Les Cahiers du GERAD ISSN: 0711–2440

Learning to enumerate shifts for large-scale flexible personnel scheduling problems

F. Rastgar-Amini, C. Contardo, G. Desaulniers, M. Gasse

G–2022–29

July 2022

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

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

Learning to enumerate shifts for large-scale flexible personnel scheduling problems

Farin Rastgar-Amini [a, b]

Claudio Contardo [a, c]

Guy Desaulniers [a, b]

Maxime Gasse [a, b, d]

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

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

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

[d] *CERC in Data Science for Real-Time Decision-Making, Montréal (Qc), Canada, H3T 1J4*

farin.rastgar-amini@polymtl.ca
claudio.contardo@concordia.ca
guy.desaulniers@gerad.ca
maxime.gasse@polymtl.ca

July 2022
Les Cahiers du GERAD
G–2022–29

Abstract : Personnel scheduling consists in determining employee work schedules (sequences of work shifts and days off) to cover the demands of multiple jobs over a planning horizon. We consider finding a near-optimal set of personnel schedules via the solution of a generalized set-covering model with side constraints in a flexible context where a large number of potential shifts can be considered as in the retail industry. Commercial solvers applied to this model often require very long computational times for practical problem sizes, and as such rely on enumeration heuristics for filtering non-promising shifts/schedules and, thus, reducing the problem size. We propose deep learning-based heuristics to drive the enumeration of promising potential shifts based on the information collected from previously solved instances. Our models predict a subset of time points at which promising shifts are more likely to either start or end, thus filtering out those that do not start nor end at those time points. Our computational results on real-life instances show that personnel scheduling problems can be solved considerably faster with an acceptable optimality gap if shifts are enumerated according to the time points predicted by our models.

Keywords : Personnel scheduling, work shifts, set-covering model, machine learning, heuristic shift enumeration

Acknowledgements: This work was funded by Kronos Canadian Systems (now part of UKG), Prompt, and the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant # RDC 530544–18. C. Contardo thanks NSERC under Grant # 2020–06311. This financial support is greatly appreciated. The authors are also grateful to the personnel of UKG for describing the problem and providing datasets.

1 Introduction

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

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

In this paper, we consider multiple jobs, mono-job shifts, and a one-week horizon with a minimum number of days off to be assigned to each employee. We model the corresponding flexible personnel scheduling problem as a generalized set-covering model with side constraints. In this model, a binary variable is defined for each shift that can be potentially assigned to each employee on each day. To reduce its size and obtain faster computational times without comprising too much solution quality, we propose to classify the shifts into promising and non-promising ones and keep in the model only the variables associated with the promising shifts. Instead of trying to classify the shifts themselves, we rather focus on predicting the subsets of start and end times of optimal shifts using a machine learning (ML) model, more precisely, a deep learning one. Promising shifts are then identified as those whose start and end times belong to these predicted times.

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

This paper is organized as follows. Section 2 presents a literature review on the subject. Section 3.3 states the problem studied and introduces the MILP model used in the optimization phase. Section 4

describes the methodology and the multiple ML models we consider in this study. We compare these models using a large dataset in Section 5. Finally, a brief conclusion and possible extensions are discussed in Section 6.

2 Literature review

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

2.1 Integer programming models

Personnel scheduling problems were introduced by Edie (1954) to decrease the vehicles' waiting time for service at toll booths with the minimum number of toll collectors. Since then, different types of problems have arisen for various application areas. Among the most popular research areas on personnel scheduling problems, Özder et al. (2020) point to health services, manufactures, and call centers. However, the retail sector has been less explored. Most studies aim to determine schedules for employees at the lowest cost. There are usually three components to the cost of a solution: 1) the cost of assigning employees to jobs; 2) the penalty associated with over-assigning employees to jobs; and 3) the cost resulting from under-assigning employees to jobs, which results in a degradation of the service quality. For generating employee schedules, mixed-integer programming-based methods are the most popular approaches in the scientific literature (Van den Bergh et al., 2013).

