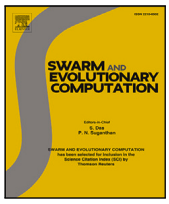
Swarm and Evolutionary Computation 99 (2025) 102160

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo



An evolutionary method with shift pattern learning for real-world multi-skilled personnel scheduling with flexible shifts

Ning Xue *, Ruibin Bai, Huan Jin, Tianxiang Cui

School of Computer Science, University of Nottingham Ningbo China, China

ARTICLE INFO

Keywords:
Irregular scheduling
Linear programming
Evolutionary algorithms
Memetic algorithm
Learning heuristic

ABSTRACT

Personnel scheduling remains a significant organizational challenge with substantial potential for cost and time savings. Despite extensive research in this domain, few studies have been successfully implemented in practice, and even fewer have gained widespread acceptance among end-users. This gap between research and application often arises from oversimplified real-world models, which may result from subjective solution evaluations or a lack of collaboration between modelers and end-users. To bridge this gap, this paper proposes a machine learning-enhanced memetic algorithm (MLMA) that mimics schedules created by experts to solve a highly complex personnel scheduling problem involving multi-skilled workers and flexible shift types (irregular workforce)—a real-world challenge commonly faced in the hospitality sector. By leveraging historical scheduling preferences, the MLMA generates solutions that align with past practices, enhancing their practicality and appeal to end-users. Experiments conducted on real-life instances demonstrate the effectiveness of the proposed approach in addressing real-world problems, where the workforce is predominantly part-time, possesses mixed skills, and requires flexible shifts. Furthermore, the results highlight the MLMA's ability to identify shift patterns that closely resemble historical schedules, underscoring its potential for practical implementation and its role in bridging the gap between research and real-world application.

1. Introduction

Irregular workforce scheduling involves assigning non-standard or on-call hours to employees, a practice common in industries such as agriculture, retail, and service sectors due to fluctuating demand. Unlike standard shift scheduling, the complexity increases significantly because part-time employees often have diverse availability, frequently juggling multiple jobs or managing personal responsibilities. Furthermore, the workforce is heterogeneous, requiring workers with different skills to meet varying staff demands, such as needing 1 manager, 2 chefs, and 1 counter staff during specific time slots (e.g., 8:00 AM to 8:30 AM). Personnel schedulers must also prioritize equity, individual worker preferences, and flexible hours to create a fair work environment and ensure employee satisfaction. These factors make it challenging to align the workforce with fluctuating demand and required skills while minimizing labor costs.

The complexity of irregular scheduling is further exacerbated when flexible shift types are introduced. Unlike rigid shift types, such as the morning, afternoon, and night shifts commonly used in nurse scheduling, flexible shifts exponentially increase the number of possible schedules. For example, consider a problem with 20 workers and 10 shift types. If each worker can be assigned any of the 10 shift types for each day of the week, the total number of possible schedules is $10^{20\times7} = 10^{140}$. In contrast, if the shift types are limited to 3 (e.g., morning, afternoon, and night), the number of possible schedules reduces to $3^{20\times7} = 3^{140}$. The larger base in the flexible shift scenario (10 vs. 3) results in a vastly larger solution space, making it computationally more demanding to explore and optimize.

Despite the considerable volume of academic research produced, the output of academic studies is often not embraced by end-users. In the context of personnel scheduling, a study by Kellogg and Walczak [1] highlights that only around 30% of the systems discussed in research articles are actually implemented, with minimal academic influence on the systems offered by third-party vendors. Van den Bergh et al. [2] found that out of 196 papers developed using real-world data, only 17% (46 papers) were applied in practice, while 76 papers relied on theoretical data. In the context of nurse rostering, the complexity of real-world problems often exceeds the capabilities of existing academic models. These models lack the depth necessary to capture the nuanced and diverse nature of practical scenarios, rendering them largely impractical for real-world application [3].

* Corresponding author.
E-mail addresses: ning.xue@nottingham.edu.cn (N. Xue), ruibin.bai@nottingham.edu.cn (R. Bai), huan.jin@nottingham.edu.cn (H. Jin), tianxiang.cui@nottingham.edu.cn (T. Cui).

https://doi.org/10.1016/j.swevo.2025.102160
Received 31 March 2025; Received in revised form 12 August 2025; Accepted 3 September 2025
Available online 13 September 2025

In real-world situations, what makes a solution high-quality can be subjective, causing differences in what is seen as optimal. This subjectivity, along with varying understandings of constraints and objectives between schedulers and modelers can make the automatically generated solution not meet real needs, rendering the optimization solution impractical in real-world use. With the widespread adoption of personnel scheduling software, historical scheduling and workers' attendance tracking records are now routinely and automatically saved. However, these valuable historical data are currently somewhat underutilized, primarily serving only for basic statistical analysis, reporting, and compliance tracking (e.g., auditing, payroll calculations, and workforce analytics). They have yet to be fully leveraged to enhance the efficiency of scheduling optimization algorithms or to improve the practicality and executability of scheduling outcomes. Given the periodic nature of demand, the preferences of both human schedulers and task executors, and the recurring patterns and fluctuations in scheduling demands across cycles, specific repeated trends and structures are often reflected in the scheduling solutions. Based on this observation, we believe that capturing these patterns and enabling optimization algorithms to utilize them could play a significant role in bridging the gap between automated scheduling theory and practical application. To this end, this paper makes the following contributions to improving user adoption of personnel scheduling solutions, with a particular focus on one of the most complex and demanding scenarios: scheduling with flexible shift types and a heterogeneous workforce:

1. To bridge the gap between research and application, this paper introduces a novel approach that leverages machine learning to learn from historical scheduling patterns. By extracting individual- and schedule-specific features, we classify shift patterns that align with end-user preferences. The trained machine learning model is integrated into our algorithm, guiding it towards solutions that better reflect human schedulers' preferences, enhancing practicality and adoption. We term this approach as Machine Learning-enhanced Memetic Algorithm (MLMA). This method addresses a common issue in real-life personnel scheduling, where models designed using Mixed Integer Programming (MIP) approaches or heuristic methods often fail to fully capture the true requirements of the problem. Unmodeled objectives or constraints, which are not explicitly included in the formulation but are reflected in historical schedules created by experts, can significantly impact the quality and practicality of the solutions. The MLMA leverages these historical patterns to bridge this gap, ensuring that the generated schedules align more closely with real-world needs.
2. A two-stage solution framework is proposed to effectively address the challenges of large-scale scheduling problems involving flexible shifts and the demand for a multi-skilled, heterogeneous workforce. This approach offers two key advantages: (1) it maintains computational tractability by avoiding prohibitively large integrated formulations for practical instances, and (2) it provides practical flexibility that aligns with real-world needs, allowing human schedulers to manually adjust designed shifts before final assignment, which is naturally supported by our decoupled approach. Experiments are conducted on real-life instances to evaluate the efficiency and effectiveness of the proposed two-stage framework.
3. An efficient Memetic Algorithm (MA) is designed to address soft constraints, such as balancing employee duty hours and accelerating the repair of constraint violations related to minimum and maximum working hours. This was difficult to resolve with heuristics in earlier studies, such as Ásgeirsson [4].

The remainder of this paper is structured as follows: Section 2 reviews relevant literature within the broader context of personnel scheduling. Section 3 provides a detailed problem description, followed by the methodology in Section 4. Sections 5 and 6 introduce the MA and the MLMA, respectively. Section 7 discusses the computational experiments and data instances. Finally, the paper concludes with a summary of key findings in the last section.

2. Literature review

Research in workforce scheduling has been classified into various types due to the diverse objectives and constraints involved, as seen in studies such as Defraeye and Van Nieuwenhuyse [5], Erhard et al. [6], and Çakırgil et al. [7]. The evolution of research on general workforce scheduling problems has advanced over the years, encompassing shift scheduling, days-off scheduling, and tour scheduling [8].

Research into irregular work scheduling is somewhat understudied compared to the numerous publications on other types of personnel scheduling. To grasp the essential characteristics of this problem, workforce scheduling can be broadly categorized into *fixed* and *flexible* scheduling based on shift decisions. The different types of shifts are defined as follows: *Fixed shifts:* These are predefined shifts where both the start times and the lengths of the shifts are fixed. Fixed shifts are commonly used in standard and regular workforce scheduling, such as in hospitals, manufacturing factories, and public services, where the demand is relatively stable and predictable. A typical example is the standard nurse rostering problem, where shift types might include morning shifts (6 am to 12 pm), afternoon shifts (12 pm to 6 pm), and night shifts (6 pm to 12 am). *Flexible shifts:* These shifts have definable start times and/or lengths. As previously mentioned, fluctuating demand is a key factor in irregular work scheduling. In such cases, the workforce can expand to meet increasing demands and revert to a baseline number when the additional workforce is no longer needed. Consequently, shifts often need to be overlapping and flexible to accommodate demand peaks without incurring the cost of overtime or hiring additional part-time staff. An example of this is personnel scheduling in a call center, where shifts can vary, such as 6 am–9 am, 6 am–9.30 am, 7 am–10 am, and so on. Compared to more regular shift scheduling, irregular shift scheduling results in a much larger set of shift patterns. The flexibility of shift decisions is a significant factor affecting the complexity and size of the problem, potentially leading to very large problems that are challenging to solve efficiently.

In both fixed and flexible workforce scheduling, the workforce can be either *homogeneous* or *heterogeneous* employees. A homogeneous workforce is characterized by a group of workers who possess the same skills and are capable of performing any assigned task. Generally, problems involving heterogeneous workforces are more complex to model and solve than those with homogeneous workforces.

Fig. 1 provides a possible classification of papers and the number of publications in each category since 2004. Our search for papers included those published since 2004, and some of the literature was sourced from and reproduced based on Van den Bergh et al. [2], which reviews publications up to 2004. The studies are categorized into seven groups by application area, including call centers, healthcare, manufacturing, retail, transportation, general, and other. A paper is classified as *general* when no specific application area is mentioned in the text and as *other* when it does not fit into any of the defined groups. The relevant bibliographies are presented in Tables 12 and 13 of Appendix.

The irregular personnel scheduling problem addressed in this study belongs to the category of flexible heterogeneous scheduling, as illustrated in Fig. 1. This category has been less explored compared to others. In terms of search space and complexity, it likely represents the most challenging personnel scheduling problem due to its vast search space.

The typical personnel scheduling process encompasses multiple stages. Tien and Kamiyama [9] broke down the problem into five distinct but interconnected stages, whereas Ernst et al. [10] categorized it into six phases. When the demand is known and time-specific (for

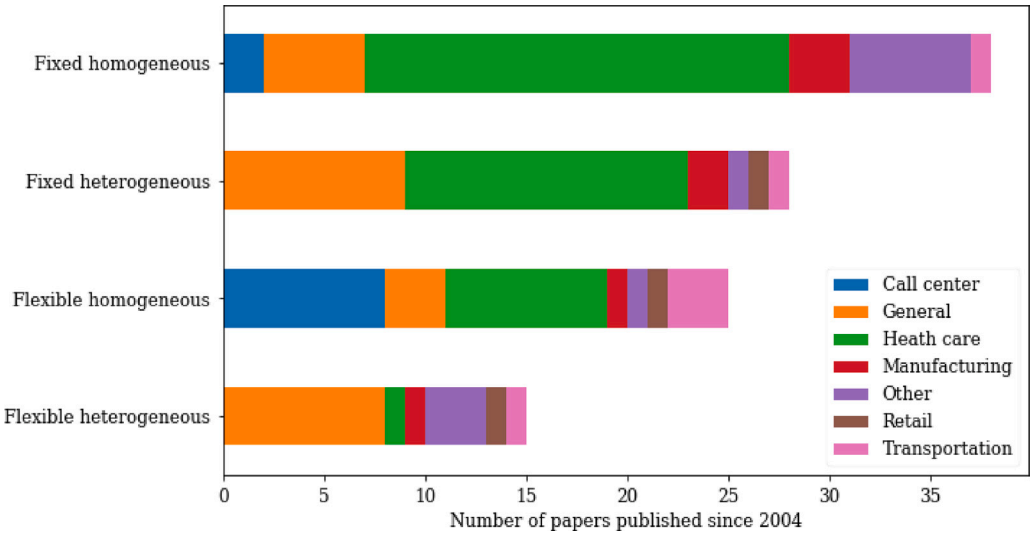


Fig. 1. A possible classification of personnel scheduling and the number of papers since 2004.

instance, three workers are needed on Tuesday from 7 am to 8 am), and worker assignment is shift-based (for example, worker A is scheduled for the morning shift from 7 am to 11 am), these stages can be broadly categorized into two main phases. The first stage, the shift designing involves designing a set of shifts and calculating their required workers to meet the demand. The second phase, shift assignment, focuses on allocating workers to each individual shift, while adhering to various constraints (such as availability, preference, and labor laws) to achieve certain objectives, such as minimizing labor costs and ensuring fairness in the distribution of workload.

Shift design and assignment can be addressed as a single phase by formulating a mathematical model and solving it with an appropriate method, such as using mixed-integer and linear programming solvers. However, the inclusion of flexible shift patterns can lead to an exceptionally large problem that is challenging to solve efficiently. To address this, exact methods of decomposition [11] and heuristic approaches [12] have been suggested to handle the integrated problem. Alternatively, shift design and shift assignment can be solved as two separate problems [13], which can simplify the complexity. However, an overall optimal solution for shift assignment, given an optimal solution for shift design, cannot be guaranteed.

As previously stated, this paper examines two-phase methods for the shift design and shift assignment problem. In this section, we offer a concise overview of the pertinent research conducted on these methodologies. Existing literature on shift design can be categorized into three main streams: papers that consider both shift design and shift assignment, papers with only shift design, and papers with only shift assignment.

2.1. Personnel scheduling with joint shift design and shift assignment

Gordon and Erkut [14] developed a three-stage approach for scheduling volunteers according to flexible shifts and preferences, with most shifts determined by personnel schedulers and some weekend shifts obtained through a set cover model. The shift assignment problem is formulated as an integer programming (IP) model. Topaloglu and Ozkarahan [15] presented a goal programming model that implicitly represents scheduling flexibility and incorporates information about employees' preferred working patterns. Zolfaghari et al. [13] formulated the problem as a large IP problem and proposed heuristics to reduce the number of shifts without solving the model exactly. Mirrazavi and Beringer [16] reported on the design and development of a workforce management solution system that respects operational and business requirements as well as employee contractual constraints. Bard and Wan [17] studied the scheduling of fixed full-time and flexible part-time workers, with full-time workers assigned to predefined shifts and part-time workers having flexible shift lengths and start times. A multi-stage solution approach was proposed. Previous work on this topic can be found in [18]. Hojati and Patil [19] studied the scheduling of heterogeneous part-time workers and proposed an IP formulation solved using heuristics. Omar et al. [20] considered a flexible shift pattern problem with shift durations ranging from 1 to 24 h and flexible starting times. Ganguly et al. [21] scheduling hospital physicians to shifts with three different durations and flexible starting times. Ásgeirsson and Sigurdhardóttir [22] introduced a mixed-integer programming (MIP) model to create staff schedules from a partial staff schedule based on employee requests and compared the results with a previously developed local search-based method [4], demonstrating the MIP solution method's suitability for several difficult instances.

2.2. Literature with solely shift design

Research on the standalone shift design problem is somewhat limited. Di Gaspero et al. [23] designed a hybrid heuristic algorithm that combines a constructive heuristic with a multi-neighborhood tabu search procedure, which demonstrates improved performance over a simpler local search approach proposed by Musliu et al. [24]. Kyngäs et al. [25] investigated the person-based multitask shift generation problem with breaks, developing a heuristic to create a shift structure separately for each day.

2.3. Literature with solely shift assignment

Gutjahr and Rauner [26] introduced an ant colony optimization (ACO) algorithm for solving a non-cyclic nurse rostering problem, which is characterized by the flexible setting of parameters such as nurses' and hospitals' preferences, the number of shift types, shift lengths, and overlaps, as well as the substitutability of nurses. Thiel [27] presented two dedicated optimization models: an extended rostering model and a roster combination model, tailored for crew scheduling problems. Smet et al. [28] developed a two-phase metaheuristic approach for the shift minimization personnel task scheduling problem (SMPTSP), focusing on assigning tasks to a pool of multi-skilled employees. Lin and Ying [29] proposed a three-phase algorithm to tackle the SMPTSP. Prot et al. [30] approached both the shift design and personnel task scheduling problems heuristically as separate entities. Brucker and Qu [31] suggested a network flow model for the task assignment problem within a heterogeneous workforce. Zamorano and Stolletz

[32] addressed a technician routing and scheduling problem where technicians, each with different skill sets, are paired into teams to carry out maintenance tasks. They proposed a mixed-integer programming (MIP) model and a branch-and-price algorithm as solutions to this problem.

In summary, the literature on workforce scheduling has significant progress in areas such as shift scheduling, days-off scheduling, and tour scheduling. However, several critical gaps remain. The flexible-heterogeneous scheduling problem, characterized by flexible shifts and a heterogeneous workforce, remains underexplored despite its unique challenges, such as a vast search space and high complexity. Current research in this area often relies on idealized models, limiting the utilization of real-world data and failing to capture practical nuances. Additionally, scalability issues persist for large-scale problems, and a disconnect between research and application often results in solutions that may not fully align with real-world needs.

