Figure 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 starts by creating one initial instance. For this instance, we randomly select a number of jobs between 5 and 15. Then, for each job, we select randomly from the real-life scenarios a job curve that 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 Porto et al. (2020), to generate 55 additional historical instances for each dataset. This simulation only perturbs the demand curves using a coefficient of variation of 1%. Out of the 56 historical instances generated for each dataset, 6 are also used for validation and parameter tuning in Section 5.2. For each generated instance, we construct the corresponding MILP model (1a)–(1k) 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 4.1, we decompose each historical instance and its solution per job and day to define data points $X^{(i)}$ in $\mathcal{D}$ and their sets of selected shifts $Y^{(i)}$. Finally, we further generate 10 additional test instances for evaluation purposes. For these instances, we also record their data points $X^{(i)}$, but they are not added to $\mathcal{D}$.

5.2 Tuning parameter k

The k parameter in kNN 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 ($\hat{S}_{h,j} = \bigcup_{i \in \mathcal{K}_{j,h}} Y^{(i)}$) 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 for the first two datasets. First, we solved these instances with an Exact algorithm that solves the MILP model (1a)–(1k), incorporating all possible shifts and a relative gap tolerance of 0.01%. Then, we solve these instances once again with the MILP-kNN 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.

Figure 3 illustrates 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 3a and 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 involve fewer decision variables, and the percentage of removed variables decreases by increasing the value of k, as anticipated.

Sub-figures 3c and 3d illustrate the impact of varying the value of k in the MILP-kNN 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