Mathematical models for personnel scheduling problems can be divided into either explicit set-covering or implicit formulations. In the explicit models, one integer variable is associated with each shift type where the type of a shift is defined by all its characteristics such as start time, end time, break time, job type, etc. In implicit models, some aspects of the shifts are determined by means of constraints. One example of implicit modeling would be to define shifts solely by their starting times and add additional constraints to the model to limit their lengths (Thompson, 1995). In many cases, the number of decision variables is considerably smaller than for an explicit formulation. According to Thompson (1995), in most cases, it is hard to express the employment cost using an implicit formulation. Explicit models are preferable when the cost of a shift is defined by its start time, end time, length, and placement of the breaks. Furthermore, Quimper and Rousseau (2010) point out that it is not possible to formulate multiple-job problems through implicit modeling.

The first explicit set-covering model for personnel scheduling was proposed by Dantzig (1954) for the problem of assigning employees to toll booths. In this formulation, all feasible working patterns are enumerated for all possible start times, end times, and nesting breaks. Then, one integer variable x_s is associated with each feasible work pattern s. Here, the problem objective is to find the shift assignment of the minimum total cost that covers the demand for open toll booths at each time interval.

Explicit modeling allows one to formulate different aspects of personnel scheduling problems. To give a few examples, explicit formulations have been used to produce employee schedules for nurse scheduling (e.g., Bard and Purnomo, 2005a,b), for tour scheduling (e.g., Brunner and Stolletz, 2014), for personnel scheduling with employee transfers between departments (e.g., Attia et al., 2019; Dahmen et al., 2020), for a multi-job multi-task shift scheduling problem (e.g., Boyer et al., 2014; Restrepo et al., 2018), and for a stochastic personnel scheduling problem with demand uncertainty (e.g., Bürgy et al., 2019).

The set-covering formulation provides the advantage of modeling flexibility. Nevertheless, additional options for placement of breaks, start times, lengths, and end times for shifts result in a larger number of possible shifts and consequently larger models. For example, in the retail industry, shifts

may start and end every quarter of an hour during a day. Enumerating all possible shifts for all available employees and for multiple jobs results in a very large model that requires efficient algorithms to be solved. These recurrent problems need to be solved in a reasonable amount of time frequently by efficient algorithms. Commercial solvers fail to achieve near-optimal solutions in a reasonable time for large-sized problems either formulated explicitly or implicitly. In the literature on personnel scheduling problems, mostly decomposition methods and heuristic algorithms are implemented to solve large-scale personnel scheduling problems (Van den Bergh et al., 2013). However, practical personnel scheduling problems encounter many shift feasibility constraints and many complicated ways to compute a shift's cost. In this case, it becomes difficult to design the pricing problems that are used to generate shifts dynamically in a column generation algorithm. To alleviate this, some authors (see, e.g., Côté et al., 2011; Restrepo et al., 2018) use context-free grammar to model these shift features.

2.2 Heuristics and metaheuristics

To the best of our knowledge, the studies in personnel scheduling that aim at reducing the number of variables in a set-covering model are limited to very early works such as Bechtold and Brusco (1994), Henderson and Berry (1976), and Mabert and Watts (1982). Their methods, called *working set generation*, select a subset of schedules from all the possible work schedules and create only schedule variables for this subset. Different procedures have been developed to determine this subset. The one that seems to be the most efficient has been proposed by Bechtold and Brusco (1994) and builds the work schedule subset in three steps: first, it computes a subset of days-off configurations based on the coverage that they offer; second, it finds for each day a subset of shifts based, again, on the coverage that these shifts offer; and, finally, it enumerates all work schedules resulting from a combination of the selected days-off configurations and shifts. Another study by Brusco and Jacobs (2001) performs experimental tests to find out how the number of required employees is affected if they restrict the shifts' starting periods to only 3, 4, and 5 periods of the day instead of 24 periods. These works cannot be directly applied to our context as they assume that there is a single job to cover and that the employees are available all the time and can, thus, be considered identical.