3. Problem description

To accurately reflect real-world challenges, problem instances were carefully constructed using historical data from a local retail store, capturing the intricate complexities of workforce scheduling in a practical context. These instances, which form the foundation of this study, are publicly available and hosted at [33], which will be introduced in Section 7.1. Building on the identified gaps in the literature—such as the under exploration of flexible-heterogeneous scheduling, limited utilization of real-world data, and the disconnect between research and application—this work aims to address these issues through a practical and data-driven approach. Before introducing the problem in detail, we first clarify the terminologies used in this paper to ensure a clear and consistent understanding of the key concepts.

3.1. Shift, shift pattern, and schedule

We define the concepts of shift, shift pattern, and schedule. A *shift* is characterized by its date, start and end times, and roles. Shifts can range in length from 3 to 12 h. A *shift pattern* comprises a sequence of days worked (on days) and days off (off days) for a particular employee, with each working day consisting of a single assigned shift. A *schedule* is a collection of shift patterns generated by selecting a combination of shifts within the specified planning horizon. A schedule is designed for a workforce with multiple employees. In Fig. 2, we present an example of a schedule covering 3 days. Each row corresponds to a worker's shift pattern.

3.2. Characteristics of this problem

Irregular shift scheduling in the retail business presents unique challenges that distinguish it from regular workforce scheduling problems. Unlike traditional scheduling, it requires managing diverse worker types, accommodating the variable availability and preferences of part-time workers, and addressing varying shift times. Furthermore, personnel scheduling is inherently more complex than general scheduling problems, as it must balance cost efficiency and worker satisfaction while adhering to intricate shift rules and constraints. These challenges necessitate advanced techniques to develop effective and practical solutions.

More specifically, the inclusion of irregular shifts exhibits the following key characteristics:

- Workers are compensated differently according to their age and years of service.
- Workers are assigned different job roles (e.g. counter attendant, chef, cleaner).
- Each worker is capable of fulfilling different qualified roles during the same or different shifts.
- Workers may switch roles within the same shift.
- The availability and preferences of workers vary due to the prevalence of part-time contracts.
- The start and end times of shifts can differ across different workdays within a planning horizon.
- Different constraints may apply to different individuals.
- Rules for constructing shifts must adhere to the working regulations (e.g., constraints related to workers' qualifications, sequence constraints, and incompatible worker constraints).
- The objectives include minimizing labor costs while ensuring employee satisfaction by satisfying various constraints and ensuring the fairness of the shifts. More details regarding the objectives and constraints can be found in Section 3.3.

3.3. Objectives and constraints

A solution to the shift assignment problem involves assigning a shift pattern to each worker while meeting specific objectives and constraints. The objectives and constraints are summarized below.

3.3.1. Objectives

- **Primary Objective**: Ensure required staffing for each shift while minimizing labor costs. The cost of a shift is calculated as the shift length (in hours) multiplied by the worker's hourly pay rate.
- **Secondary Objective**: Fairly distribute *unpopular shifts* among workers. Unpopular shifts are defined as those starting before 12 pm or ending after 8 pm.
- **Tertiary Objective**: Ensure equitable distribution of total working hours among workers across the planning period.

3.3.2. Hard constraints

To ensure effective and harmonious workforce scheduling, the system incorporates several key constraints that address worker availability, qualifications, and interpersonal dynamics. The constraints include:

- **Worker Availability**:
  - Start and end times of availability within a working day.
  - Availability for duty on each day of the week.
- **Qualification Constraints**: Workers must possess the necessary skills (e.g., counter role, cleaner role) for assigned shifts.
- **Incompatible Worker Constraints**: Workers who do not get along or are too familiar with each other cannot be scheduled together to prevent conflicts or collusion.

3.3.3. Soft constraints

To ensure a fair and sustainable work environment, the scheduling process incorporates a set of critical constraints designed to protect employee well-being and maintain operational efficiency. These constraints are:

- **Working Hours**:
  - Minimum and maximum working hours per week.
  - Minimum and maximum working hours per day.
- **Working Days**:
  - Minimum and maximum working days per week.
  - Maximum consecutive working days allowed in the planning horizon.
- **Sequence Constraints**: For any worker, the time gap between the end of an assigned shift and the start of their next shift must not be less than the minimum rest period (e.g., 8 h in this study).

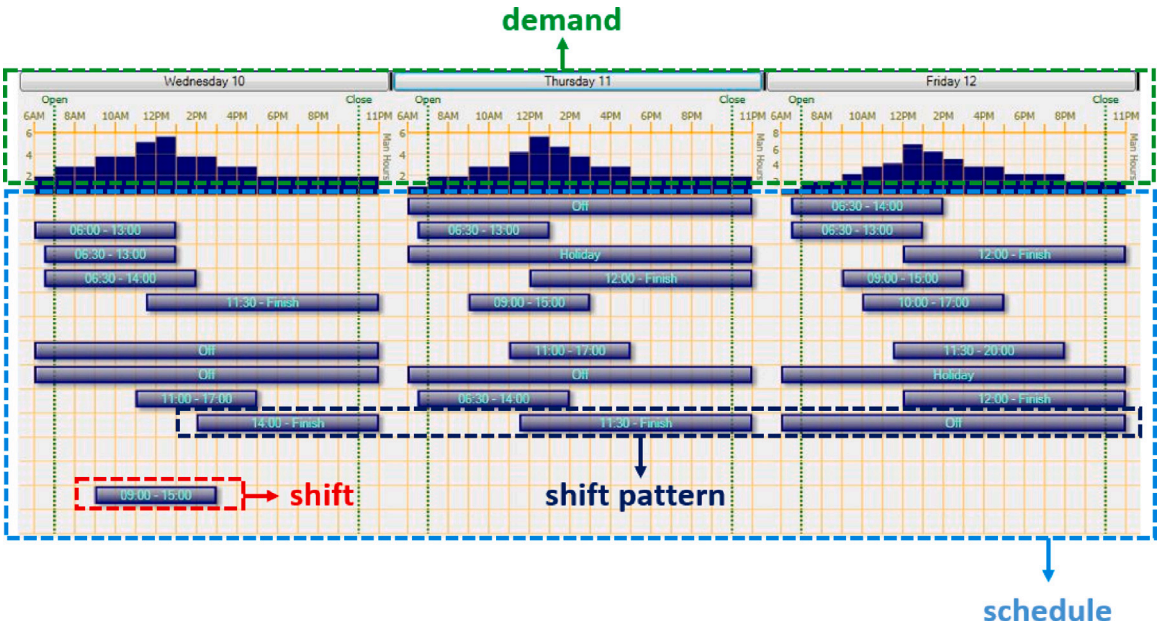


Fig. 2. An example of 3 days' schedule for a workforce.

4. Methodology

In this paper, we investigate a two-phase solution method to address the complexities of irregular personnel scheduling. The process begins with the *shift design* phase, where shifts are created to meet demand. This phase is formulated as a Mixed-Integer Linear Programming (MILP) model. The second phase, *shift assignment*, focuses on allocating the designed shifts to workers, ensuring availability, fairness and compliance with additional constraints such as working hours and preferences. This phase employs a memetic algorithm, building on our previous work [34]. By treating the two phases as independent problems, our approach allows for flexibility and intervention in the shift design output, which is particularly useful in real-world scenarios. Experiments on real-life instances demonstrate the effectiveness of this method for retail store scheduling, where the workforce is predominantly part-time, multi-skilled, and requires flexible shifts.

The demand data, which is characterized by four key attributes: day of the week, period of the day, required role, and the number of personnel needed. For example, a demand entry might specify (Monday, 8:00–8:30 AM, Cleaner, 2). This demand data serves as the input for the shift design phase, which outputs a set of shifts defined by the day of the week, start and end times, and the required role. For instance, a resulting shift could be (Monday, 8:00–11:30 AM, Cleaner). These shifts are subsequently assigned to workers in the shift assignment phase, adhering to the objectives and constraints outlined in Section 3.3.

The general pipeline of our solution process is illustrated in Fig. 3, which outlines the key stages of the proposed methodology, from data preprocessing to the final scheduling solution.

4.1. Machine learning for shift pattern prediction

As mentioned, to bridge the gap between research and application, this paper leverages machine learning to learn from historical scheduling patterns, enhancing the practicality and adoption of automated scheduling solutions. Recently, predicting optimal solutions for combinatorial optimization problems using machine learning (ML) has garnered significant attention. A series of studies, such as Sun et al. [35], Abbasi et al. [36], and Ding et al. [37], have shown that predicting the optimal solution values for individual variables can attain a reasonable level of accuracy. The predicted solutions can be employed in multiple ways: for instance, to narrow down the search space of a problem, to initialize a search method more effectively (warm-start), to choose an appropriate problem decomposition strategy, or to identify promising columns in the column generation approach. However, directly predicting a complete optimal solution for a problem proves to be rather challenging. This is primarily due to the NP-hard nature of such problems and the difficulty in devising a generalizable representation for solutions of varying sizes.

In this section, we aim to develop an efficient machine learning model capable of predicting shift patterns that are most appealing to end-users. This model is trained on historical scheduling data generated by experts. Following this, the trained model is used to guide the Memetic Algorithm (MA) towards a solution space that more closely aligns with the preferences of the experts. We refer to this approach as the Machine Learning-assisted Memetic Algorithm (MLMA).

In pursuit of refining heuristic methods, the integration of ML techniques has emerged as a promising approach [35]. By strategically designing appropriate features, we can significantly enhance the performance and efficiency of heuristics in solving complex computational problems. We can train an ML model using various learners capable of classification tasks, such as neural networks or decision trees. These models use the designed features as inputs to predict shift patterns that are appealing as outputs. These features are derived from historical scheduling data compiled by seasoned human schedulers, or experts. The resulting output is a set of shift patterns favored highly by end-users, including human schedulers and store managers.

The features are designed at two levels: individual-specific features and schedule-specific features. The former pertains to the unique aspects of shift patterns associated with each individual, while the latter evaluates the impact of an individual's shift patterns on the overall schedule composition.

4.1.1. Individual-specific features

We initially conducted feature engineering to create an extensive set of potential features that capture the characteristics of shift patterns. The criteria for selecting features are designed to ensure that they are interpretable and relevant to the problem's structure, including its objectives and constraints. Additionally, some features were inspired by insights provided by store managers. Following this, we prepared the data for the feature selection process. More specifically, we generated a series of schedule samples (i.e., solutions) using the MA method described in Section 5. The number of samples generated by the MA that

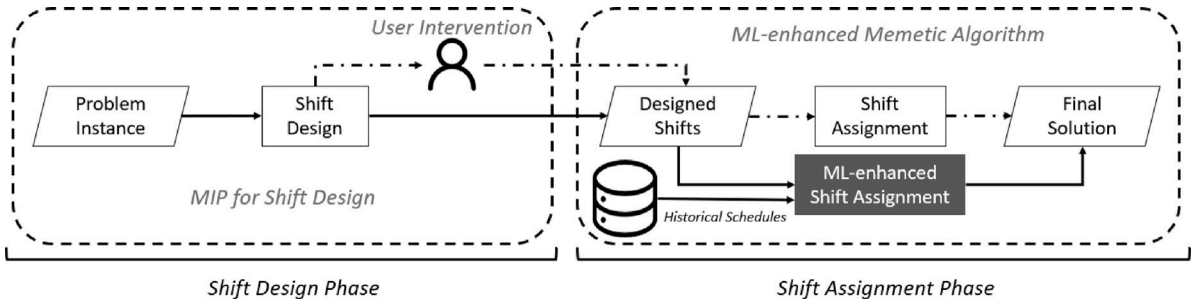
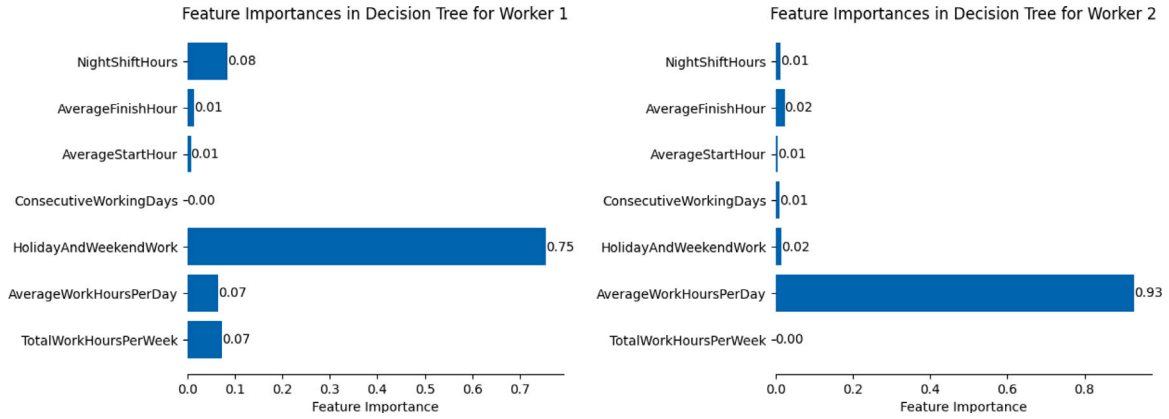


Fig. 3. Overview of the solution pipeline: From data preprocessing to schedule optimization.



(a) First worker's schedule (b) Second worker's schedule

Fig. 4. Feature importances for two workers.

included a particular worker corresponded to the number of historical schedules (created by experts) involving that worker. This ensured a balanced dataset for the machine learning classification model, where the labels were binary, indicating whether the data derived from historical records (labeled as *Appealing*) or from the MA-generated samples (labeled as *Not Appealing*). Note that the distribution of *Appealing* and *Not Appealing* samples may not be perfectly balanced, as some workers may not have been selected by the algorithm in the synthetically generated schedules. To maximize the utility of the dataset, we did not exclude any samples or apply downsampling to achieve perfect balance. Subsequently, the following seven features are identified as problem-specific attributes through the feature engineering process and are listed in Table 1, accompanied by their respective abbreviations.

As the shift patterns of individual workers can vary significantly, the importance of features (measured by Gini impurity) also differs. This variability is apparent in the accompanying figures (see Fig. 4), which show that the significance of features in the shift patterns of worker 1 and worker 2 is not consistent. This is also reflected in Figs. 5 and 6, which show the Decision Tree Analysis for individual-specific features to determine attractive shift patterns for Workers 3 and 4. Similarly, Figs. 5 and 6 showcase the Decision Tree analyses used to identify the appealing shift patterns for Workers 3 and 4, respectively, further highlighting this inconsistency.

Table 1
Summary of problem-specific features.

Feature	Abbreviation	Description
$f1$	NightShiftHours	Total number of hours worked during night shifts for a worker.
$f2$	AverageFinishHour	Average finishing hour for shifts within the shift pattern for a worker.
$f3$	AverageStartHour	Average starting hour for shifts within the shift pattern for a worker.
$f4$	ConsecutiveWorkingDays	Maximum number of consecutive days worked within a shift pattern for a worker.
$f5$	HolidayAndWeekdendWork	Total work hours during holidays and weekends for a worker.
$f6$	AverageWorkHoursPerDay	Daily average of work hours across all days of the shift pattern for a worker.
$f7$	TotalWorkHoursPerWeek	Sum of all work hours associated with each shift pattern for a worker.

4.1.2. Schedule-specific features

To measure how an individual shift pattern impacts the schedule, consider a worker i with a set of potential shift patterns, S^i. Each shift pattern in the set is represented as $s^i \in S^i$. A collection of schedules (i.e., solutions) that contain shift patterns for worker i, represented as $A = \{sol^i_1, sol^i_2, \ldots, sol^i_m\}$, is produced through the approach outlined in Section 5. Each sample is generated until the specified termination criteria are fulfilled and the number of samples generated by MA is the same as the number of historical samples on record for worker i, termed $H = \{sol^i_{m+1}, sol^i_{m+2}, \ldots, sol^i_n\}$. To create a balanced dataset for machine learning, the two sets are combined to form the training data, $A \cup H = \{sol^i_1, sol^i_2, \ldots, sol^i_n\}$. In this dataset, let $s^i_q = 1$ if and only if the shift pattern s^i appears in the qth sample.

Let $R = \{r^i_1, r^i_2, \ldots, r^i_n\}$ denote the ranking of n schedule samples. The ranking process involves sorting the patterns within sets A and H based on their computed cost. For each worker i, the cost is determined

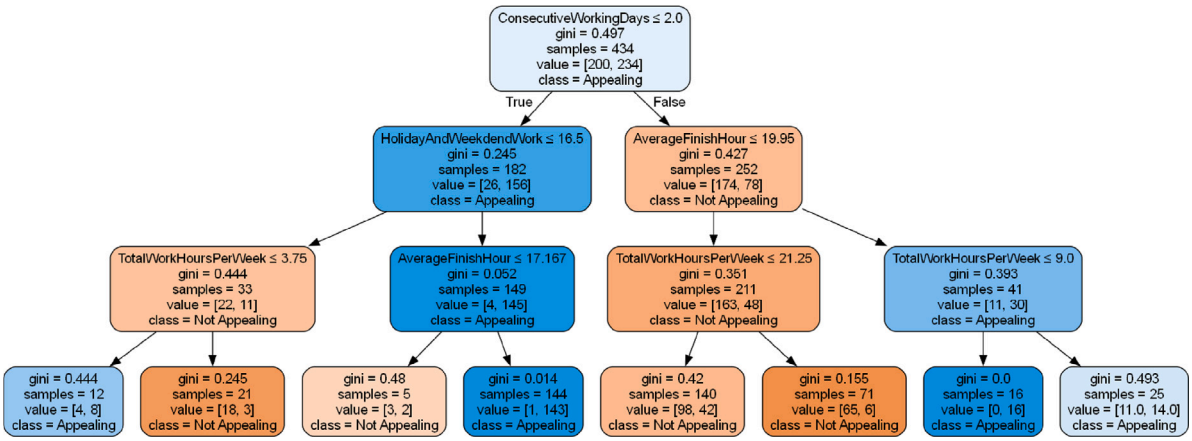


Fig. 5. Classifying shift patterns: Decision tree model for worker 3.