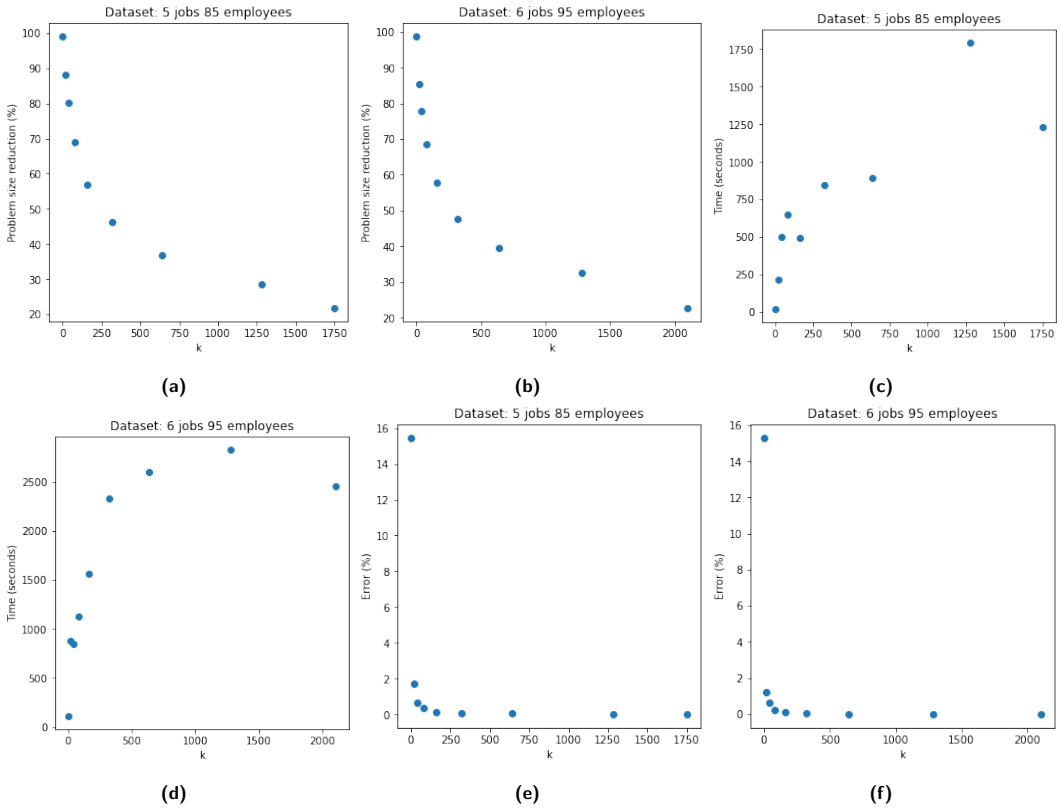

Figure 3: Impact of k on problem size, computational time, and solution quality

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 3e and 3f indicate the quality of the solutions produced by MILP-kNN by displaying the error in percentage on the y-axis. Notably, when $k = 1$, a large error occurs, whereas increasing the value of k results in a decreased error, eventually converging towards zero. 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-kNN 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 $e_v(k^v_{max}) \leq \tau$, where $k^v_{max} = |J| \times 7 \times 50$ is the maximal k value to consider for instance v. The algorithm starts by setting the search interval $[\underline{k}, \overline{k}] = [1, k^v_{max}]$ and iteratively halves it until $\overline{k} = \underline{k} + 1$, ensuring that $e_v(\underline{k}) > \tau$ and $e_v(\overline{k}) \leq \tau$ throughout the search. Each iteration proceeds as follows:

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

When $\overline{k} = \underline{k} + 1$, the algorithm stops and we define the optimal value of k for instance v as $k^*_v = \overline{k}$. When 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-kNN for the test instances associated with this dataset.

Figure 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-kNN requires 1176 seconds to yield a solution with an error of 0.94%.

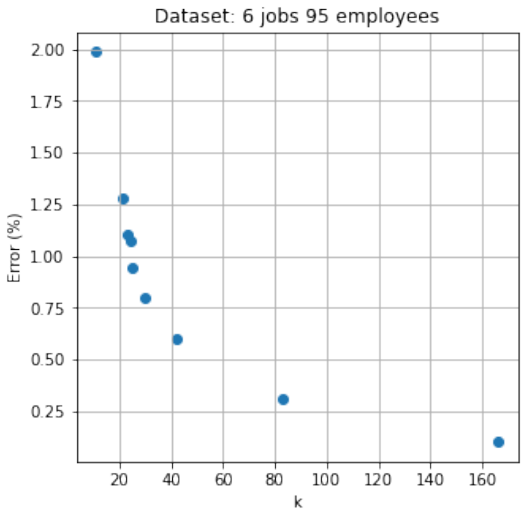

Figure 4: Search for the value of k for one validation instance in Dataset 2

5.3 Performance of MILP-kNN heuristic

The performance of the proposed MILP-kNN heuristic is evaluated in this section on 80 test problems (10 per dataset). Beside comparing MILP-kNN against the Exact algorithm, we compare it to the MILP-UNet* heuristic proposed by Rastgar et al. (2022). Like MILP-kNN, MILP-UNet* applies a pre-processing procedure to reduce the size of the MILP model (1a)–(1k). This pre-processing differs from the one that we propose 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 (1a)–(1k). 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-kNN 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 kNN in MILP-kNN are negligible, we compare these algorithms based only on the computational time of the MILP solver. For both MILP-UNet* and MILP-kNN, 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-kNN algorithms.

The results of our experiments are summarized in Tables 5 to 7 reflecting the average over 10 test problem instances for each dataset. Tables 8 to 15 in Appendix A report detailed results.

Table 5 reports statistics on the size of the MILP model (1a)–(1k) solved by the three algorithms (Exact, MILP-UNet*, and MILP-kNN). For the Exact algorithm which solves 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. Then, for MILP-kNN and MILP-UNet*, it 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 5.2 for parameter k).

From the results in Table 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-kNN significantly reduces the number of z-variables in each dataset, with removal percentages ranging from 77.7% to 96.8%. This highlights the effectiveness of the MILP-kNN method in simplifying the MILP model. While MILP-UNet* generally removes fewer z-variables than MILP-kNN, it still achieves substantial reductions with removal percentages ranging from 66.5% to 89.3%. On average, MILP-kNN and MILP-UNet* remove 85.5% and 78.6% of z-variables, respectively, showing a consistent and significant reduction across these datasets.

Table 5: MILP model size

Dataset	Exact			MILP-kNN		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 6 focuses on the solution quality obtained by the two heuristics on the test instances. For each heuristic and dataset, it presents the average error, measured as the difference between the costs of the solutions computed by the heuristic and 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.