Most heuristics and metaheuristics, for personnel scheduling problems, are developed to address complexities resulting from numerous and complicated constraints and not from the flexibility in shift enumeration. Such issues arise mostly in the healthcare domain where the preferences of the staff are important but the possible working shifts are very limited (3 to 5 per day). Generally, in these problems, the constraints are categorized into two groups: hard and soft constraints, depending on the legal regulations and personal preferences of the individual institutions and countries. Hard constraints must be met in order to obtain feasible solutions, whereas soft constraints may be violated with penalties. Burke et al. (2004) review the state-of-the-art heuristics and metaheuristics such as simulated annealing, tabu search, and genetic algorithms for nurse rostering problems. Among the more recent research, we can refer to work of Bard and Purnomo (2005b), Burke et al. (2008), Burke et al. (2010), and Brucker et al. (2010).

A few papers (Dahmen and Rekik, 2015; Attia et al., 2019; Dahmen et al., 2020; Hassani et al., 2021) develop heuristics for flexible personnel scheduling problems in the domain of service and retail industry. Attia et al. (2019) and Dahmen et al. (2020) focus on large-scale multi-department problems. In the former work, a large neighborhood search heuristic is proposed, where the neighborhoods are explored using a MILP model. In the latter, a two-stage solution method is devised. In the first stage, an aggregated MILP model that considers a coarser discretization of the time horizon for the demand curves is solved to yield an approximate solution. In the second stage, this solution is disaggregated by solving one or several small MILP models restricted to a small subset of shifts.

Dahmen and Rekik (2015) and Hassani et al. (2021) consider a similar environment to our problem. Assuming that the employees' days off are known, Dahmen and Rekik (2015) introduce a hybrid heuristic that combines tabu search and branch-and-bound to perform a local search. This algorithm requires average computational times exceeding two hours for instances with 20 to 30 employees and 2

to 4 jobs. Very recently, Hassani et al. (2021) develop a local search heuristic, called the Parallel Stimulation of Disruptions heuristic (PSD), that generates and assigns shifts during the solution process. A set of generator decisions (e.g., replace/extend/shorten a shift) forms the basic moves that lead to new feasible and infeasible solutions. PSD can be cast as an iterative ruin and recreate heuristic. At each iteration, it stimulates a well-targeted disruption (shift shortening) before correcting it with a sequence of generator decisions that is computed by a truncated depth-first search. Furthermore, to improve efficiency, the search is performed in parallel by defining at each iteration multiple subproblems, each initiated by a disruption that can be corrected by a selected subset of employees.

2.3 Machine learning application

Recently, there has been an increasing interest within the mathematical optimization community in applying machine learning techniques to improve the performance of MILP solvers. A survey on the contribution of ML to combinatorial optimization problems is presented by Bengio et al. (2021).

Abbasi et al. (2020) summarized the previous works on this subject in the following five avenues:

1. To improve the performance of an optimization solver by predicting if a reformulation is required for the problem before being solved by a MILP solver. For example, Kruber et al. (2017) employed ML to predict whether a Dantzig–Wolfe decomposition is required for solving a MILP problem.
2. To improve the performance of solution methods. For instance, ML enhances the performance of some algorithms by finding good initial feasible solutions (Xavier et al., 2020) and accelerates enumerative algorithms in branching variable selection (Gasse et al., 2019; Lodi and Zarpellon, 2017). ML is also employed to facilitate column selection in column generation algorithms (Morabit et al., 2021) and to select cutting planes (Tang et al., 2020).
3. To reduce the computational complexity of solving optimization models under uncertainty by implementing dimensionality reduction methods (Xu et al., 2016).
4. To generate the solutions (ideally optimal) of a combinatorial optimization problem (Abbasi et al., 2020) or to predict the value of a subset of decision variables (Lodi et al., 2020; Xavier et al., 2020).
5. To predict the objective value (ideally optimal) of a problem (Fischetti and Fraccaro, 2019).