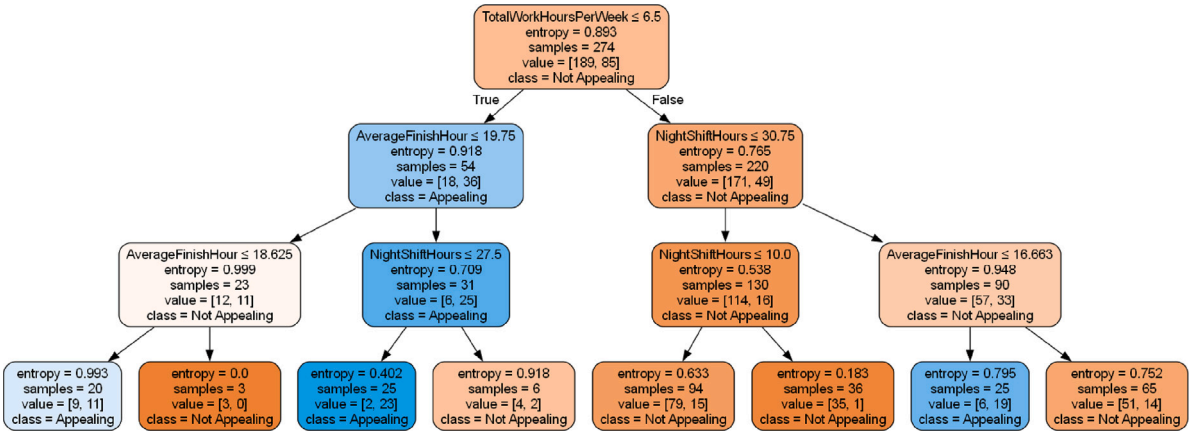


Fig. 6. Classifying shift patterns: Decision tree model for worker 4.

by the following formula:

$$obj_i = C^i + \sum_{n=1}^{8} W_n P_n^i \quad \text{for } n \neq 7$$

This formula is similar to the total objective but calculates the objectives contributed by an individual shift pattern for worker i. Note that penalty 7 is excluded from this calculation because it pertains to penalties associated with other shift patterns, not the one currently being evaluated.

Rankings from H and A are then merged, with historical data (with shift patterns labeled as *Appealing*) from H always assigned a lower rank than new samples from A. The statistical feature used, based on the ranking, is defined as

$$f_8 = \sum_{q=1}^{n} \frac{s_q^i}{r_q^i}$$

This score increases when a shift pattern s_q^i frequently appears in the historical data with a lower rank, suggesting a higher likelihood of a better ranking-based score. To ensure comparability, this feature is normalized by dividing by the maximum value of f_8 across all n items.

Another statistical feature is introduced to encapsulate the preference of human schedulers to minimize the number of workers needed to meet demand, a tendency observed in both historical data and through interviews with human schedulers. Whilst this preference is not directly formalized in our model, it is acknowledged as an implicit objective. We chose not to incorporate this as a specific objective or constraint for three main reasons: first, in practice, fulfilling demand with a reduced number of workers is not an absolute requirement but rather a flexible objective, as long as the number of workers is limited by the lower and upper bounds. Second, we aim to keep our model simple, given the numerous soft constraints and objectives already in place. Third, another reason for retaining this feature is to facilitate the design of our experiment, which will be explained in detail in Section 7.3 later. Generally, it helps to generate synthetic data that mimics a human scheduler's preference for using the fewest number of employees in the schedules as possible.

To translate this scheduler preference for fewer workers into our MLMA approach, we add a new statistical feature, f_9, defined as follows: as mentioned, the probability of a particular shift pattern being part of a historical schedule is positively correlated with the pattern's size (that is, the number of shifts it includes), since larger patterns cover more shifts. Hence, f_9 is calculated as

$$f_9 = \frac{|s^i|}{\max_{s'^i \in S^i} |s'^i|},$$

where $|s^i|$ represents the size of the shift pattern s^i.

4.1.3. Machine learning training and testing

We used various historical datasets from several years to build our training set. Due to staff turnover during this time, additional preprocessing was necessary. For the experimental analysis in Section 7.3, later in the paper, we generated synthetic instances to simulate expert-created schedules. This was done to validate our method, thereby bypassing the data cleaning step. The procedure for producing synthetic instances is described in detail in Section 7.3 as well.

Each shift pattern was treated as an individual training instance. The features and class labels for each shift pattern were determined

Table 2
Notations used in the problem formulation for the shift design and shift assignment phases.

Notation	Description
Decision variables	
$x^i_{(j,k)}$	1 if worker i is scheduled to perform shift (j,k) (shift type j on day k), 0 otherwise.
$y^r_{(j,k)}$	The number of shifts (j,k) (shift type j on day k) scheduled for role r.
Dependent decision variables	
z^i_k	1 if worker i is on duty on day k, 0 otherwise.
b_i	The number of unpopular shifts assigned to worker i throughout the entire schedule.
b_{avg}	The average number of unpopular shifts worked by each worker throughout the entire schedule.
t_i	The total working hours assigned to worker i throughout the entire schedule.
t_{avg}	The average working hours per worker throughout the entire schedule.
Data sets	
K	Set of consecutive days in a planning horizon.
J	A set of all possible shifts over the planning horizon, where each shift is uniquely identified by a combination of **shift type** j and **day** k. Formally, J is defined as a set of tuples (j,k), where j represents the shift type and $k \in K$ denotes the day in the planning horizon.
R	A collection of job roles, such as accountant, manager, and cleaner, where $r \in R$.
I	Set of workers scheduled in the planning horizon.
S'_I	Set of incompatible workers (e.g., {worker1, worker2}).
S_I	Set of incompatible sets S'_I of workers (e.g., {{worker1, worker2}, {worker3, worker4}}), where $S'_I \subset S_I, S_I \subset I$.
J^i_k	Set of worker i's available shifts on day k.
N^i_k	Set of shifts (j,k) for worker i where the shift length plus h_{rest} would overlap with the next day's shifts.
F^i_{k+1}	For worker i, shifts on day $k+1$ that cannot be assigned if worker worked any shift in N^i_k on day k.
Parameters	
Pb	Penalty cost due to the difference of one shift from b_{avg}.
Pt	Penalty cost due to the difference of one hour from t_{avg}.
W_n	Weight associated with the nth component of the penalty function P_n.
Data constants	
$\alpha_{t(j,k)}$	1 if period t is a work period covered by shift (j,k) (shift type j on day k), 0 otherwise.
m_k	Number of workers allowed to work on day k.
d_{rtk}	Staff demand of job role r at period t of day k.
$c^i_{(j,k)}$	Cost of worker i on shift (j,k) (shift type j on day k).
$l_{(j,k)}$	Shift length of shift (j,k) (shift type j on day k).
l^i_{min}	Allowed minimum working hours of worker i in a week.
l^i_{max}	Allowed maximum working hours of worker i in a week.
w^i_{min}	Allowed minimum number of days per week for worker i.
w^i_{max}	Allowed maximum number of days per week for worker i.
w^i_{cons}	Allowed consecutive working days per week for worker i.
h^i_{max}	Allowed daily maximum working hours for worker i.
h^i_{min}	Allowed daily minimum working hours for worker i.
h_{rest}	Minimum required rest hours between consecutive shifts.

as described earlier. After the training set was ready, we utilized a standard classification algorithm to identify the difference between *Appealing* and *Not Appealing* shift patterns. We opted for Decision Trees for this classification task due to their quick inference ability and high interpretability, even though we knew they might not provide the highest accuracy when contrasted with other algorithms such as Neural Networks, Naive Bayes, or Support Vector Machines. However, because the inference process needs to be compatible with our MA, reducing inference time was a key consideration. Additionally, the model's interpretability is crucial for explaining these methods to stakeholders in real-world applications. The quality score for a shift pattern is a probability between 0 and 1, calculated by determining the proportion of each class in the leaf node where the instance ends up after traversing the decision tree.

4.2. Shift design

Given the day's start time, end time, and shift increment, all feasible shifts are generated. The objective of this problem is to select a set of shifts from the feasible shifts, ensuring that the worker demand for each specified role r at each period t is met, while minimizing the total working hours. The length of period t is set to 0.5 h in this problem. The notations used in the shift design and shift assignment models are listed in Table 2.

4.2.1. Model

$$\min \sum_{r \in R} \sum_{(j,k) \in J} l_{(j,k)} y^r_{(j,k)} \tag{1}$$

subject to

$$\sum_{(j,k) \in J} y^r_{(j,k)} \alpha_{t(j,k)} \geq d_{rtk} \quad \forall r \in R, \forall k \in K, \forall t \in T \tag{2}$$

$$\sum_{r \in R} \sum_{(j,k) \in J} y^r_{(j,k)} \leq m_k \quad \forall k \in K \tag{3}$$

$$y^r_{(j,k)} \geq 0, \text{ integer} \quad \forall r \in R, \forall (j,k) \in J \tag{4}$$

Constraint (2) guarantees that the staff demand is met for each role and shift. Constraint (3) imposes a limit on the total number of workers that can be scheduled per day.

4.2.2. New demand constraints for the flexibility of demand coverage

Each worker possesses a specific set of qualified roles. For instance, a counter worker may be qualified to fulfill two roles: that of a counter attendant and a cleaner. The solution of the shift design model ensures that there are sufficient workers with the appropriate roles to meet the demand, as illustrated in Table 3.

Table 3
Roles and staff demands.

Week index	Day index	Period	Role ($r \in R$)	Staff demand (d_{rtk})
1	3	8:00–8:30	Manager (M)	n_m
1	3	8:00–8:30	Accountant (A)	n_a
1	3	8:00–8:30	Cleaner (C)	n_c

Table 4
Role combinations and derived staff demand requirements.

Role combination ($o \in O$)	Combination ($R_o \subset R$)	New staff demand (d_{otk})
1	M	n_m
2	A	n_a
3	C	n_c
4	$M \cup A$	$n_m + n_a$
5	$M \cup C$	$n_m + n_c$
6	$A \cup C$	$n_a + n_c$
7	$M \cup A \cup C$	$n_m + n_a + n_c$

As mentioned in the problem characteristics, for the flexibility of shift and role allocation, each worker is capable of fulfilling various qualified roles during the same shift. To meet this requirement, the minimum number of workers from each role who must be on duty in each period can be defined cumulatively. For instance, in Table 3, let $n_m = 1$, $n_a = 2$ and $n_c = 3$. Suppose worker i_1 is a manager but also qualified for role account and cleaner, Worker i_2 and i_3 qualified to take roles account and cleaner. If these three workers are assigned to the same shift, we will have 1 worker (i_1) take the manager role, 2 workers (i_1 and i_2) take the 2 accountant roles and 3 workers (i_1, i_2 and i_3) take the 3 cleaner roles. However, even through there are enough workers cover the specific roles but the demand requires 6 workers in total.

To ensure sufficient coverage for assigning workers with the correct roles, the initial demand definition (i.e., d_{rtk}) must be expanded. This expansion is necessary to account for all potential combinations of roles, ensuring adequate coverage by workers with the required qualifications. The generation of these new role combinations is illustrated in Table 4. In this table, n_m, n_a, and n_c represent the original demand (i.e., d_{rtk}) for specific roles, such as manager, accountant, and cleaner, respectively. It is important to note that these roles are provided as examples only; the formulation is not restricted to managers, accountants, or cleaners and can accommodate any set of roles. The new demand representation is labeled as d_{otk}.

In formulating the model to represent the new staffing demand, one approach employs a four-dimensional decision variable, denoted as $x^{ir}_{(j,k)}$. This expands upon the previously mentioned three-dimensional variable $x^{i}_{(j,k)}$ by incorporating an additional dimension for the job role r. This variable is directly linked to the demand variable d_{rtk}, which specifies the required number of staff for a given role r at time period t on day k. An alternative strategy to meet the new demand could involve reframing the demand as a breakdown of required skills in percentage terms. For instance, the demand could be represented as: management skills at 30%, accounting skills at 30%, and cleaning skills at 40%.

In this paper, we employ a three-dimensional decision variable $x^{i}_{(j,k)}$ instead of a four-dimensional variable $x^{ir}_{(j,k)}$ to address the new demand representation. This choice is motivated by two key considerations. First, using a higher-dimensional variable would significantly increase computational complexity during the optimization phase, making the problem more challenging to solve efficiently. Second, from a practical standpoint, human schedulers typically focus on assigning workers to shifts on specific days rather than micromanaging roles within each shift. This approach aligns with the realities of the hospitality sector, where employees often perform multiple roles, and role assignments can be dynamically adjusted as needed. Such flexibility is crucial for effective workforce planning, ensuring adaptability in response to operational demands. In fact, this can be modeled as a covering constraint, with the actual assignment of workers to roles efficiently resolved using a maximal matching algorithm on a bipartite graph. However, this aspect is beyond the scope of this paper.

Table 5
Additional notations in the problem formulation.

Additional notations	
O	A set of role combinations, $o \in O$.
d_{otk}	Represents the staffing requirement for role combination o during period t on day k.

The decision-making process revolves around assigning workers to specific shifts on given days, without detailing the exact role, responsibility, or skills to be utilized during that shift. This approach accommodates the dynamic nature of the hospitality industry by ensuring that there are enough workers with the necessary skills to meet each time period's demands. As long as these conditions are met, the demand is considered satisfied. For example, consider a restaurant that needs to staff its front desk, food preparation, and maintenance roles throughout the day. Instead of rigidly assigning each worker to a specific role for every shift, the model focuses on ensuring that there are enough workers with the appropriate skills available to cover each area of demand. This could mean that a worker typically assigned to maintenance might be tasked with covering the front desk if the demand requires it, again, illustrating the need for a flexible workforce scheduling strategy.

Following the explanation of demand requirements outlined above, we will now introduce additional notations (see Table 5) and the new constraint as a replacement for the original one (Constraints (2)), as detailed below:

$$\sum_{(j,k)\in J} y^{r}_{(j,k)} \alpha_{t(j,k)} \geq d_{otk} \quad \forall o \in O, \forall t \in T, \forall k \in K \tag{5}$$

Constraint (5) guarantees that staffing requirements are met and further restricts workers from being assigned to shifts for which they lack the necessary qualifications.

4.3. Shift assignment

The shift assignment problem aims to assign the shifts that have been determined by the model presented in Section 4.2 to each worker, ensuring that all constraints are met, and the objective is to minimize the total labor cost while balancing the workload among the employees.

As previously mentioned, the problem addressed in this paper involves flexible shifts characterized by variable lengths and start/end times. This flexibility distinguishes our problem from traditional nurse rostering problems, such as those studied in [38,39], and prevents the direct application of their techniques. Specifically, the high variability in workers' availability and the resulting large number of potential "shift patterns" [38] significantly increase the problem's complexity. Furthermore, while there are similarities in constraints between this problem and other personnel scheduling problems involving flexible working patterns (e.g., [4,12]), the heuristics developed for those problems are tailored to their specific structures and cannot be directly applied here. Consequently, the problem addressed in this paper cannot be effectively solved using "off-the-shelf" approaches from the existing literature.

In our earlier work [34], we proposed a genetic algorithm (GA) that employs an effective composite chromosome encoding scheme to solve a simplified version of the personnel scheduling problem with fewer constraints. Building on this foundation, this paper extends the previous study by applying evolutionary algorithms to address the more complex real-life personnel scheduling problem, incorporating additional constraints and practical considerations.

4.3.1. Constraint handling

There are three common ways to handle constraints in evolutionary algorithms: (1) Use a specialized encoding that restricts the search to feasible solutions, (2) Repair solutions that become infeasible due to genetic operators, and (3) Use penalties in the objective function to discourage infeasible solutions. Our scheduling problem involves constraints on worker hours, days worked, and compatibility. Since these constraints make repairing solutions difficult, we combine the first and third approaches. Mutation and crossover ensure that workers are not assigned multiple shifts in a day. Our encoding scheme also handles availability and skill constraints (i.e., ensuring enough workers with the required roles to cover demand). The remaining constraints are managed through penalties in the fitness function. The notations and fitness function relevant to this problem are provided below:

4.3.2. Objective and penalty functions

A high-level description of the objectives and constraints has been provided in Section 3.3. In this section, we present the mathematical formulation of the penalty terms.

4.3.2.1 The labor cost is calculated by:

$$C = \sum_{i\in I}\sum_{k\in K}\sum_{j\in J_k^i} c_{(j,k)}^i x_{(j,k)}^i \tag{6}$$

This represents the total wage expenditure, calculated by summing the cost of every assigned shift. Each term $c_{(j,k)}^i x_{(j,k)}^i$ becomes active (non-zero) only when worker i is scheduled for shift j on day k ($x_{(j,k)}^i = 1$).

4.3.2.2. The penalty for the assignment of unpopular shifts is given by:

$$P_1 = \sum_{i\in I} pb\left|b_i - b_{avg}\right| \tag{7}$$

This measures equity in undesirable shift assignments. For each worker, we calculate how much their count of unpopular shifts (b_i) deviates from the team average (b_{avg}), then multiply by penalty weight Pb.