Table 6: Error (in %)

Dataset	Average		Standard deviation		Minimum		Maximum	
	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*
1	**1.3**	2.5	**0.2**	1.2	0.9	**0.6**	**1.6**	4.6
2	1.3	**1.2**	**0.2**	0.7	0.8	**0.5**	**1.8**	4.5
3	**1.2**	1.7	**0.1**	1.4	-0.1	**-0.2**	**1.2**	8.6
4	**1.1**	1.6	**0.3**	0.4	**0.4**	0.8	**1.9**	2.4
5	1.3	**1.0**	**0.2**	0.4	0.9	**0.3**	**1.5**	2.4
6	**1.2**	1.9	**0.2**	0.4	**0.9**	1.2	**1.7**	3.0
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				

Table 7: Computational time (in seconds)

Dataset	Exact	Average		Standard deviation		Minimum		Maximum	
		MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*
1	1,867	**214**	272	**39**	113	**141**	171	**319**	817
2	4,197	**267**	498	**48**	171	**150**	230	**408**	941
3	17,641	1,221	**865**	**7**	414	1221	**367**	**1,234**	1,982
4	1,858	**228**	238	**50**	**50**	**143**	162	**319**	380
5	4,429	**133**	342	**25**	57	**90**	105	**198**	333
6	6,675	473	**342**	99	**58**	327	**150**	709	**478**
7	20,000	**922**	2,830	**376**	1,472	**352**	1,070	**2,038**	6,521
8	20,000	**1,503**	2,347	**598**	1,258	**303**	654	**2,660**	6,743
Avg.	9,584	**620**	967	**155**	449				

On average, MILP-UNet* yields a lower error (0.7%) compared to MILP-kNN (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-kNN, minimizing the risk of using bad predictions. On the other hand, MILP-kNN 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-kNN and MILP-UNet* show competitive performance, MILP-kNN 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.

Table 7 is similar to Table 6 but concerns the computational time. For each heuristic, it gives the average, standard deviation, minimum and maximum computational time in seconds 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-kNN and MILP-UNet* demonstrate significantly lower average computational times compared to the Exact algorithm, indicating the efficiency of the heuristics. On average, MILP-kNN 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). This suggests that MILP-kNN tends to be more computationally efficient on average across the datasets. Furthermore, MILP-kNN 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-kNN, this heuristic entails, in general, smaller minimum and maximum computational times. In summary, this table highlights the efficiency of MILP-kNN and MILP-UNet* in solving test instances within a reasonable computational time. MILP-kNN, 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 6).

Overall, MILP-kNN consistently outperforms MILP-UNet* in terms of computational efficiency. Although MILP-UNet* generally yields better average solution quality, as indicated by lower average and minimum errors across datasets, MILP-kNN shows more stability with lower standard deviations and maximum errors. Thus, the choice between MILP-kNN 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.

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 kNN algorithm, we dynamically adapt to historical instances, ensuring a focused search space for potential solutions. Our optimization heuristic, MILP-kNN, efficiently utilizes historical shifts to generate near-optimal personnel schedules.

The experimental results showcase the adaptability and effectiveness of the MILP-kNN heuristic across various datasets. By systematically exploring the impact of the kNN 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 kNN approach contributes to the efficiency and stability of MILP-kNN 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 kNN 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 coould 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.

A Detailed results

The results of our experiments are comprehensively presented for the test instances in each dataset, spanning Tables 8 through 15. In the *Exact* columns, we indicate the number of z-variables and other variables in the complete MILP model (1a)–(1k). The *z-Removed* columns offer a comparative analysis, illustrating the percentage of z-variables removed by MILP-kNN 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 method, MILP-kNN, and MILP-UNet* to solve the given instances. Bold results are the best ones.

Table 8: Dataset 1: 5 jobs, 85 employees

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

Table 9: Dataset 2: 6 jobs, 95 employees

Inst.	Exact		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Exact	MILP-kNN	MILP-UNet*
1	1258007	80187	82.3	**65.8**	1.5	**0.6**	2956	210	**230**
2	1257270	80120	82.5	**65.9**	1.6	**0.8**	2372	**150**	762
3	1255285	79939	82.9	**65.8**	1.2	**0.9**	4609	**254**	527
4	1251144	79558	82.7	**66.4**	1.2	**1.1**	5470	**273**	941
5	1250145	79636	83.2	**69.3**	**1.4**	4.5	4581	408	**371**
6	1251564	79654	83.0	**66.8**	**0.8**	1.1	6591	**213**	431
7	1249268	79515	82.9	**66.6**	1.0	**0.7**	1706	**291**	315
8	1252600	79827	82.9	**66.5**	1.8	**0.9**	4142	**281**	580
9	1253100	79835	83.0	**66.8**	1.4	**0.9**	5468	**285**	538
10	1256527	80136	82.8	**65.4**	1.3	**0.5**	3053	306	**288**
Avg.	1253580	79841	82.8	**66.5**	1.3	**1.2**	4197	**267**	498