We focus on the work of Lodi et al. (2020) and Xavier et al. (2020) from the fourth category in which we observe some aspects that are common to our research. They studied recurrent problems for which there are minor differences between the instance input data and are solved by a MILP solver. To accelerate the solution time, they used ML methods to get information from previously solved instances and fixed the value of some variables in similar instances to be solved in the future. Subsequently, the computational performance of MILP solvers is improved due to the elimination of some variables and the reduction of the problem size.

Lodi et al. (2020) assume the existence of a reference instance to a combinatorial optimization problem. This instance can be modeled as a MILP model and solved optimally. The problem needs to be solved again if a perturbation happens in the input of the initial instance. A database of resolutions can be built by collecting the changes that have happened so far or that can be simulated a priori. The authors study how information can be retrieved from this database to support the analysis of potentially perturbed instances. To predict whether a change will influence the reference optimal solution and to what degree, they formulate this problem as a supervised learning problem. A binary classifier is trained to predict whether the entire or a part of an optimal solution for the reference instance is applicable to the perturbed instance. If only a part of the reference solution is valid for the new instance, the value of the other variables is predicted by regression. These decisions are translated as additional constraints and imposed to the perturbed problem. The authors employed their approach to the facility location problem and obtained satisfactory solutions in a decreased time.

Xavier et al. (2020) study a class of optimization problems that deal with power-generating unit scheduling. These problems are formulated as MILP models and solved by a MILP solver to determine the schedule and production level of power generators over a planning horizon. To improve the computational performance of the MILP solver, the authors propose three approaches including the prediction of affine subspaces that include the optimal solution, with very high likelihood. Their contributions are inspired by intuitive patterns related to these problems which are not included in the MILP modeling. For example, the operators observed that a specific type of generator was mostly scheduled to operate throughout the day whereas other types of generators were usually assigned to peak demand periods. The authors developed an ML model to predict the value of the binary decision variables by learning from previously solved instances. The output of the ML model determines the value that variables should be fixed at before solving the MILP. To do so, supplementary constraints are added to the MILP model that indicates the value of the variables. Results of this research showed that a substantial size reduction of the problem reduced the complexity and solution time of the MILP solver.

To the best of our knowledge, no research has been carried out on learning from previously solved instances for instance size reduction in the field of personnel scheduling.

3 Problem description and mathematical modeling

We start by describing the flexible personnel scheduling problem studied (Section 3.1). Then, Section 3.2 provides the notation used in the proposed mathematical model, which is presented afterwards in Section 3.3.

3.1 Problem statement

In the retail industry, the personnel scheduling process starts with forecasting the workload approximately two to eight weeks before the beginning of the planning horizon (Bürgy et al., 2019). In fact, the forecasts focus on the expected number of sales/transactions by department, taking into account several factors such as seasonality and promotions. These sales are subsequently converted into a demand in employees per job that can vary over the planning horizon. Several types of jobs may be relevant in a retail context: salespersons, cashiers, clerks, and multiple others. Once these demands are established, personnel schedules can be computed with the goal of covering these demands as best as possible with the available employees. In many organizations, this scheduling process is repeated on a weekly basis.

The flexible personnel scheduling problem to be solved each week can be stated as follows. There is a set of jobs J to be performed by a set of skilled employees E. Among all employees in E, there is only a subset of the employees, denoted E^j, that are qualified to perform job $j \in J$. Employees work mono-job shifts, where a shift is defined by a valid start time (say, at any quarter of an hour) and a valid duration (say, between 4 and 9 hours). Because the demands of different jobs may follow different patterns (for example, some jobs may start at 6:00 AM, while others only at 8:00 AM), the subset of candidate shifts S^j for a job $j \in J$ may differ from one job to another. Also, given the skills of an employee $e \in E$ and their availability over the planning horizon, they can be assigned to a subset of shifts S^e. For each job $j \in J$, a demand curve (a number of employees required as a function of time) to cover at best is provided. Dividing the planning horizon into a set P of disjoint 15-minute periods, this curve indicates the number of employees d_p^j required on the job j at period $p \in P$. We assume that the periods in P are chronologically numbered from 1 to $|P|$ and that the planning horizon is also divided into 7 days and denoted $H = \{1, 2, \ldots, 7\}$.