4.3.2.3. The penalty for unfair assignment of working hours is given by:

$$P_2 = \sum_{i\in I} pt\left|t_i - t_{avg}\right| \tag{8}$$

This penalty ensures balanced workload distribution by penalizing deviations from average weekly hours. The penalty term scales the absolute difference between each worker's total hours (t_i) and the mean (t_{avg}) by a penalty factor Pt.

4.3.2.4. The penalty for exceeding the maximum number of working hours or falling short of the minimum number of working hours in a week is given by:

$$P_3 = \sum_{i\in I}\max\left\{0, \sum_{k\in K}\sum_{j\in J_k^i} x_{(j,k)}^i l_{(j,k)} - l_{max}^i\right\} + \sum_{i\in I}\max\left\{0, l_{min}^i - \sum_{k\in K}\sum_{j\in J_k^i} x_{(j,k)}^i l_{(j,k)}\right\} \tag{9}$$

The first term penalizes situations where a worker's weekly hours exceed their maximum allowed hours (l_{max}^i), while the second term activates when their hours fall below the minimum requirement (l_{min}^i).

4.3.2.5. The penalty for exceeding the maximum number of working hours or falling short of the minimum number of working hours in a day is given by:

$$P_4 = \sum_{k\in K}\sum_{i\in I}\max\left\{0, \sum_{j\in J_k^i} x_{(j,k)}^i l_{(j,k)} - h_{max}^i z_k^i\right\} + \sum_{k\in K}\sum_{i\in I}\max\left\{0, h_{min}^i z_k^i - \sum_{j\in J_k^i} x_{(j,k)}^i l_{(j,k)}\right\} \tag{10}$$

Like P_3, this penalty checks daily hours against limits (h_{max}^i, h_{min}^i), with z_k^i restricting evaluation to worker i's scheduled days.

4.3.2.6. The penalty for exceeding the maximum number of working days or falling short of the minimum number of working days in a week is given by:

$$P_5 = \sum_{i\in I}\max\left\{0, \sum_{k\in K} z_k^i - w_{max}^i\right\} + \sum_{i\in I}\max\left\{0, w_{min}^i - \sum_{k\in K} z_k^i\right\} \tag{11}$$

This penalty term regulates the number of working days per worker, preventing both under-scheduling (below w_{min}^i) and over-scheduling (exceeding w_{max}^i).

4.3.2.7. The penalty for exceeding the maximum number of consecutive working days in a week is given by:

$$P_6 = \sum_{i\in I}\max\left\{0, w_{cons}^i - \sum_{k=k'}^{k'+w_{cons}^i} z_k^i\right\}, \quad k' = 1, 2, \ldots, |K| - w_{cons}^i \tag{12}$$

This penalty term systematically evaluates the schedule using sliding windows to detect violations of the maximum consecutive working days constraint (w_{cons}^i).

4.3.2.8. The penalty for violating worker incompatibility is given by:

$$P_7 = \sum_{k\in K}\sum_{t\in T}\max\left\{0, \sum_{i\in S_I'}\sum_{j\in J_k^i} x_{(j,k)}^i \alpha_{t(j,k)} - 1\right\} \tag{13}$$

For each time period, this penalty term counts the number of forbidden pairings (from set S_I') that appear together in the schedule.

4.3.2.9. The penalty for insufficient rest between shifts is given by:

$$P_8 = \sum_{i\in I}\sum_{k=1}^{|K|-1}\max\left\{0, \sum_{j\in N_k^i} x_{(j,k)}^i + \sum_{j'\in F_{k+1}^i} x_{(j',k+1)}^i - 1\right\} \tag{14}$$

This penalty enforces minimum rest periods by penalizing assignments where worker i is scheduled for both: (1) any shift $j \in N_k^i$ on day k (providing less than h_{rest} hours of rest), and (2) any shift $j' \in F_{k+1}^i$ on day $k+1$ (starting before h_{rest} hours have elapsed). The violation sets are defined as $N_k^i = \{j \in J_k^i \mid \text{rest after } j < h_{rest}\}$ and $F_{k+1}^i = \{j' \in J_{k+1}^i \mid \text{start time} < \text{end time of } j \in N_k^i + h_{rest}\}$.

4.3.2.10. The overall objective function is to minimize the weighted sum of the above eight penalty functions:

$$obj = \min(C + \sum_{n=1}^{8} W_n P_n) \tag{15}$$

Algorithm 1 Pseudo-Code of local search

Require:
E_{above}: Set of employees with scheduled working hours above duty hours
E_{below}: Set of employees with scheduled working hours below duty hours
for $Pairs(e_a, e_b) : e_a \in E_{above}, e_b \in E_{below}$ **do**
  for $s \in e_a.shifts$ **do**
    if e_b can work on shift s and e_a can be removed from shift s **then**
      $e_a.shifts \leftarrow e_a.shifts \setminus \{s\}$
      $e_b.shifts \leftarrow e_b.shifts \cup \{s\}$
      if e_a is at or below duty hours **then**
        $E_{above} \leftarrow E_{above} \setminus \{e_a\}$
      end if
      if e_b is at or above duty hours **then**
        $E_{below} \leftarrow E_{below} \setminus \{e_b\}$
      end if
    end if
  end for
end for

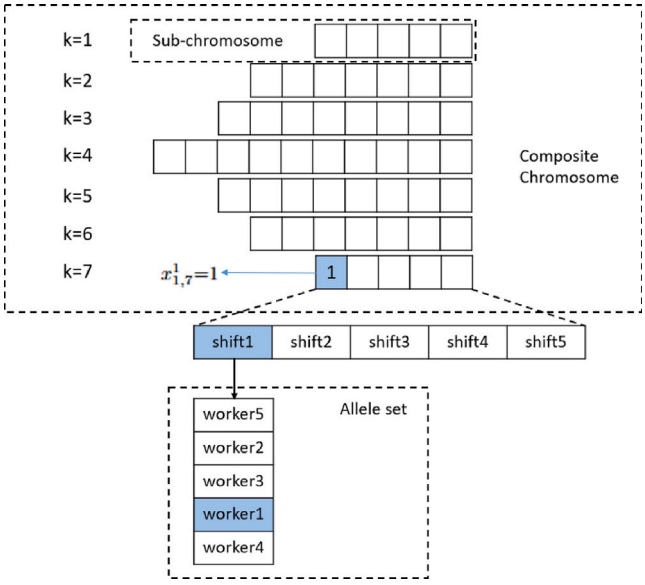


Fig. 7. Example of composite chromosome.

The complete optimization goal combines all components, where weights W_n allow adjusting priorities between cost savings and schedule quality.

5. Proposed memetic algorithm

Our solution method is based on a Memetic Algorithm (MA), and the solution is encoded as a composite chromosome composed of seven vectors, each serving as a sub-chromosome that encapsulates the schedule for a week. These vectors also account for the daily variations in demand typical of a retail store. Each vector corresponds to the shift assignments for a single day of the week. The position of a gene within the vector signifies a specific shift, and its value denotes the worker assigned to that shift. Each gene is associated with an allele set, which contains all available workers and their respective skill sets that can be assigned to the shift.

Fig. 7 shows an example of a composite chromosome constructed with 7 sub-chromosomes of index k. The 7th sub-chromosome contains the 5 required shifts in that day. The allele set of the first shift (shift1) contains 5 workers who are available and able to work on shift1. In the example, $x^1_{1,7} = 1$ stands for worker 1 ($i = 1$) is assigned to shift 1 ($j = 1$) on day 7 ($k = 7$). Each sub-chromosome behaves differently and is not affected by genetic operations on the other sub-chromosomes. The mutation operator assigns a shift to a randomly selected worker from its corresponding allele set. There are three types of crossover implemented here: single point, two points and uniform crossovers. The mutation and crossover functions are implemented to ensure the constraints regarding shifts assigned to no more than one worker are met in each sub-chromosome.

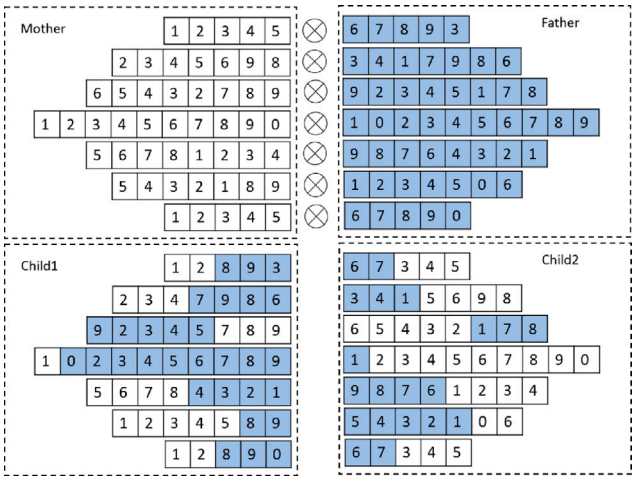


Fig. 8. Example of one point crossover.

Fig. 8 shows an example of the one point crossover operator. For detailed information on this composite chromosome, please refer to Xue et al. [34].

However, the new problem instances introduced in this paper for data instances of group 2 (see Section 7.1) incorporate several additional constraints, including minimum working hours per day, minimum working hours per week, and minimum working days per week. These constraints heighten the complexity of the problem, especially for the proposed genetic algorithm, which relies on penalty functions to maintain solution feasibility.

A direct method for addressing constraint violations is to exclude infeasible individuals from subsequent generations. However, we discovered that the incidence of violations for constraints on minimum and maximum working hours is frequently substantial, leading to a large number of infeasible solutions. Under these circumstances, the straightforward approach of excluding infeasible individuals becomes inadequate, as it often fails to yield even a single feasible solution, let alone a set of Pareto-optimal solutions.

To advance towards the Pareto front, it is essential to manage infeasible solutions more effectively than by simply discarding them. This motivates the integration of a local search process into the GA, forming a MA that combines the broad exploration capabilities of GA with problem-specific refinements. The local search mechanism, tailored to address violations of minimum and maximum working hour constraints, iteratively redistributes shifts between employees to balance duty hours and repair infeasible solutions efficiently. By incorporating this approach, the MA not only enhances the fairness of employee schedules but also accelerates the resolution of constraint violations, ensuring compliance with regulatory requirements and improving the overall quality of the solutions.

5.1. Local search

In this paper, we adopt a repair mechanism for infeasible solutions by integrating a local search process at the end of each GA iteration above mentioned. This methodology, known as a MA, extends the genetic algorithm by integrating genetic search with local refinements. The MA proposed herein endeavors to strike a balance between

the broad applicability of the GA and the problem-specific insights provided by local search.

The local search method presented in this research is tailored to address violations of the constraints related to minimum and maximum working hours. This method was initially described in [4] (see Algorithm 1). The algorithm is designed to optimize employee schedules by reducing the number of workers whose scheduled hours exceed their duty hours. It does this by iterating through pairs of employees, one with scheduled hours above duty hours (E_{above}) and the other with scheduled hours below duty hours (E_{below}). For each pair, the algorithm checks if the employee with below duty hours can take on one of the shifts of the employee with above duty hours. If this is possible, the shift is reassigned, and the schedules of the affected employees are updated. This process continues until no further improvements can be made, effectively redistributing work to ensure that employees do not exceed their duty hour limits.

Algorithm 2 Pseudo-Code of local search with Jaccard Distance measurement

Require:
E_{above}: Set of employees with scheduled working hours above duty hours
E_{below}: Set of employees with scheduled working hours below duty hours

for $e_a \in E_{above}, e_b \in E_{below}$ **do**
  for $s \in e_a.shifts$ **do**
    if e_b can work on shift s and e_a can be removed from shift s **then**
      $currentPool \leftarrow currentPool \cup \{< e_a, e_b, s >\}$
    end if
  end for
end for

while currentPool is not empty or counter < currentPool.Size **do**
  $< e_a, e_b, s > \leftarrow currentPool.Pop()$
  if $Jaccard\,Distance(currentPool, historyPool) >= J(currentPool.Size)$ **then**
    $counter \leftarrow counter + 1$
  end if
  $e_a.shifts \leftarrow e_a.shifts \setminus \{s\}$
  $e_b.shifts \leftarrow e_b.shifts \cup \{s\}$
  if obj improved **then**
    $counter \leftarrow 0$
    if e_a is at or below duty hours **then**
      $E_{above} \leftarrow E_{above} \setminus \{e_a\}$
    end if
    if e_b is at or above duty hours **then**
      $E_{below} \leftarrow E_{below} \setminus \{e_b\}$
    end if
    continue
  end if
  $historyPool \leftarrow currentPool$
  $currentPool \leftarrow currentPool \cup \{< e_a, e_b, s >\}$
  $Randomlise(currentPool)$
  $e_a.shifts \leftarrow e_a.shifts \cup \{s\}$
  $e_b.shifts \leftarrow e_b.shifts \setminus \{s\}$
end while

Incorporating this local search into the GA yields two primary advantages: First, it enhances the balance of employee duty hours, ensuring that no individual is excessively burdened with work. Second, it accelerates the repair of constraint violations related to the minimum and maximum working hours, thus ensuring that the generated schedules comply with these regulations. However, the local search algorithm outlined in Algorithm 1 is inherently greedy. Our preliminary tests indicate that it may become trapped in cycles rather than converging to global optima due to the lack of consideration for the total objective information in this local search process. To mitigate this limitation, we introduce an enhanced version of the algorithm (see Algorithm 2) that incorporates two key strategies: (1) an adaptive approach to limit the number of iterations; and (2) the consideration of global optima during local search moves.

In the refined methodology, we introduce two stacks, denoted as the $currentPool$ and the $historyPool$, to maintain records of paired employees and the shifts they are swapping. The $currentPool$ retains information on employee pairs e_a, e_b and the shift s since the commencement of the local search, while the $historyPool$ mirrors this structure but with data from previous iterations. We employ the Jaccard Distance metric to assess the similarity between these stacks. The primary objective is to monitor for local optima by identifying cycles; if the difference between the stacks is minimal, it signifies a local optima. In such cases, the $counter$ is incremented. Note that the $counter$ is incremented only if the calculated Jaccard Distance surpasses a predefined threshold, defined as $\text{JaccardDistance}(currentPool, historyPool) \geq J(currentPool.Size)$, where the threshold $J(currentPool.Size)$ is determined by the formula:

$$J(len) = \frac{1}{1 + e^{\lambda \cdot len}}$$

Note that the parameter λ requires tuning, which will be explained in Section 7.2.1. The search will terminate when the value of the $counter$ exceeds the size of the $currentPool$, or when the $currentPool$ has been completely exhausted. The Jaccard Distance is calculated as follows:

$$\text{Jaccard Distance} = 1 - \frac{|currentPool \cap historyPool|}{|currentPool \cup historyPool|}$$

6. Machine learning based Memetic Algorithm (MLMA)

This section presents the integration methodology between the machine learning model (Section 4.1) and the proposed MA. Let $\mathcal{T}$ denote a decision tree model trained on a dataset with the 9 features $f_1, f_2, \ldots, f_9$. The decision tree performs binary classification by recursively partitioning the feature space based on the values of the features. At each internal node of the tree, a feature f_i (where $i \in \{1, 2, \ldots, 9\}$) is selected, and a threshold τ_i is applied to split the data into two subsets:Left Branch: $f_i \leq \tau_i$, Right Branch: $f_i > \tau_i$. The splitting process continues recursively until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf). Each leaf node of the tree assigns a class label $y \in \{0, 1\}$ based on the majority class of the training samples that reach that node. The decision tree $\mathcal{T}$ can be formally expressed as a function: $\mathcal{T} : \mathbb{R}^9 \rightarrow \{0, 1\}$, where $\mathbb{R}^9$ represents the 9-dimensional feature space, and $\{0, 1\}$ represents the binary classification output. Given an input feature vector $\mathbf{f} = (f_1, f_2, \ldots, f_9)$, the decision tree $\mathcal{T}$ predicts the class label y as: $\mathcal{T}(\mathbf{f})$. After the models for predicting appealing shift patterns have been trained, integrating them with MA becomes straightforward. Due to the satisfactory performance achieved by employing Elitism as the selection method in the experiments detailed in [34], as well as in the MA algorithm proposed in this work, we have adopted elitism as the selection approach within the MLMA algorithm. Unlike the conventional MA, our approach introduces an additional prediction phase for each worker's shift pattern. Only shift patterns that are identified as positive (i.e. $\mathcal{T}(\mathbf{f}) = 1$), meaning they closely align with historical patterns, are the ones selected for advancement to the next generation.

The ML integration enhances MA by incorporating worker-specific decision tree models $\mathcal{T}_i$ that evaluate each worker's shift pattern appeal. As shown in Algorithm 3, for each offspring solution generated through MA operations (crossover, mutation, and local search), we verify its quality through two criteria: (1) traditional fitness evaluation and (2) ML-based appeal verification using $\mathcal{T}_i$ models. Each worker i in a solution's shift pattern is checked against their personalized model $\mathcal{T}_i$, with the solution only being considered appealing if all workers' patterns are predicted as positive ($\mathcal{T}_i(\mathbf{f}_i) = 1$ for all i). This dual evaluation strategy ensures the evolutionary process preserves both high-quality solutions (through elitism) and historically preferred shift patterns (through ML verification). The integration maintains MA's exploration capabilities while guiding the search towards solutions that balance operational requirements with worker preferences, as evidenced by the preservation mechanism in Algorithm 3 that only allows appealing solutions to progress to subsequent generations.