Table 10: Dataset 3: 7 jobs, 128 employees

Inst.	Exact		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Exact	MILP-kNN	MILP-UNet*
1	1755479	92691	82.4	**84.1**	1.7	**1.0**	6419	**752**	785
2	1754165	92624	82.4	**85.7**	**1.2**	8.6	20000	867	**568**
3	1750828	92443	82.5	**84.6**	1.6	**0.9**	20000	1193	**367**
4	1743056	92062	82.5	**84.8**	-0.1	**-0.2**	20000	2303	**1809**
5	1742457	92140	82.9	**84.7**	2.1	**1.4**	9999	1084	**612**
6	1743818	92158	82.7	**84.6**	**1.2**	1.7	19994	**795**	877
7	1740630	92019	82.8	**84.5**	1.3	**0.9**	20000	**502**	457
8	1746367	92331	82.8	**84.4**	0.8	**0.5**	20000	2102	**729**
9	1747081	92339	82.5	**84.4**	1.7	**1.3**	20000	**1379**	1982
10	1753440	92640	82.7	**84.2**	1.0	**0.8**	20000	1234	**466**
Avg.	1747850	92345	82.6	**84.6**	**1.2**	1.7	17641	1221	**865**

Table 11: Dataset 4: 8 jobs, 95 employees

Inst.	Exact		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Exact	MILP-kNN	MILP-UNet*
1	1290300	103491	**80.3**	71.3	**0.4**	1.3	1262	**251**	319
2	1289950	103477	**80.7**	71.9	1.4	**1.2**	1515	**143**	256
3	1290300	103491	**81.0**	71.9	**0.5**	2.4	1570	**191**	223
4	1289936	103477	**81.4**	71.5	**1.9**	2.1	1110	**189**	236
5	1289950	103477	**81.3**	71.9	**1.1**	2.3	3026	211	**162**
6	1289600	103463	**80.7**	71.5	**1.1**	1.6	2304	275	**213**
7	1289936	103477	**81.0**	71.7	**0.8**	**0.8**	1926	317	**178**
8	1289572	103463	**81.0**	71.7	**1.2**	1.4	1756	185	**164**
9	1289936	103477	**80.9**	72.0	**1.2**	1.4	2086	319	**246**
10	1289600	103463	**80.1**	71.0	**0.9**	1.0	2030	**197**	380
Avg.	1289998	103476	**80.8**	71.6	**1.1**	1.6	1858	**228**	238

Table 12: Dataset 5: 8 jobs, 119 employees

Inst.	Exact		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Exact	MILP-kNN	MILP-UNet*
1	1678538	105356	**86.4**	77.6	0.9	**0.4**	6801	121	**109**
2	1676636	105289	**86.6**	77.7	1.5	**1.3**	3950	**116**	136
3	1672365	105108	**86.5**	77.9	1.3	**0.3**	3099	**135**	159
4	1663409	104727	**86.7**	78.3	1.3	**0.8**	3245	109	**105**
5	1664266	104805	**87.0**	78.4	**1.2**	**1.2**	9111	**90**	112
6	1664932	104823	**86.6**	78.6	**1.5**	2.4	2330	**198**	237
7	1661480	104684	**86.8**	78.2	1.0	**0.9**	4638	191	**121**
8	1668465	104996	**87.1**	78.0	1.5	**1.0**	7559	**115**	333
9	1676668	105004	**86.5**	77.9	1.4	**0.6**	1315	**133**	208
10	1668910	105305	**86.9**	77.5	1.1	**0.7**	2473	119	**114**
Avg.	1669661	105010	**86.7**	78.0	1.3	**1.0**	4429	**133**	163