The demand for employees fluctuates during the time horizon and even during the day. The ideal scheduling for such a problem happens when the number of employees assigned to a job is equal to the job's demand at each period while considering the following working rules: there must be a minimum

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

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

The proposed model (see Section 3.3) is an explicit one that requires the enumeration of all possible shifts. In the retail industry, there is high flexibility in the start time, the end time, and the duration of the shifts. Therefore, the number of possible shifts is very large. For instance, we consider that shifts can start at every quarter of an hour and can last between 3 and 8 hours by steps of 15 minutes: The set of times at which a shift can start and the set of possible shift lengths are shift_starts = {00:00, 00:15, 00:30, ..., 23:45} and shift_lengths = {3h, 3h15m, ..., 8h}, respectively. Note that we assume as in real life that there is no isolated demand spanning less than the minimum shift duration. Therefore, there is no need to consider a shift assigned to a job with a zero demand at the beginning or the end of the shift without any harm to the feasibility of the problem.

3.2 Notation

The notation used in the mathematical model is presented in Tables 1 to 3.

Table 1: Sets and subsets

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

Table 2: Parameters

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

Table 3: Decision variables

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

3.3 Mathematical model

Given the above notation, the flexible personnel scheduling problem can be modeled as the following integer program, which is similar to that of Hassani et al. (2021):

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

$$s.t.: \quad \sum_{s\in S^p\cap S^j} u_{s,j} + \sum_{e\in E_p^j}\sum_{s\in S^e\cap S^p\cap S^j} z_{s,e,j} - \sum_{k\in K} v_{p,j,k} = d_p^j \qquad \forall p\in P, j\in J \tag{2}$$

$$\sum_{j\in J^e}\sum_{s\in S^e\cap S^j\cap S^h} z_{s,e,j} + o_{e,h} = 1 \qquad \forall e\in E, h\in H \tag{3}$$

$$\sum_{j\in J^e}\sum_{s\in S^e\cap S^j} g_s z_{s,e,j} - \sum_{q\in Q} w_{e,q} = 0 \qquad \forall e\in E \tag{4}$$

$$\sum_{h\in H} o_{e,h} \geq n_e^O \qquad \forall e\in E \tag{5}$$

$$\sum_{j\in J^e}\sum_{s\in S^e\cap S^j\cap S^{h+1}} a_s z_{s,e,j} + M o_{e,h+1} - \sum_{j\in J^e}\sum_{s\in S^e\cap S^j\cap S^h} (b_s + 1 + n^R) z_{s,e,j} \geq 0 \qquad \forall e\in E, h\in H\setminus\{7\} \tag{6}$$

$$z_{s,e,j} \in \{0,1\} \qquad \forall j\in J, e\in E^j, s\in S^j\cap S^e \tag{7}$$

$$u_{s,j} \geq 0, \text{ integer} \qquad \forall j\in J, s\in S^j \tag{8}$$

$$v_{p,j,k} \in [0, m_k] \qquad \forall p\in P, j\in J, k\in K \tag{9}$$

$$w_{e,q} \in [0, m_q] \qquad \forall e\in E, q\in Q \tag{10}$$

$$o_{e,h} \in \{0,1\} \qquad \forall e\in E, h\in H \tag{11}$$

The objective function (1) minimizes the sum of the personnel costs, the anonymous shift costs, and the over-covering penalties. Constraints (2) ensure that the demand for each job at each time period is covered by personalized or anonymous shifts and make the calculation of over-coverings possible. Constraints (3) impose that each employee is assigned to a shift or a day off on each day of the planning horizon. The amount of working hours of each employee that falls on each step $q \in Q$ is calculated through the constraints (4). Each employee e receives at least n_e^O days off according to constraints (5). The big-M constraints (6) enforce a minimum rest time of n^R periods between two working shifts that are assigned to the same employee on consecutive days, wherein $M = \max_{s \in S}\{b_s + 1 + n^R\}$. Finally, constraints (7)–(11) restrict the domain of the variables.