Algorithm 3 Pseudo-Code of Memetic Algorithm with ML Integration

Require:
P: Population size
G_{max}: Maximum number of generations
$\mathcal{T}_i$: Trained decision tree model for worker i

Initialize population P_0 with P random individuals
Evaluate fitness of all individuals in P_0
$t \leftarrow 0$
▷ Helper function to check solution appeal
function IsAppealing($solution$)
  for all $worker_i \in solution.workers$ **do**
    if $\mathcal{T}_i(worker_i.features) \neq 1$ **then**
      return false
    end if
  end for
  return true
end function
▷ Helper function to process offspring
function ProcessOffspring($offspring$, $population$)
  if IsAppealing($offspring$) **then**
    Evaluate fitness of $offspring$
    if $|population| < P$ **then**
      return $population \cup \{offspring\}$
    end if
  end if
  return $population$
end function

while $t < G_{max}$ **do**
  $P_{t+1} \leftarrow \emptyset$
  while $|P_{t+1}| < P$ **do**
    Select parents p_1, p_2 using Elitism-based selection
    Generate offspring c_1, c_2 via MA operations ▷ See Section 5 for details
    $P_{t+1} \leftarrow$ ProcessOffspring(c_1, P_{t+1})
    $P_{t+1} \leftarrow$ ProcessOffspring(c_2, P_{t+1})
  end while
  Replace P_t with P_{t+1}
  $t \leftarrow t + 1$
end while
return best individual with highest fitness and $appealing$ = true

7. Experiments

7.1. Problem instance data

Problem instances were created using historical data from a local retail store, categorized into small, medium, and large sizes, with 10 instances for each category. These instances are labeled I1, I4, and I8, where the number corresponds to the number of job skills (roles) required. For example, I4 instances require four distinct skills. In this study, h_{rest}, representing the minimum required rest period between consecutive shifts, is set to 8 h. This ensures that all workers receive at least 8 h of rest between shifts. In total, there are 30 instances, with indexes ending in 1 to 5 classified as group 1 (e.g., I1_1 to I1_5) and those ending in 6 to 10 as group 2. Group 2 introduces additional constraints, such as minimum working hours per day, per week, and minimum working days per week. The data for these instances is hosted at [33].

7.1.1. Staff demand data

Staff demand refers to the number of workers required for different roles during specific periods. A *period* is a 0.5-h interval, resulting in 48 periods per day. The planning horizon is typically one week. Table 6 provides an example of staff demand across different roles and periods.

7.1.2. Worker data

Worker data includes availability, working hours, and qualifications. Table 8 details worker-specific constraints, such as minimum and maximum working hours and days. Table 7 indicates worker availability for each day of the week, and Table 9 specifies worker qualifications for different roles.

Table 6
Example of staff demand.

Weekday index	Period	Role	Staff demand
0	8:00–8:30	0	0
0	8:00–8:30	1	1
0	8:00–8:30	2	1
0	8:00–8:30	3	0
…			

Table 7
Worker availability by day: Binary representation of weekly availability.

Worker ID	Mon	Tue	Wed	Thu	Fri	Sat	Sun
0	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1
2	1	1	1	1	1	1	1

Table 8
Worker attributes and constraints: Hourly payment, availability, and working hour limits.

Attribute	Worker 0	Worker 1	Worker 2
Hourly payment	7.7	7.7	5.6
Start time	6	6	6
End time	23	15	23
Max consecutive days	5	5	5
Max weekly hours	50	40	40
Max daily hours	12	12	12
Max weekly days	5	7	7
Min weekly hours	10	15	20
Min daily hours	5	3	4
Min weekly days	0	2	3

Table 9
Worker roles/qualifications.

Role	Worker 0	Worker 1	Worker 2
Role 0	1	1	1
Role 1	0	1	1
Role 2	0	1	1
Role 3	1	1	1

7.2. Experiments on MA

Our experimental evaluation follows a two-stage approach: First, we evaluate the baseline MA, then proceed to analyze the MLMA. This separation is motivated by fundamental differences in their solution processes. Unlike the MA which operates directly on problem instances, the MLMA requires an initial phase of model training and subsequent integration with the optimization framework. Due to these distinct methodological requirements, we dedicate Section 7.3 exclusively to the MLMA experiments. For the MA approach, there are several critical parameters that require tuning. We initially tune these critical parameters. Following this tuning process, we compare the outcomes of these methods against those obtained using a Linear Programming (LP) solver for the shift design problem and a Genetic Algorithm (GA) for the shift assignment problem (LP-GA), as well as those from a solution approach implemented and solved by a one-stage linear programming method (1S-LP) and a two-stage decomposed linear programming method (2S-LP). Additionally, we compare the results against methods that employ an LP solver for the shift design problem and a MA for the shift assignment problem (LP-MA). After testing the performance of the MA, we proceed to evaluate the MLMA to demonstrate the effectiveness of our MLMA method in learning from historical data.

7.2.1. Algorithm tuning

The proposed MA with composite chromosome was implemented in C++ and executed on a PC equipped with an Intel i7 2.40 GHz processor and 4 GB RAM. The penalty weights are as follows: $W_1 = 10$,

$W_2 = 5$, $W_3 = 50$, $W_4 = 100$, $W_5 = 200$, $W_6 = 150$, $W_7 = 50$, $W_8 = 50$. The values of W_1 and W_2 (i.e., the penalty weights for the unfair assignment of unpopular shifts and the unfair assignment of total working hours, respectively) were determined in consultation with the store manager. The remaining weight values were established based on preliminary experiments conducted on a few instances of the problem, with the understanding that these may not be the optimal penalty weights for the problem at hand.

Given the complexity and the multitude of parameters involved in the MA, we automated the configuration of the algorithm using Irace [40]. A total of 6 data instances (I1_6, I1_7, I4_6, I4_7, I8_6, I8_7) were considered. A subset of the most critical parameters was identified for tuning within the MA. The parameters used in the previous experiments (for instance group 1, as referenced in [34]) remain unchanged. Preliminary experimentation revealed that a small population size evolved over a large number of generations produces good solutions in short computation time. Population parameters were set by categorical values within (5, 10, 20, 30, 40, 50), while mutation parameters were real values within the range (0.01, 0.5). The maximum number of experiments was set as large as 1000; other parameters for Irace were set to default values. The results indicated that a population size of 10, a mutation rate of 0.007, with $\lambda = -0.0671$ (the parameter utilized in the local search phase), and a one-point crossover are the most appropriate for the problem.

7.2.2. Algorithm performance experiments

The models in Section 4 were solved using Gurobi 11.0 with its default optimization parameters, including a relative optimality gap tolerance of 0.01%, integer feasibility tolerance of 1×10^{-5}, and no imposed runtime limit. To verify that our conclusions are not solver-dependent, we also solved the instances using FICO Xpress with the same settings as those used for Gurobi. As FICO Xpress demonstrated inferior performance compared to Gurobi for all problem instances, we present only the Gurobi results in the main text for simplicity. The comparative results between Gurobi and FICO Xpress are summarized in Table 14 in Appendix. To assess the efficacy of the proposed solution methods, we subjected the algorithms to testing on 30 problem instances, as detailed in Section 3, employing the parameter values and penalty weights specified in Section 7.2.1. To ensure statistical robustness, the LP-GA and LP-MA methods were independently executed 20 times each across all 30 datasets, with mean and standard deviation values recorded. For each instance, all runs began with a randomly generated initial solution.

Table 10 displays the computational outcomes for all solution methods, including the runtimes for the LP-GA and LP-MA approaches. The results for 1S-LP and 2S-LP are the objective function values of the integer feasible solutions found by the Gurobi solver at the same computational time as the LP-GA or LP-MA method for the corresponding problem instance (whichever takes longer). 1S-LP-LB and 2S-LP-LB represent the best known lower bounds obtained by the Gurobi solver within 600 s, as the solver typically runs out of memory after this time. Another reason is that this amount of solving time is also somewhat comparable to that of the proposed heuristics. Data instance group 1 is missing the specific constraints that are essential for the LP-MA algorithm to address. Running the LP-MA on such instances would be inappropriate given the algorithm's intended use. Hence, no experiments were performed on this group of instances.

Regarding computational time, the 1S-LP and 2S-LP methods necessitate the use of the Gurobi solver to achieve optimal solutions with default settings, which typically cannot be completed within a few hours. As such, their runtimes are not comparable to those of the heuristics proposed in this paper, and thus, the computational times for these methods are excluded from comparison in this study.

For the elapsed time of LP-GA and LP-MA, the shift design phase is solved using a linear programming model via Gurobi with consistent default settings. Consequently, for each instance, the output is identical across multiple runs. Therefore, in our experiments, we executed the shift design solver only once per instance, and the results were stored as input for the shift assignment phase. The times reported in the columns labeled LP-GA-TIME and LP-MA-TIME represent the runtime of the shift assignment phase alone, excluding the shift design phase. To ensure a fair comparison, the average values presented in the final row of results are calculated exclusively for instance group 2 (i.e., those instances not marked as NA for LP-MA-related outcomes: I1_6–I1_10, I4_6–I4_10, I8_6–I8_10).

7.2.3. Comparison of different methods

Fig. 9 illustrates the comparison between the LP-GA, 2S-LP, and 2S-LP-LB. It is evident that the LP-GA demonstrates the effectiveness of the proposed approach in generating high-quality solutions for the data instances in instance group 1. However, the LP-GA encounters challenges in finding feasible solutions for some instances in instance group 2, particularly for instances I4_8, I4_9, I4_10, I8_8, I8_9, and I8_10, where a significant gap is observed due to the penalty imposed on infeasible solutions. To address this issue, as previously discussed, a local search procedure was incorporated into the GA (i.e. LP-MA) to enhance its capability to generate feasible solutions.

Fig. 10 illustrates the comparison between the LP-MA and 2S-LP approaches. The average results as well as their standard deviations from all experiments are presented in Table 10. Due to LP-MA's local search component being tailored to address the additional constraints present in data instance group 2, LP-MA exclusively solves problems for this group. The Wilcoxon signed-rank test reveals statistically significant differences between the methods (two-tailed, $p < 0.001$), with LP-MA demonstrating superior solution quality relative to LP-GA. LP-MA provides solutions that are closer to the 2S-LP-LB, which represents the best-known lower bound, demonstrating its capability.

In summary, the overall performance comparison of all the solution methods is presented in Fig. 11. When considering the penalty cost, the results are ordered in the following sequence: 1S-LP, 2S-LP, LP-MA, LP-GA. This ordering indicates that the 1S-LP method incurs the lowest penalty cost, followed by the 2S-LP method, then the LP-MA method, and finally the LP-GA method, which has the highest penalty cost. The Wilcoxon Signed-Rank Test confirms statistically significant differences between all pairs ($p < 0.001$ for 1S-LP vs. 2S-LP, LP-GA vs. LP-MA, 1S-LP vs. LP-MA, and 2S-LP vs. LP-GA). This trend indicates that the LP-GA method may be less proficient in satisfying all constraints, which can result in a higher penalty for infeasible solutions in certain cases. However, this issue can be mitigated by employing the LP-MA method. In terms of computational time, it is evident that the LP-MA approach required a longer duration, averaging nearly 39% more time. This increase is anticipated, given the incorporation of a local search mechanism to address the more challenging constraints.

7.3. Experiment with MLMA

We conduct experiment to demonstrate how well our MLMA method learns from historical data. For this experiment, the initial step involved constructing training data, which includes both historical schedules and schedules produced by the MA, as described earlier in Sections 5 and 6. In practice, preparing training data for the ML component of MLMA can be challenging due to the scarcity of historical schedules that span a continuous time frame and include all employees. Moreover, determining whether shift patterns are appealing based on their similarity to past schedules created by human schedulers can be very subjective. To facilitate measurement and analysis, we chose not to use real historical schedules for the training data in this experiment. Instead, we generate synthetic data. To streamline its preparation and assess MLMA's effectiveness, the data are designed to reflect end-user preferences (i.e., the human scheduler's tendency to minimize the number of scheduled employees). This synthetic dataset is intentionally structured for clearer benchmarking, evaluating MLMA's core capability in workforce

Table 10
Computational results comparing solution methods: objective values and runtimes (Mean ± Standard deviation) for LP-GA, LP-MA, and MIP-based approaches (1S-LP, 2S-LP).

Instance	1S-LP	1S-LP-LB	2S-LP	2S-LP-LB	LP-GA	LP-GA-TIME	LP-MA	LP-MA-TIME
I1_1	2698.34	2645.77	2836.45	2668.24	2890.12 ± 18.2	141.16 ± 5.3	NA	NA
I1_2	2834.52	2734.39	2890.35	2799.15	2955.05 ± 19.4	98.35 ± 4.7	NA	NA
I1_3	2884.33	2372.91	2494.55	2421.33	2525.74 ± 17.3	81.04 ± 3.9	NA	NA
I1_4	2515.38	2410.09	2568.09	2466.67	2577.89 ± 14.6	104.51 ± 5.1	NA	NA
I1_5	3197.43	2937.88	3221.74	3086.82	3243.68 ± 21.2	145.42 ± 6.8	NA	NA
I1_6	3842.75	3366.23	3574.68	3499.75	3766.08 ± 24.3	180.64 ± 8.2	3664.76 ± 23.5	187.71 ± 8.9
I1_7	3257.79	2994.48	3277.27	3247.27	3389.83 ± 22.6	291.62 ± 12.4	3356.83 ± 21.9	323.22 ± 14.1
I1_8	3181.73	2868.88	3169.14	2997.73	3314.45 ± 21.8	258.58 ± 11.7	3297.11 ± 20.6	308.97 ± 13.8
I1_9	3269.82	3136.61	3240.57	3223.64	3523.05 ± 24.1	203.54 ± 9.6	3379.55 ± 22.7	261.95 ± 12.3
I1_10	2945.68	2694.11	2965.27	2754.43	3052.84 ± 20.7	295.81 ± 13.5	3030.06 ± 19.8	316.05 ± 14.9
I4_1	1735.73	1577.55	2321.91	2126.82	2318.41 ± 15.1	57.49 ± 3.2	NA	NA
I4_2	2069.38	1875.72	2275.76	2183.33	2300.21 ± 16.2	146.51 ± 7.1	NA	NA
I4_3	1960.33	1861.84	2287.09	2090.48	2295.91 ± 15.7	153.04 ± 7.4	NA	NA
I4_4	1942.03	1901.51	2760.51	2475.82	2768.36 ± 18.1	170.78 ± 8.3	NA	NA
I4_5	2094.29	1968.61	2661.05	2494.88	2686.67 ± 17.9	155.08 ± 7.6	NA	NA
I4_6	2522.91	2409.32	2614.93	2594.34	2728.94 ± 18.7	191.55 ± 9.2	2716.16 ± 18.3	242.32 ± 11.6
I4_7	2533.26	2407.95	2617.69	2617.48	2737.74 ± 18.9	244.79 ± 11.8	2695.45 ± 18.1	304.38 ± 14.5
I4_8	3021.91	2959.58	3156.47	3155.56	5155.87 ± 32.7	301.44 ± 14.9	3229.63 ± 21.3	452.18 ± 21.3
I4_9	3212.89	3110.42	3309.38	3309.31	5164.69 ± 33.1	213.36 ± 10.7	3423.79 ± 22.6	333.14 ± 16.2
I4_10	3142.81	3052.02	3257.48	3256.42	5359.28 ± 35.8	190.57 ± 9.8	3447.65 ± 23.1	239.36 ± 12.1
I8_1	2535.04	2268.31	3212.04	2899.55	3253.93 ± 21.5	131.88 ± 6.7	NA	NA
I8_2	2377.23	2173.51	2983.44	2615.41	2991.93 ± 20.0	169.96 ± 8.3	NA	NA
I8_3	2356.53	2170.51	2832.86	2683.58	2838.79 ± 19.5	170.59 ± 8.4	NA	NA
I8_4	2473.62	2203.31	2715.72	2478.89	2725.26 ± 18.9	160.89 ± 8.0	NA	NA
I8_5	2506.16	2222.57	2897.46	2630.99	2951.91 ± 20.3	176.43 ± 8.8	NA	NA
I8_6	2737.95	2678.59	2962.71	2904.65	3072.02 ± 21.6	328.52 ± 15.9	3059.33 ± 20.1	448.13 ± 21.1
I8_7	2951.91	2860.85	3135.31	3134.99	3279.33 ± 22.8	342.01 ± 16.7	3270.87 ± 21.5	457.15 ± 22.3
I8_8	3232.57	3164.77	3530.14	3526.75	4772.38 ± 30.2	325.35 ± 16.1	3609.45 ± 23.9	348.15 ± 17.3
I8_9	3310.98	3221.63	3475.39	3474.12	4584.48 ± 29.3	220.11 ± 11.2	3774.35 ± 24.8	429.95 ± 20.9
I8_10	3117.31	3010.41	3295.77	3295.28	4725.39 ± 30.1	456.93 ± 22.4	3518.06 ± 23.3	539.28 ± 26.3
Average	2731.04	2651.43	2963.27	2892.12	3260.63 ± 24.7	244.61 ± 13.1	3049.66 ± 23.1	341.18 ± 18.4

Wilcoxon signed-rank test (Two-tailed).
p-values: 1S-LP vs. 2S-LP: < 0.001 LP-GA vs. LP-MA: < 0.001 1S-LP vs. LP-MA: < 0.001 2S-LP vs. LP-GA: < 0.001.