Table 13: Dataset 6: 9 jobs, 133 employees

Inst.	Exact		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Exact	MILP-kNN	MILP-UNet*
1	1924558	117992	**79.2**	78.0	**1.3**	1.4	9512	**425**	434
2	1925538	118025	**79.0**	78.6	**0.9**	1.7	4399	401	**316**
3	1924368	117985	**79.1**	78.2	**1.3**	2.0	5608	618	**348**
4	1924648	117995	**79.1**	78.3	**0.9**	2.4	7075	709	**351**
5	1923710	117962	**79.0**	77.9	**1.5**	3.0	11286	426	**278**
6	1924578	117993	**79.2**	78.2	**0.9**	1.2	4919	**327**	387
7	1925478	118024	**79.2**	78.1	**1.2**	1.8	4439	406	**345**
8	1924314	117982	**79.2**	78.1	**0.9**	2.0	3832	360	**332**
9	1924970	118004	79.1	**79.9**	**1.7**	2.0	4556	510	**150**
10	1925568	118025	**78.9**	77.7	**1.4**	1.5	9411	552	**478**
Avg.	1924712	117999	**79.1**	78.3	**1.2**	1.9	6675	473	**342**

Table 14: Dataset 7: 10 jobs, 165 employees

Inst.	Exact		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Exact	MILP-kNN	MILP-UNet*
1	2327892	131721	**94.5**	88.3	1.2	**-2.2**	20000	**1161**	2726
2	2327279	131654	**94.6**	88.8	0.2	**-2.7**	20000	**352**	6521
3	2325325	131473	**94.8**	88.7	1.6	**-1.3**	20000	**521**	2507
4	2321301	131092	**94.9**	88.6	1.1	**-2.1**	20000	**455**	4227
5	2319815	131170	**94.8**	88.5	0.9	**-0.5**	20000	**1081**	5102
6	2321668	131188	**94.7**	88.7	-0.2	**-1.8**	20000	**1052**	1785
7	2319018	131049	**94.8**	88.6	1.0	**-1.3**	20000	2038	**1599**
8	2322563	131361	**94.9**	88.4	1.6	**-1.8**	20000	1155	**1222**
9	2323179	131369	**94.9**	88.3	0.5	**-0.7**	20000	**613**	1070
10	2326202	131670	**94.8**	88.3	0.7	**-0.8**	20000	**791**	1543
Avg.	2323550	131375	**94.8**	88.5	0.9	**-1.5**	20000	**922**	2830

Table 15: Dataset 8: 12 jobs, 190 employees

Inst.	Exact		z-Removed (in %)		Error (in %)		Time (in seconds)		
	z-Variables	Other Variables	MILP-kNN	MILP-UNet*	MILP-kNN	MILP-UNet*	Exact	MILP-kNN	MILP-UNet*
1	2626969	158254	**96.7**	89.3	0.5	**-3.4**	20000	1809	**654**
2	2626221	158187	**96.8**	89.4	2.8	**-3.4**	20000	**1022**	2507
3	2624212	158006	**96.8**	89.3	1.1	**-3.1**	20000	**1643**	2492
4	2620163	157625	**96.9**	89.5	1.4	**-2.8**	20000	**1197**	1488
5	2619513	157703	**96.8**	89.4	2.1	**-3.0**	20000	**1421**	6743
6	2620746	157721	**96.8**	89.2	1.6	**-2.6**	20000	**580**	1732
7	2618547	157582	**96.8**	89.3	1.3	**-2.2**	20000	2660	**847**
8	2621906	157894	**96.8**	89.0	1.8	**-3.6**	20000	**1840**	2523
9	2622367	157902	**96.8**	88.8	0.4	**-2.2**	20000	**303**	3761
10	2625681	154588	**96.8**	89.0	1.0	**-3.4**	20000	2555	**721**
Avg.	2622678	157546	**96.8**	89.2	1.4	**-3.0**	20000	**1503**	2347

References

Charu C. Aggarwal. 2015. Data Mining: The Textbook. Springer Publishing Company, Incorporated.

Dalia Attia, Reinhard Bürgy, Guy Desaulniers, and François Soumis. 2019. A decomposition-based heuristic for large employee scheduling problems with inter-department transfers. EURO Journal on Computational Optimization 7, 4 (2019), 325–357.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research 290, 2 (2021), 405–421.