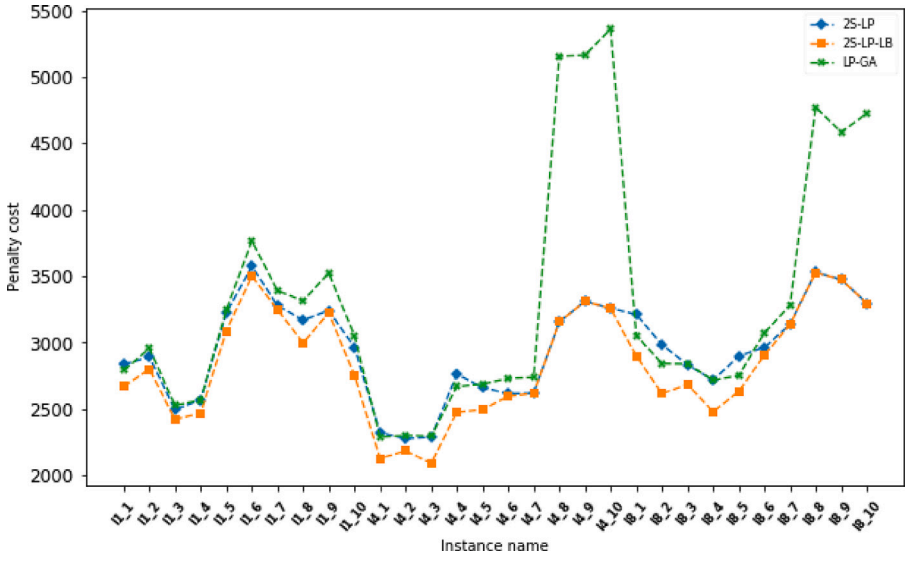


Fig. 9. Comparison of 2S-LP and 2S-LS-LB and LP-GA.

allocation optimization with minimal staffing as the implicit objective. While this simplified scenario does not explicitly model individual shift preferences, it serves two critical purposes: (1) establishing a baseline for algorithmic performance measurement, and (2) isolating the metric that most directly impacts scheduling solutions.

For the machine learning component of the MLMA, we prepared the training and test datasets using instances that had previously been prepared for the solver, as detailed in Section 6. In the field of machine learning, the terms **training instances** and **training set** are often used interchangeably. However, in this paper, to maintain clarity, we clarify our terminology as follows:

Training Set: This refers to the data utilized by the machine learning component of the MLMA. Specifically, it is the data used to derive a decision tree model, as detailed in Section 6.

Training Instance: As outlined in Section 7.1, these are the **problem instances** used to generate the **Training Set**. For example, the training instance I1_6 was used to create the training set by executing

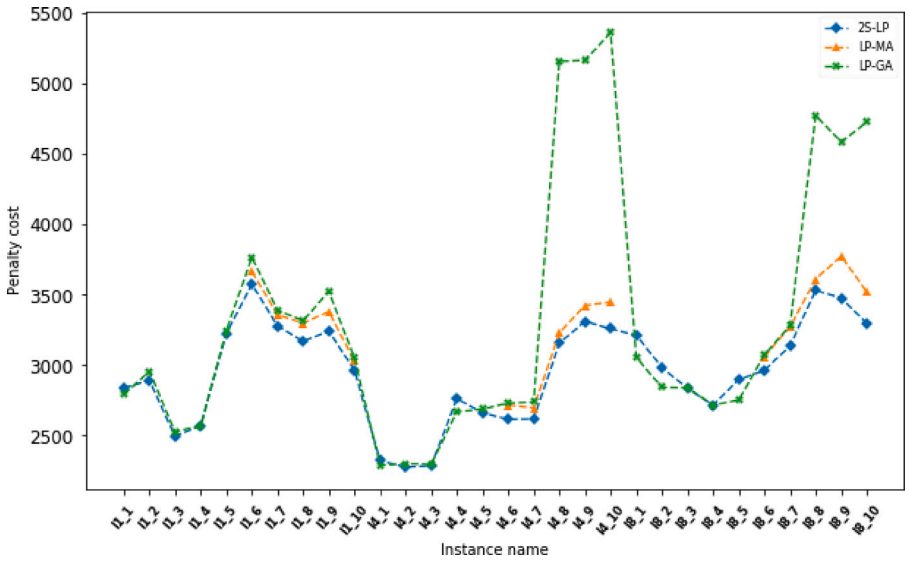


Fig. 10. Comparison of 2S-LP and LP-MA and LP-GA.

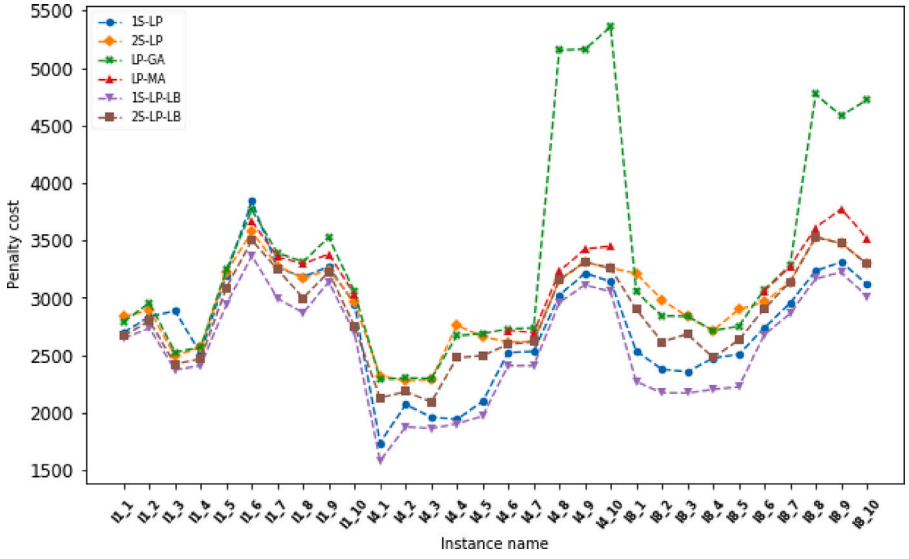


Fig. 11. Comparison of results of all solution methods.

the MA multiple times. Consequently, the training set comprises the schedules produced by the MA.

Test Instance: After the ML prediction model has been obtained, these models will be used to test the performance of the MLMA. The performance is also compared with that of the MA. For the sake of simplicity in our discussion, the **problem instances** used for this test purpose are referred to as **test instances** when they are being discussed in this context.

Another point that requires clarification is that we have proposed a two-phase solution. However, since the first phase, involving shift design, is based on a deterministic approach, we execute it only once and save the output for multiple experimental runs.

We selected three similar training instances based on demand and available employees from each of the three instance groups: 1 skill (I1), 4 skills (I4), and 8 skills (I8). From each group, we designated two or three instances for generating the training set, and one instance was chosen as the test instance. The selected training instances and test instance are listed in Table 11.

Once the training instances are chosen, the following step is to build the training set. For this, we used the MA algorithm to generate solutions. The process for creating the data involved the following steps: First, we modified the MA's objective function to incorporate a substantial additional cost for adding an extra employee to the schedule. This adjustment was intended to produce schedules with fewer employees. The data obtained from this method were labeled as positive samples. Next, we used the same data and the MA algorithm without altering the objective function to create negative samples. The final training set comprises 1000 positive samples (i.e., *Appealing*, designated as class 1) and 1000 negative samples (i.e., *Not Appealing*, designated as class 0), totaling 2000 samples, or schedules, for the training set.

We used the Decision Tree algorithm from the scikit-learn library [41] to obtain models for each worker offline, and then we hard-coded the learned decision trees into the MA algorithm. The parameters of the classification algorithm were fine-tuned manually. The training set, which comprised 9 features, was used to train the algorithm. The ten-fold cross-validation accuracy rates for the Decision Tree across the different datasets were ranging from 71% to 78%. The goal is to learn as much knowledge as possible from historical schedules (e.g., schedules created by experts).

Table 11
Computational results comparing optimization methods: Objective values, employees, times, and OC (Mean ± Standard deviation) for MA and MLMA.

Training instance	Test instance	Objective value		Employees		Runtime (s)		OC	
		MA	MLMA	MA	MLMA	MA	MLMA	MA	MLMA
I1_7, I1_8	I1_6	3664.76 ± 21.3	3582.54 ± 20.8	48.00	46.85 ± 0.32	222.46 ± 12.7	305.86 ± 12.3	0.314 ± 0.021	0.512 ± 0.034
I1_6, I1_8	I1_7	3356.83 ± 18.7	3346.97 ± 18.2	48.00	46.37 ± 0.28	266.79 ± 15.2	376.45 ± 14.7	0.302 ± 0.019	0.497 ± 0.032
I1_6, I1_7	I1_8	3297.11 ± 19.5	3372.37 ± 19.1	48.00	45.29 ± 0.35	208.38 ± 11.8	294.22 ± 11.4	0.327 ± 0.022	0.528 ± 0.035
I4_8	I4_7	2695.45 ± 16.2	2663.42 ± 15.9	48.00	47.93 ± 0.25	310.87 ± 17.6	437.29 ± 17.0	0.291 ± 0.018	0.486 ± 0.031
I4_7	I4_8	3229.63 ± 20.1	3233.76 ± 19.6	48.00	46.46 ± 0.26	314.54 ± 18.9	443.62 ± 18.3	0.283 ± 0.017	0.479 ± 0.030
I8_7, I8_8	I8_6	3059.33 ± 17.8	3118.49 ± 17.4	49.00	44.34 ± 0.38	475.20 ± 19.5	669.66 ± 18.9	0.268 ± 0.016	0.463 ± 0.029
I8_6, I8_8	I8_7	3270.87 ± 19.2	3225.27 ± 18.8	49.00	45.65 ± 0.30	425.64 ± 16.3	593.87 ± 15.8	0.257 ± 0.015	0.452 ± 0.028
I8_6, I8_7	I8_8	3609.45 ± 21.9	3642.61 ± 21.4	48.66 ± 0.48	44.84 ± 0.39	407.55 ± 15.7	568.83 ± 15.2	0.246 ± 0.014	0.441 ± 0.027
Average		3272.80 ± 19.4	3273.05 ± 18.9	48.33 ± 0.48	45.97 ± 0.32	328.92 ± 16.1	461.20 ± 15.6	0.286 ± 0.018	0.482 ± 0.031

Wilcoxon signed-rank test (Two-tailed).
MA vs. MLMA comparisons: Objective: $p = 0.15$ Employees: $p = 0.008$ Runtime: $p < 0.001$ OC: $p < 0.001$.

Table 12
Classification of personnel scheduling literature based on shift flexibility and skill heterogeneity.

Fixed and homogeneous	Bard and Wan [42], Bester et al. [43], Bhatnagar et al. [44], Akbari et al. [45], Awadallah et al. [46], Bard and Purnomo [47], Bilgin et al. [48,49], Haspeslagh et al. [50], Burke and Curtois [51], Cipriano et al. [52], Ertogral and Bamuqabel [53], Hadwan and Ayob [54], Helber and Henken [55], Lin et al. [56], Krishnamoorthy et al. [57], Li et al. [58], Lü and Hao [59], Maenhout and Vanhoucke [60], Nonobe [61], Ovchinnikov and Milner [62], Rönnberg and Larsson [63], Safaei et al. [64], Stark and Zimmermann [65], Topaloglu [66], Valouxis et al. [67], Post and Veltman [68], Wright et al. [69], Yang et al. [70], Zhu and Sherali [71], Syberfeldt et al. [72], Bruni and Detti [73], De Bruecker et al. [74], Dahmen et al. [75] and Guo and Bard [76]
Fixed and heterogeneous	Aickelin et al. [77], Aickelin and Dowsland [38], Aickelin and White [78], Al-Yakoob and Sherali [79], Azaiez and Al Sharif [80], Bai et al. [39], Bard [81], Bard and Purnomo [82], Bard and Wan [17], Beddoe et al. [83], Beddoe and Petrovic [84], Brucker et al. [85], Burke et al. [86], Eiselt and Marianov [87], Goodman et al. [88], Günther and Nissen [89], Li and Aickelin [90], Li et al. [91], Özcan [92], Qi and Bard [93], Seçkiner et al. [94], Shahnazari-Shahrezaei et al. [95], Parr and Thompson [96], Valls et al. [97], Wan and Bard [18], Henao et al. [98] and Parisio and Jones [99]
Flexible and homogeneous	Avramidis et al. [100,101,102], Brunner et al. [103], Brunner and Edenharter [104], Canon [105], Cappanera and Gallo [106], Cezik and L'Ecuyer [107], Drezet and Billaut [108], Eitzen et al. [109], Ertogral and Bamuqabel [53], Eveborn et al. [110], Eveborn et al. [111], Hanne et al. [112], Hojati and Patil [19], Ingolfsson et al. [113], Knust and Schumacher [114], Sabar et al. [115], Wright and Bretthauer [116], Zolfaghari et al. [117], Omar et al. [20], Ganguly et al. [21], Kazemian et al. [118] and Ásgeirsson and Sigurdhardóttir [22]
Flexible and heterogeneous	Bard [81,119], Bard and Wan [17], Gordon and Erkut [14], Gutjahr and Rauner [26], Zolfaghari et al. [13], Mirrazavi and Beringer [16], Qi and Bard [93], Thiel [27], Wan and Bard [18], Brucker and Qu [31], Prot et al. [30], Smet et al. [28], Lin and Ying [29] and Zamorano and Stolletz [32]

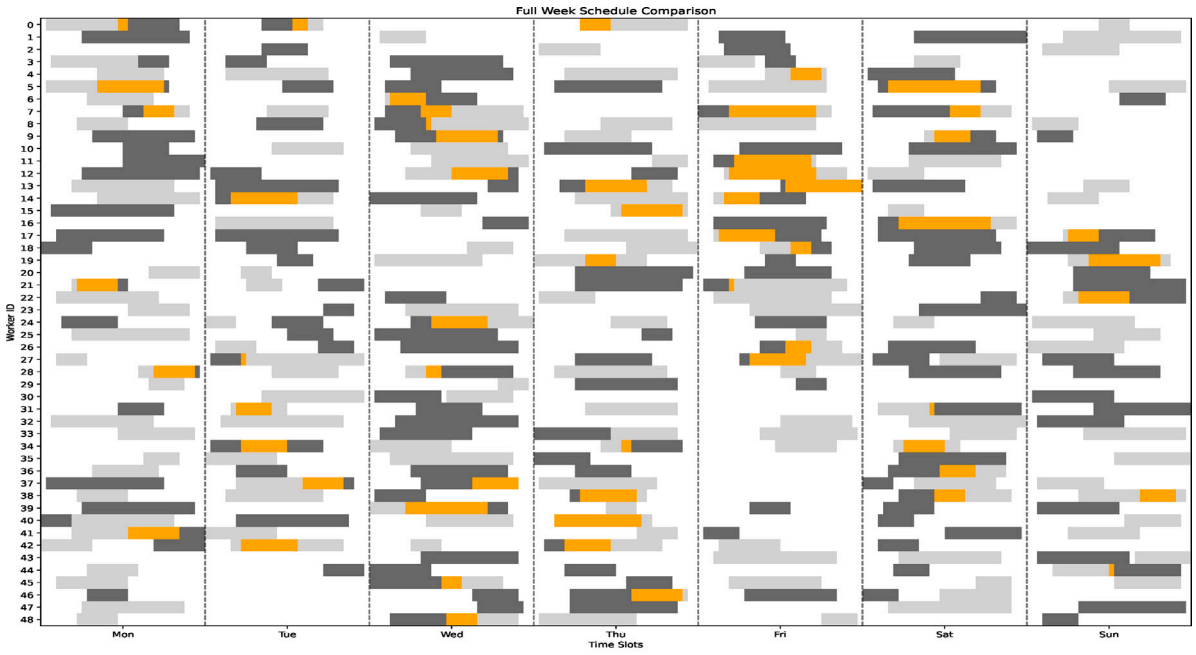
Table 13
Applications of personnel scheduling models across industries.

Call center	Avramidis et al. [100,101,102], Bhulai et al. [120], Canon [105], Avramidis et al. [101], Ertogral and Bamuqabel [53], Ingolfsson et al. [113], Ertogral and Bamuqabel [53] and Helber and Henken [55]
General	Bard [81], Bard and Wan [17], Qi and Bard [93], Wan and Bard [18], Brucker and Qu [31], Prot et al. [30], Smet et al. [28], Lin and Ying [29], Drezet and Billaut [108], Eitzen et al. [109], Ásgeirsson and Sigurdhardóttir [22], Al-Yakoob and Sherali [79], Bard [81], Bard and Wan [17], Beddoe et al. [83], Eiselt and Marianov [87], Qi and Bard [93], Seçkiner et al. [94], Wan and Bard [18], Henao et al. [98], Bard and Wan [42], Krishnamoorthy et al. [57], Stark and Zimmermann [65], Yang et al. [70] and Dahmen et al. [75]
Health care	Gutjahr and Rauner [26], Brunner et al. [103], Brunner and Edenharter [104], Eveborn et al. [110], Eveborn et al. [111], Wright and Bretthauer [116], Omar et al. [20], Ganguly et al. [21], Kazemian et al. [118], Aickelin et al. [77], Aickelin and Dowsland [38], Aickelin and White [78], Azaiez and Al Sharif [80], Bai et al. [39], Bard and Purnomo [82], Brucker et al. [85], Burke et al. [86], Goodman et al. [88], Li and Aickelin [90], Li et al. [91], Özcan [92], Parr and Thompson [96], Akbari et al. [45], Awadallah et al. [46], Bard and Purnomo [47], Bester et al. [43], Bilgin et al. [48,49], Burke and Curtois [51], Cipriano et al. [52], Hadwan and Ayob [54], Harper et al. [121], Li et al. [58], Lü and Hao [59], Maenhout and Vanhoucke [60], Nonobe [61], Ovchinnikov and Milner [62], Rönnberg and Larsson [63], Valouxis et al. [67], Wright et al. [69], Bruni and Detti [73], Guo and Bard [76], Chen et al. [122] and Koruca et al. [123]
Manufacturing	Bard and Wan [17], Sabar et al. [115], Shahnazari-Shahrezaei et al. [95], Bard and Wan [42], Bhatnagar et al. [44] and Hadwan and Ayob [54]
Other	Bard [119], Gordon and Erkut [14], Zamorano and Stolletz [32], Hojati and Patil [19], Valls et al. [97], Lin et al. [56], Safaei et al. [64], Topaloglu [66], Post and Veltman [68], Zhu and Sherali [71], Syberfeldt et al. [72] and Chen et al. [124]
Retail	Mirrazavi and Beringer [16], Zolfaghari et al. [117] and Parisio and Jones [99]
Transportation	Thiel [27], Cappanera and Gallo [106], Hanne et al. [112], Knust and Schumacher [114], Günther and Nissen [89], De Bruecker et al. [74] and Zhang et al. [125]

7.3.1. Pattern learning performance analysis

Following the development of our decision tree models, we conducted a comprehensive performance evaluation comparing MLMA with conventional MA across two critical dimensions. First, we assessed fundamental optimization capabilities through solution quality (Objective Value) and computational efficiency (Runtime), with detailed comparative results presented in Table 11. Second, and more importantly, we evaluated the quality of learned shift patterns by examining how closely MLMA-generated schedules replicate desirable characteristics from the training data. This evaluation focused specifically on two key metrics: (1) workforce efficiency, measured through the number of employees required in MLMA-generated versus MA-generated schedules, and (2) schedule similarity, quantified by the alignment between algorithm-generated schedules and historical patterns, is measured by Overlap Coefficient (OC); details are provided in Appendix A.3.

The experiment was executed 30 times for both MLMA and MA to compare solution quality across multiple metrics. The average results as well as their standard deviations from all experiments are presented. Wilcoxon signed-rank tests suggested that there was no significant difference between MA and MLMA in objective values. The results show



(a) MA schedule alignment with training data (OC=0.232) for I4_1

(b) MLMA schedule alignment with training data (OC=0.476) for I4_1

Fig. 12. Comparison of schedule similarity for instance I4_1.

that both methods achieved comparable objective values (3272.80 for MA vs. 3273.05 for MLMA), demonstrating similar optimization capability for the primary scheduling criteria. However, MA consistently uses more employees than MLMA in each instance - on average, the MLMA method used 45.97 employees, while the MA method used an average of 48.33 employees. As anticipated, the MLMA method required more computation time than the MA method, with an average run time of 461.20 s for MLMA compared to 328.92 s for MA. These results underscore the efficiency of the MLMA algorithm in discovering shift patterns that minimize the number of employees needed. The algorithm achieves this without the necessity of imposing additional objectives to penalize extra workers within the solution, relying solely on the guidance provided by machine learning predictions.

We also use the Overlap Coefficient (OC; Appendix A.3) to quantitatively measure how well our ML-enhanced algorithms learn scheduling patterns from training data. Fig. 12 in the Appendix provides visual evidence of this relationship by showing how different OC values manifest in actual schedules relevant to data instance I4_1. The three-color scheme clearly distinguishes between training data schedules (light gray), generated schedules (dark gray), and their overlapping shifts (orange). This visualization demonstrates the performance difference between the algorithms: while MA's alignment (OC = 0.232) shows limited pattern matching, MLSA's correspondence (OC = 0.476) visually demonstrates its superior learning capability. The OC results, computed across all experimental trials, are presented in Table 11. The

Table 14
Comparison of solution methods: Gurobi vs. FICO Xpress.

Method	I1_1	I1_2	I1_3	I1_4	I1_5	I1_6	I1_7	I1_8	I1_9	I1_10
1S-LP_G[a]	**2698.34**	**2834.52**	**2884.33**	**2515.38**	**3197.43**	**3842.75**	**3257.79**	**3181.73**	**3269.82**	**2945.68**
1S-LP_F[b]	2725.32	2862.87	2913.17	2540.53	3229.40	3881.18	3290.37	3213.55	3302.52	2975.14
2S-LP_G[a]	**2836.45**	**2890.35**	**2494.55**	**2568.09**	**3221.74**	**3574.68**	**3277.27**	**3169.14**	**3240.57**	**2965.27**
2S-LP_F[b]	2864.81	2919.25	2519.50	2593.77	3253.96	3610.43	3310.04	3200.83	3272.98	2994.92
Method	I4_1	I4_2	I4_3	I4_4	I4_5	I4_6	I4_7	I4_8	I4_9	I4_10
1S-LP_G[a]	**1735.73**	**2069.38**	**1960.33**	**1942.03**	**2094.29**	**2522.91**	**2533.26**	**3021.91**	**3212.89**	**3142.81**
1S-LP_F[b]	1753.09	2090.07	1979.93	1961.45	2115.23	2548.14	2558.59	3052.13	3245.02	3174.24
2S-LP_G[a]	**2321.91**	**2275.76**	**2287.09**	**2760.51**	**2661.05**	**2614.93**	**2617.69**	**3156.47**	**3309.38**	**3257.48**
2S-LP_F[b]	2345.13	2298.52	2309.96	2788.12	2687.66	2641.08	2643.87	3188.04	3342.47	3290.05
Method	I8_1	I8_2	I8_3	I8_4	I8_5	I8_6	I8_7	I8_8	I8_9	I8_10
1S-LP_G[a]	**2535.04**	**2377.23**	**2356.53**	**2473.62**	**2506.16**	**2737.95**	**2951.91**	**3232.57**	**3310.98**	**3117.31**
1S-LP_F[b]	2560.39	2401.00	2380.10	2498.36	2531.22	2765.33	2981.43	3264.90	3344.09	3148.48
2S-LP_G[a]	**3212.04**	**2983.44**	**2832.86**	**2715.72**	**2897.46**	**2962.71**	**3135.31**	**3530.14**	**3475.39**	**3295.77**
2S-LP_F[b]	3244.16	3013.27	2861.19	2742.88	2926.43	2992.34	3166.66	3565.44	3510.14	3328.73

[a] 1S-LP_G, 2S-LP_G: Results obtained by Gurobi solver.
[b] 1S-LP_F, 2S-LP_F: Results obtained by FICO Xpress solver.

comprehensive results reveal significant performance differences between MA and MLMA, with MLMA consistently achieving substantially higher OC values (0.441–0.528 across all test instances) compared to MA (0.246–0.327), demonstrating superior alignment with historical scheduling patterns.

8. Conclusion

Personnel scheduling is a well-studied optimization problem, yet its diverse nature and varying complexity present significant challenges in developing universally effective solutions. This paper addresses the intricate shift design and shift assignment problems prevalent in retail and service industries, characterized by multi-skilled workers, flexible shifts, and multiple scheduling objectives. We propose a novel two-phase approach to tackle these complexities, combining integer programming with a memetic algorithm grounded in an efficient composite chromosome encoding. The memetic algorithm integrates local search mechanisms to resolve constraints that traditional genetic algorithms struggle to address, significantly enhancing solution quality.

Our study introduces a robust two-phase methodology combining integer programming and a memetic algorithm to effectively solve multi-skilled personnel scheduling problems, ensuring feasibility while balancing fairness and working hours. By incorporating historical scheduling data, our enhanced memetic algorithm learns from past expert-created schedules, producing solutions that are both optimized and operationally practical. Furthermore, we propose a Machine Learning-assisted Memetic Algorithm (MLMA) that leverages machine learning to prioritize human-preferred scheduling patterns, significantly improving the real-world applicability of automated scheduling systems.

Extensive computational experiments on 30 real-world instances demonstrate the effectiveness of our proposed approach. Comparative analysis shows that our MA achieves superior solution quality compared to a standard GA approach, particularly for large-scale instances involving eight distinct work skill requirements. Notably, the MA solution quality is close to that obtained by commercial MILP solvers, while maintaining practical applicability. Furthermore, our MLMA improves upon these results by generating shift patterns that better align with historical scheduling preferences, thereby enhancing both solution quality and operational practicality.

While the current results demonstrate significant progress, future research could explore additional refinements and extensions to further enhance the proposed methods. Although our two-phase solution is theoretically close to optimal, the gap to the lower bound indicates potential for further improvement through refined parameter tuning of the memetic algorithm. Additionally, the MLMA approach requires a more comprehensive analysis, particularly in training set preparation and the selection of machine learning algorithms. In this work, we selected 7 individual-specific features and 2 schedule-specific features; however, future efforts will focus on feature engineering and selection to refine and expand the schedule-specific features. These directions will be central to our future research, aiming to further bridge the gap between theoretical optimization and real-world application.

CRediT authorship contribution statement

Ning Xue: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Ruibin Bai:** Writing – review & editing, Supervision. **Huan Jin:** Writing – review & editing. **Tianxiang Cui:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Ningbo Science and Technology Bureau (Project IDs: 2022Z217 and 2024S057) as well as the Ningbo Natural Science Foundation (Project ID: 2023J194).

Appendix

A.1. Tables

See Tables 12–14.

A.2. Figures

See Fig. 12.

A.3. Overlap coefficient

In this study, we use Overlap Coefficient (Szymkiewicz–Simpson Coefficient) to calculated the similarity between two schedules. The Overlap Coefficient is calculated as following:

A.3.1. Worker-level calculation

The **Overlap Coefficient (OC)** between two workers' shift schedules S_1^w and S_2^w is formally defined as:

$$\mathrm{OC}_w = \begin{cases} \frac{\sum_{t=1}^{T} S_1^w(t) \cdot S_2^w(t)}{\min\left(\sum_{t=1}^{T} S_1^w(t), \sum_{t=1}^{T} S_2^w(t)\right)} & \text{if } \min\left(\sum_{t=1}^{T} S_1^w(t), \sum_{t=1}^{T} S_2^w(t)\right) > 0 \\ 0 & \text{otherwise} \end{cases}$$

At the worker level, we compute the OC for each worker w by comparing their shift schedules S_1^w and S_2^w across T time periods. The numerator counts overlapping shifts ($\sum_{t=1}^{T} S_1^w(t) \cdot S_2^w(t)$), while the denominator normalizes this by the smaller schedule's total shifts ($\min(\sum S_1^w(t), \sum S_2^w(t))$), ensuring the measure ranges between 0 (no overlap) and 1 (perfect alignment).

A.3.2. Aggregate measure

The overall similarity across N workers:

$$\overline{\mathrm{OC}} = \frac{1}{N} \sum_{w=1}^{N} \mathrm{OC}_w$$

The worker-level average method calculates the OC for each worker individually and then takes the average across all workers. This approach treats each worker equally, measuring how much of each worker's shifts in Schedule 1 overlap with their shifts in Schedule 2, regardless of the total number of shifts they work. In contrast, the whole-schedule method computes a single OC by comparing all shift slots across all workers at once, effectively weighting workers with more shifts more heavily in the final result. The key difference is that the worker-level average ensures each worker contributes equally to the similarity measure, while the whole-schedule method is disproportionately influenced by workers with longer or more frequent shifts.

The benefit of using the worker-level average is that it provides a more equitable assessment of schedule similarity, particularly when workers have varying shift lengths or frequencies. For example, if most workers have minor overlaps but a few workers with many shifts dominate the calculation, the whole-schedule method may overstate overall similarity. The worker-level approach avoids this bias, making it more reliable for scenarios where fairness across workers is important, such as evaluating schedule changes in a team with diverse shift patterns. This method better captures whether schedules are consistently similar across all workers, rather than being skewed by a subset of high-volume workers.

Data availability

I have shared the link to my data.

References

[1] D.L. Kellogg, S. Walczak, Nurse scheduling: from academia to implementation or not? Interfaces 37 (4) (2007) 355–369.

[2] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, L. De Boeck, Personnel scheduling: A literature review, European J. Oper. Res. 226 (3) (2013) 367–385.

[3] P. Smet, B. Bilgin, P. De Causmaecker, G. Vanden Berghe, Modelling and evaluation issues in nurse rostering, Ann. Oper. Res. 218 (2014) 303–326.

[4] E.I. Ásgeirsson, Bridging the gap between self schedules and feasible schedules in staff scheduling, Ann. Oper. Res. 218 (1) (2014) 51–69.

[5] M. Defraeye, I. Van Nieuwenhuyse, Staffing and scheduling under nonstationary demand for service: A literature review, Omega 58 (2016) 4–25.

[6] M. Erhard, J. Schoenfelder, A. Fügener, J.O. Brunner, State of the art in physician scheduling, European J. Oper. Res. (2017).

[7] S. Çakırgil, E. Yücel, G. Kuyzu, An integrated solution approach for multi-objective, multi-skill workforce scheduling and routing problems, Comput. Oper. Res. 118 (2020) 104908.

[8] J.S. Loucks, F.R. Jacobs, Tour scheduling and task assignment of a heterogeneous work force: A heuristic approach, Decis. Sci. 22 (4) (1991) 719–738.

[9] J.M. Tien, A. Kamiyama, On manpower scheduling algorithms, SIAM Rev. 24 (3) (1982) 275–287.

[10] A.T. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, Staff scheduling and rostering: A review of applications, methods and models, European J. Oper. Res. 153 (1) (2004) 3–27.

[11] B. Jaumard, F. Semet, T. Vovor, A generalized linear programming model for nurse scheduling, European J. Oper. Res. 107 (1) (1998) 1–18.

[12] M.J. Brusco, L.W. Jacobs, A simulated annealing approach to the solution of flexible labour scheduling problems, J. Oper. Res. Soc. 44 (12) (1993) 1191–1200.

[13] S. Zolfaghari, A. El-Bouri, B. Namiranian, V. Quan, Heuristics for large scale labour scheduling problems in retail sector, INFOR Inf. Syst. Oper. Res. 45 (3) (2007) 111–122.

[14] L. Gordon, E. Erkut, Improving volunteer scheduling for the Edmonton folk festival, Interfaces 34 (5) (2004) 367–376.

[15] S. Topaloglu, I. Ozkarahan, An implicit goal programming model for the tour scheduling problem considering the employee work preferences, Ann. Oper. Res. 128 (1–4) (2004) 135–158.

[16] S.K. Mirrazavi, H. Beringer, A web-based workforce management system for Sainsburys supermarkets ltd, Ann. Oper. Res. 155 (1) (2007) 437–457.

[17] J.F. Bard, L. Wan, Workforce design with movement restrictions between workstation groups, Manuf. Serv. Oper. Manag. 10 (1) (2008) 24–42.

[18] L. Wan, J. Bard, Weekly staff scheduling with workstation group restrictions, J. Oper. Res. Soc. 58 (8) (2007) 1030–1046.

[19] M. Hojati, A.S. Patil, An integer linear programming-based heuristic for scheduling heterogeneous, part-time service employees, European J. Oper. Res. 209 (1) (2011) 37–50.

[20] E.-R. Omar, T. Garaix, V. Augusto, X. Xie, A stochastic optimization model for shift scheduling in emergency departments, Heal. Care Manag. Sci. 18 (3) (2015) 289–302.

[21] S. Ganguly, S. Lawrence, M. Prather, Emergency department staff planning to improve patient care and reduce costs, Decis. Sci. 45 (1) (2014) 115–145.

[22] E.I. Ásgeirsson, G.L. Sigurdhardóttir, Near-optimal MIP solutions for preference based self-scheduling, Ann. Oper. Res. 239 (1) (2016) 273–293.

[23] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf, W. Slany, The minimum shift design problem, Ann. Oper. Res. 155 (1) (2007) 79–105.

[24] N. Musliu, A. Schaerf, W. Slany, Local search for shift design, European J. Oper. Res. 153 (1) (2004) 51–64.

[25] N. Kyngäs, K. Nurmi, J. Kyngäs, Solving the person-based multitask shift generation problem with breaks, in: 2013 5th International Conference on Modeling, Simulation and Applied Optimization, ICMSAO, IEEE, 2013, pp. 1–8.

[26] W.J. Gutjahr, M.S. Rauner, An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria, Comput. Oper. Res. 34 (3) (2007) 642–666.

[27] M.P. Thiel, Team-oriented airline crew rostering for cockpit personnel, Comput.-Aided Syst. Public Transp. (2008) 91–114.

[28] P. Smet, T. Wauters, M. Mihaylov, G.V. Berghe, The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights, Omega 46 (2014) 64–73.

[29] S.-W. Lin, K.-C. Ying, Minimizing shifts for personnel task scheduling problems: A three-phase algorithm, European J. Oper. Res. 237 (1) (2014) 323–334.

[30] D. Prot, T. Lapègue, O. Bellenguez-Morineau, A two-phase method for the shift design and personnel task scheduling problem with equity objective, Int. J. Prod. Res. 53 (24) (2015) 7286–7298.

[31] P. Brucker, R. Qu, Network flow models for intraday personnel scheduling problems, Ann. Oper. Res. 218 (1) (2014) 107–114.