F.K.-P. Chan, A.W.-C. Fu, and C. Yu. 2003. Haar wavelets for efficient similarity search of time-series: with and without time warping. IEEE Transactions on Knowledge and Data Engineering 15, 3 (2003), 686–705.

T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13, 1 (1967), 21–27.

George B Dantzig. 1954. Letter to the editor—A comment on Edie's "Traffic delays at toll booths". Journal of the Operations Research Society of America 2, 3 (1954), 339–341.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. 2019. Exact combinatorial optimization with graph convolutional neural networks. Advances in Neural Information Processing Systems 32 (2019).

Alberto Guastalla, Emilio Sulis, Roberto Aringhieri, Stefano Branchi, Chiara Di Francescomarino, and Chiara Ghidini. 2022. Workshift scheduling using optimization and process mining techniques: An application in healthcare. In 2022 Winter Simulation Conference (WSC). IEEE, 1116–1127.

Rachid Hassani, Guy Desaulniers, and Issmail El Hallaoui. 2023. A parallel ruin and recreate heuristic for personnel scheduling in a flexible working environment. Forthcoming in: Journal of Scheduling (2023), 1–18. https://doi.org/10.1007/s10951-023-00794-6

Jingpeng Li, Edmund K Burke, and Rong Qu. 2012. A pattern recognition based intelligent search method and two assignment problem case studies. Applied Intelligence 36 (2012), 442–453.

Andrea Lodi, Luca Mossina, and Emmanuel Rachelson. 2020. Learning to handle parameter perturbations in combinatorial optimization: an application to facility location. EURO Journal on Transportation and Logistics 9, 4 (2020), 100023.

Andrea Lodi and Giulia Zarpellon. 2017. On learning and branching: a survey. TOP 25, 2 (2017), 207–236.

Morabit Morabit, Guy Desaulniers, and Andrea Lodi. 2021. Machine-learning-based column selection for column generation. Transportation Science 55, 4 (2021), 815–831.

Emir Hüseyin Özder, Evrencan Özcan, Tamer Eren, et al. 2020. A Systematic Literature Review for Personnel Scheduling Problems. International Journal of Information Technology & Decision Making 19, 6 (2020), 1695–1735.

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. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

Andrés Felipe Porto, César Augusto Henao, Héctor López-Ospina, Esneyder Rafael González, and Virginia I. González. 2020. Dataset for solving a hybrid flexibility strategy on personnel scheduling problem in the retail industry. Data in Brief 32 (2020), 106066.

Farin Rastgar, Claudio Contardo, Guy Desaulniers, and Maxime Gasse. 2022. Learning to enumerate shifts for large-scale flexible personnel scheduling problems. Les Cahiers du GERAD G–2022–29. HEC Montréal, Canada. 24 pages.

Monia Rekik, Jean-François Cordeau, and François Soumis. 2004. Using Benders decomposition to implicitly model tour scheduling. Annals of Operations Research 128, 1–4 (2004), 111–133.

María I Restrepo, Bernard Gendron, and Louis-Martin Rousseau. 2018. Combining Benders decomposition and column generation for multi-activity tour scheduling. Computers & Operations Research 93 (2018), 151–165.

Yandong Shi, Lixiang Lian, Yuanming Shi, Zixin Wang, Yong Zhou, Liqun Fu, Lin Bai, Jun Zhang, and Wei Zhang. 2023. Machine learning for large-scale optimization in 6g wireless networks. IEEE Communications Surveys & Tutorials (2023).

Yunhao Tang, Shipra Agrawal, and Yuri Faenza. 2020. Reinforcement learning for integer programming: Learning to cut. In International Conference on Machine Learning. PMLR, 9367–9376.

Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. 2013. Personnel scheduling: A literature review. European Journal of Operational Research 226, 3 (2013), 367–385.

Xiaochun Wang, Xiali Wang, and Mitch Wilkes. 2020. New Developments in Unsupervised Outlier Detection: Algorithms and Applications. Springer.

Álinson S Xavier, Feng Qiu, and Shabbir Ahmed. 2020. Learning to solve large-scale security-constrained unit commitment problems. INFORMS Journal on Computing 33, 2 (2020), 739–756.

Huan Xu, Constantine Caramanis, and Shie Mannor. 2016. Statistical optimization in high dimensions. Operations Research 64, 4 (2016), 958–979.