[32] E. Zamorano, R. Stolletz, Branch-and-price approaches for the multiperiod technician routing and scheduling problem, European J. Oper. Res. 257 (1) (2017) 55–68.

[33] N. Xue, Instances, 2025, https://github.com/Ning-Xue-Nottingham/WorkforceManagement/blob/main/Instances.zip. Irregular personnel scheduling problem data instances.

[34] N. Xue, D. Landa-Silva, I. Triguero, G.P. Figueredo, A genetic algorithm with composite chromosome for shift assignment of part-time employees, in: 2018 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2018, pp. 1–8.

[35] Y. Sun, S. Wang, Y. Shen, X. Li, A.T. Ernst, M. Kirley, Boosting ant colony optimization via solution prediction and machine learning, Comput. Oper. Res. 143 (2022) 105769.

[36] B. Abbasi, T. Babaei, Z. Hosseinifard, K. Smith-Miles, M. Dehghani, Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management, Comput. Oper. Res. 119 (2020) 104941.

[37] J.-Y. Ding, C. Zhang, L. Shen, S. Li, B. Wang, Y. Xu, L. Song, Accelerating primal solution findings for mixed integer programs based on solution prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 1452–1459.

[38] U. Aickelin, K.A. Dowsland, An indirect genetic algorithm for a nurse-scheduling problem, Comput. Oper. Res. 31 (5) (2004) 761–778.

[39] R. Bai, E.K. Burke, G. Kendall, J. Li, B. McCollum, A hybrid evolutionary approach to the nurse rostering problem, IEEE Trans. Evol. Comput. 14 (4) (2010) 580–590.

[40] L. Theis, M. Horn, The Irace Package: Iterated Racing for Automatic Algorithm Configuration, Technical Report 1651, Max Planck Institute for Intelligent Systems, 2016, URL: https://CRAN.R-project.org/package=irace. R package version 1.0.1.

[41] 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, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[42] J.F. Bard, L. Wan, The task assignment problem for unrestricted movement between workstation groups, J. Sched. 9 (4) (2006) 315–341.

[43] M. Bester, I. Nieuwoudt, J.H. Van Vuuren, Finding good nurse duty schedules: a case study, J. Sched. 10 (6) (2007) 387–405.

[44] R. Bhatnagar, V. Saddikutti, A. Rajgopalan, Contingent manpower planning in a high clock speed industry, Int. J. Prod. Res. 45 (9) (2007) 2051–2072.

[45] M. Akbari, M. Zandieh, B. Dorri, Scheduling part-time and mixed-skilled workers to maximize employee satisfaction, Int. J. Adv. Manuf. Technol. 64 (5–8) (2013) 1017–1027.

[46] M.A. Awadallah, A.T. Khader, M.A. Al-Betar, A.L. Bolaji, Nurse rostering using modified harmony search algorithm, in: International Conference on Swarm, Evolutionary, and Memetic Computing, Springer, 2011, pp. 27–37.

[47] J.F. Bard, H.W. Purnomo, Short-term nurse scheduling in response to daily fluctuations in supply and demand, Heal. Care Manag. Sci. 8 (4) (2005) 315–324.

[48] B. Bilgin, P. De Causmaecker, B. Rossie, G.V. Berghe, Local search neighbourhoods for dealing with a novel nurse rostering model, Ann. Oper. Res. 194 (1) (2012) 33–57.

[49] B. Bilgin, P. Demeester, M. Mısır, W. Vancroonenburg, G.V. Berghe, T. Wauters, A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition, in: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, PATAT'10, 2010.

[50] S. Haspeslagh, P. De Causmaecker, A. Schaerf, M. Stølevik, The first international nurse rostering competition 2010, Ann. Oper. Res. 218 (1) (2014) 221–236.

[51] E.K. Burke, T. Curtois, New Computational Results for Nurse Rostering Benchmark Instances, Technical Report. Technical Report, 2011.

[52] R. Cipriano, L. Di Gaspero, A. Dovier, Hybrid approaches for rostering: A case study in the integration of constraint programming and local search, in: International Workshop on Hybrid Metaheuristics, Springer, 2006, pp. 110–123.

[53] K. Ertogral, B. Bamuqabel, Developing staff schedules for a bilingual telecommunication call center with flexible workers, Comput. Ind. Eng. 54 (1) (2008) 118–127.

[54] M. Hadwan, M.B. Ayob, An exploration study of nurse rostering practice at hospital universiti kebangsaan Malaysia, in: Data Mining and Optimization, 2009. DMO'09. 2nd Conference on, IEEE, 2009, pp. 100–107.

[55] S. Helber, K. Henken, Profit-oriented shift scheduling of inbound contact centers with skills-based routing, impatient customers, and retrials, Or Spectr. 32 (1) (2010) 109–134.

[56] H.-T. Lin, Y.-T. Chen, T.-Y. Chou, Y.-C. Liao, Crew rostering with multiple goals: An empirical study, Comput. Ind. Eng. 63 (2) (2012) 483–493.

[57] M. Krishnamoorthy, A.T. Ernst, D. Baatar, Algorithms for large scale shift minimisation personnel task scheduling problems, European J. Oper. Res. 219 (1) (2012) 34–48.

[58] J. Li, E.K. Burke, R. Qu, A pattern recognition based intelligent search method and two assignment problem case studies, Appl. Intell. 36 (2) (2012) 442–453.

[59] Z. Lü, J.-K. Hao, Adaptive neighborhood search for nurse rostering, European J. Oper. Res. 218 (3) (2012) 865–876.

[60] B. Maenhout, M. Vanhoucke, An integrated nurse staffing and scheduling analysis for longer-term nursing staff allocation problems, Omega 41 (2) (2013) 485–499.

[61] K. Nonobe, INRC2010: An approach using a general constraint optimization solver, in: The First International Nurse Rostering Competition, INRC 2010, 2010.

[62] A. Ovchinnikov, J. Milner, Spreadsheet model helps to assign medical residents at the university of Vermont's college of medicine, Interfaces 38 (4) (2008) 311–323.

[63] E. Rönnberg, T. Larsson, Automating the self-scheduling process of nurses in Swedish healthcare: a pilot study, Heal. Care Manag. Sci. 13 (1) (2010) 35–53.

[64] N. Safaei, D. Banjevic, A.K. Jardine, Workforce-constrained maintenance scheduling for military aircraft fleet: a case study, Ann. Oper. Res. 186 (1) (2011) 295–316.

[65] C. Stark, J. Zimmermann, An exact branch-and-price algorithm for workforce scheduling, in: Operations Research Proceedings 2004, Springer, 2005, pp. 207–212.

[66] S. Topaloglu, A multi-objective programming model for scheduling emergency medicine residents, Comput. Ind. Eng. 51 (3) (2006) 375–388.

[67] C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, E. Housos, A systematic two phase approach for the nurse rostering problem, European J. Oper. Res. 219 (2) (2012) 425–433.

[68] G. Post, B. Veltman, Harmonious personnel scheduling, in: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling. PATAT, 2004, pp. 557–559.

[69] P.D. Wright, K.M. Bretthauer, M.J. Côté, Reexamining the nurse scheduling problem: Staffing ratios and nursing shortages, Decis. Sci. 37 (1) (2006) 39–70.

[70] K.-K. Yang, S. Webster, R.A. Ruben, An evaluation of worker cross training and flexible workdays in job shops, Iie Trans. 39 (7) (2007) 735–746.

[71] X. Zhu, H.D. Sherali, Two-stage workforce planning under demand fluctuations and uncertainty, J. Oper. Res. Soc. 60 (1) (2009) 94–103.

[72] A. Syberfeldt, M. Andersson, A. Ng, V. Bengtsson, Multi-objective evolutionary simulation-optimization of personnel scheduling, Int. J. Artif. Intell. Appl. 6 (1) (2015) 41.

[73] R. Bruni, P. Detti, A flexible discrete optimization approach to the physician scheduling problem, Oper. Res. Heal. Care 3 (4) (2014) 191–199.

[74] P. De Bruecker, J. Van den Bergh, J. Beliën, E. Demeulemeester, A model enhancement heuristic for building robust aircraft maintenance personnel rosters with stochastic constraints, European J. Oper. Res. 246 (2) (2015) 661–673.

[75] S. Dahmen, M. Rekik, F. Soumis, An implicit model for multi-activity shift scheduling problems, J. Sched. (2017) 1–20.

[76] J. Guo, J.F. Bard, A column generation-based algorithm for midterm nurse scheduling with specialized constraints, preference considerations, and overtime, Comput. Oper. Res. 138 (2022) 105597.

[77] U. Aickelin, E.K. Burke, J. Li, An evolutionary squeaky wheel optimization approach to personnel scheduling, IEEE Trans. Evol. Comput. 13 (2) (2009) 433–443.

[78] U. Aickelin, P. White, Building better nurse scheduling algorithms, Ann. Oper. Res. 128 (1–4) (2004) 159–177.

[79] S.M. Al-Yakoob, H.D. Sherali, Multiple shift scheduling of hierarchical workforce with multiple work centers, Informatica 18 (3) (2007) 325–342.

[80] M.N. Azaiez, S.S. Al Sharif, A 0-1 goal programming model for nurse scheduling, Comput. Oper. Res. 32 (3) (2005) 491–507.

[81] J.F. Bard, Selecting the appropriate input data set when configuring a permanent workforce, Comput. Ind. Eng. 47 (4) (2004) 371–389.

[82] J.F. Bard, H.W. Purnomo, Preference scheduling for nurses using column generation, European J. Oper. Res. 164 (2) (2005) 510–534.

[83] G. Beddoe, S. Petrovic, J. Li, A hybrid metaheuristic case-based reasoning system for nurse rostering, J. Sched. 12 (2) (2009) 99.

[84] G.R. Beddoe, S. Petrovic, Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering, European J. Oper. Res. 175 (2) (2006) 649–671.

[85] P. Brucker, E.K. Burke, T. Curtois, R. Qu, G.V. Berghe, A shift sequence based approach for nurse scheduling and a new benchmark dataset, J. Heuristics 16 (4) (2010) 559–573.

[86] E.K. Burke, T. Curtois, R. Qu, G.V. Berghe, A scatter search methodology for the nurse rostering problem, J. Oper. Res. Soc. 61 (11) (2010) 1667–1679.

[87] H.A. Eiselt, V. Marianov, Employee positioning and workload allocation, Comput. Oper. Res. 35 (2) (2008) 513–524.

[88] M.D. Goodman, K.A. Dowsland, J.M. Thompson, A grasp-knapsack hybrid for a nurse-scheduling problem, J. Heuristics 15 (4) (2009) 351–379.

[89] M. Günther, V. Nissen, Sub-daily staff scheduling for a logistics service provider, KI-Künstliche Intell. 24 (2) (2010) 105–113.

[90] J. Li, U. Aickelin, The application of Bayesian optimization and classifier systems in nurse scheduling, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 581–590.

[91] J. Li, U. Aickelin, E.K. Burke, A component-based heuristic search method with evolutionary eliminations for hospital personnel scheduling, INFORMS J. Comput. 21 (3) (2009) 468–479.

[92] E. Özcan, Memetic algorithms for nurse rostering, in: International Symposium on Computer and Information Sciences, Springer, 2005, pp. 482–492.

[93] X. Qi, J.F. Bard, Generating labor requirements and rosters for mail handlers using simulation and optimization, Comput. Oper. Res. 33 (9) (2006) 2645–2666.

[94] S.U. Seçkiner, H. Gökçen, M. Kurt, An integer programming model for hierarchical workforce scheduling problem, European J. Oper. Res. 183 (2) (2007) 694–699.

[95] P. Shahnazari-Shahrezaei, R. Tavakkoli-Moghaddam, H. Kazemipoor, Solving a new fuzzy multi-objective model for a multi-skilled manpower scheduling problem by particle swarm optimization and elite tabu search, Int. J. Adv. Manuf. Technol. 64 (9–12) (2013) 1517–1540.

[96] D. Parr, J.M. Thompson, Solving the multi-objective nurse scheduling problem with a weighted cost function, Ann. Oper. Res. 155 (1) (2007) 279–288.

[97] V. Valls, Á. Pérez, S. Quintanilla, Skilled workforce scheduling in service centres, European J. Oper. Res. 193 (3) (2009) 791–804.

[98] C.A. Henao, J.C. Muñoz, J.C. Ferrer, The impact of multi-skilling on personnel scheduling in the service sector: a retail industry case, J. Oper. Res. Soc. 66 (12) (2015) 1949–1959.

[99] A. Parisio, C.N. Jones, A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand, Omega 53 (2015) 97–103.

[100] A.N. Avramidis, W. Chan, M. Gendreau, P. L'ecuyer, O. Pisacane, Optimizing daily agent scheduling in a multiskill call center, European J. Oper. Res. 200 (3) (2010) 822–832.

[101] A.N. Avramidis, W. Chan, P. L'ecuyer, Staffing multi-skill call centers via search methods and a performance approximation, Iie Trans. 41 (6) (2009) 483–497.
[102] A.N. Avramidis, M. Gendreau, P. L'Ecuyer, O. Pisacane, Simulation-based optimization of agent scheduling in multiskill call centers, 2007.
[103] J.O. Brunner, J.F. Bard, R. Kolisch, Midterm scheduling of physicians with flexible shifts using branch and price, IIE Trans. 43 (2) (2010) 84–109.
[104] J.O. Brunner, G.M. Edenharter, Long term staff scheduling of physicians with different experience levels in hospitals using column generation, Heal. Care Manag. Sci. 14 (2) (2011) 189–202.
[105] C. Canon, Personnel Scheduling in the Call Center Industry (Ph.D. thesis), Springer, 2007.
[106] P. Cappanera, G. Gallo, A multicommodity flow approach to the crew rostering problem, Oper. Res. 52 (4) (2004) 583–596.
[107] M.T. Cezik, P. L'Ecuyer, Staffing multiskill call centers via linear programming and simulation, Manag. Sci. 54 (2) (2008) 310–323.
[108] L.-E. Drezet, J.-C. Billaut, A project scheduling problem with labour constraints and time-dependent activities requirements, Int. J. Prod. Econ. 112 (1) (2008) 217–225.
[109] G. Eitzen, D. Panton, G. Mills, Multi-skilled workforce optimisation, Ann. Oper. Res. 127 (1–4) (2004) 359–372.
[110] P. Eveborn, P. Flisberg, M. Rönnqvist, Laps care—an operational system for staff planning of home care, European J. Oper. Res. 171 (3) (2006) 962–976.
[111] P. Eveborn, M. Rönnqvist, H. Einarsdóttir, M. Eklund, K. Lidén, M. Almroth, Operations research improves quality and efficiency in home care, Interfaces 39 (1) (2009) 18–34.
[112] T. Hanne, R. Dornberger, L. Frey, Multiobjective and preference-based decision support for rail crew rostering, in: Evolutionary Computation, 2009. CEC'09. IEEE Congress on, IEEE, 2009, pp. 990–996.
[113] A. Ingolfsson, F. Campello, X. Wu, E. Cabral, Combining integer programming and the randomization method to schedule employees, European J. Oper. Res. 202 (1) (2010) 153–163.
[114] S. Knust, E. Schumacher, Shift scheduling for tank trucks, Omega 39 (5) (2011) 513–521.
[115] M. Sabar, B. Montreuil, J.-M. Frayret, Competency and preference based personnel scheduling in large assembly lines, Int. J. Comput. Integr. Manuf. 21 (4) (2008) 468–479.
[116] P.D. Wright, K.M. Bretthauer, Strategies for addressing the nursing shortage: Coordinated decision making and workforce flexibility, Decis. Sci. 41 (2) (2010) 373–401.
[117] S. Zolfaghari, V. Quan, A. El-Bouri, M. Khashayardoust, Application of a genetic algorithm to staff scheduling in retail sector, Int. J. Ind. Syst. Eng. 5 (1) (2009) 20–47.
[118] P. Kazemian, Y. Dong, T.R. Rohleder, J.E. Helm, M.P. Van Oyen, An IP-based healthcare provider shift design approach to minimize patient handoffs, Heal. Care Manag. Sci. 17 (1) (2014) 1–14.
[119] J.F. Bard, Staff scheduling in high volume service facilities with downgrading, Iie Trans. 36 (10) (2004) 985–997.
[120] S. Bhulai, G. Koole, A. Pot, Simple methods for shift scheduling in multiskill call centers, Manuf. Serv. Oper. Manag. 10 (3) (2008) 411–420.
[121] P.R. Harper, N. Powell, J.E. Williams, Modelling the size and skill-mix of hospital nursing teams, J. Oper. Res. Soc. 61 (5) (2010) 768–779.
[122] P.-S. Chen, W.-T. Huang, G.Y.-H. Chen, J.-F. Dang, E.-C. Yeh, Constructing modified variable neighborhood search approaches to solve a nurse scheduling problem, Int. J. Prod. Res. (2024) 1–19.
[123] H.İ. Koruca, M.S. Emek, E. Gulmez, Development of a new personalized staff-scheduling method with a work-life balance perspective: case of a hospital, Ann. Oper. Res. 328 (1) (2023) 793–820.
[124] R. Chen, D. Gu, C. Liang, L. Jiang, A multi-skilled staff scheduling and team configuration optimisation model for artificial intelligence project portfolio considering competence development and innovation-driven, Int. J. Prod. Res. (2024) 1–30.
[125] T. Zhang, Y. Liu, X. Yang, J. Chen, J. Huang, Home health care routing and scheduling in densely populated communities considering complex human behaviours, Comput. Ind. Eng. 182 (2023) 109332.