4 Methodology

We develop deep learning models to predict a subset of time periods that are likely to be the start or the end of the shifts in an optimal solution of a particular problem. To provide a training dataset, we generate artificial instances from real data and solve the corresponding MILP models (1)–(11) to near optimality (i.e., with an optimality gap tolerance of 0.01%) using a commercial MILP solver. The parameters and optimal solutions of these instances form the training data, which is used to train a deep learning classifier that can predict simultaneously shift start time periods and shift end time periods. Once the potential shift starting and ending periods are predicted for a new instance, shifts are enumerated according to these subsets of periods, and the MILP model with a reduced number of variables is solved.

In this section, we start by presenting the real-life instances that were made available to us by our industrial partner and how we generated additional instances from them (Section 4.1). Then, we present the features of an instance to be considered for prediction and define the output labels (Section 4.2). Next, we discuss the learning strategy (Section 4.3) used to train the ML models which are then described (Section 4.4). Finally, we present the pipeline of the optimization heuristics (Section 4.5).

4.1 Real-life instances and data generation

For our study, 14 real-life problem instances were made available to us by our industrial partner. Each instance provides all employee information and workload data over one week. Table 4 presents the numbers of jobs and employees associated with each instance.

Table 4: Number of jobs and employees in the reference instances and number of generated artificial sub-instances from each instance

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

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

1. Job demand curves: The demand for jobs changes during the day and the patterns of the demand over different days of a week are also different. To generate variations from an original instance, we perturb the demand curves by adding noises to the level of the demand. More precisely, for each job $j \in J$, we pick uniformly at random between 10% and 20% of each day's periods and we

apply a Gaussian noise to its demand as follows: $d_p^j \leftarrow d_p^j + \lfloor \mathtt{x} + 0.5 \rfloor$, where $\mathtt{x}$ is sampled from a normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.2 \times d_p^j$. Note that the demand remains unchanged when it is zero at the beginning or the end of a day.

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

All the remaining data is left unchanged.

In total, 100 artificial instances for each original instance (i.e., 1,400 instances) were generated in this way and solved. To increase the number of data points in the training set and given that the shift start and end times can be selected almost independently from one day to another and from one job to another, we have chosen to decompose each instance and its associated MILP solution into sub-instances and sub-solutions with respect to each job and each day. Each sub-instance is considered in the training dataset, denoted $\mathcal{D}$, as an individual training sample and represents the shift scheduling problem for one job $j \in J$ over one day $h \in H$. Therefore, we extract $|J| \times |H|$ sub-instances from each instance which gives a further boost to the size of training data (a total of 46,200 sub-instances distributed over the original instances as shown in Table 4). Moreover, decomposing the instances per job helps to have a fixed dimension input size for all instances, which can be fed into a standard learning model. Finally, we use 67% of these artificial instances (67% of sub-instances for each original instance) for training and the remaining 33% for validation. Additional instances are also generated for testing purposes as mentioned in Section 5.

4.2 Features and labels

Each data point $i \in \mathcal{D}$ is associated with a job j and a day h and is defined by a set of features and two labels. For the features, we consider two sets that yield two different ML models. In the first set, we only use the information stemming from the demands for job j on day h, and from the qualifications and the availability of the employees. For the second set, we also consider information obtained from the linear relaxation optimal solution of the MILP model (1)–(11).

More precisely, denoting by $\tilde{p}(h)$ the index of the first period of day h and L the number of time periods in a day, the features of a data point i associated with job j and day h, include the two vectors:

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

Beside the feature vectors X_1 and X_2, we can extract information from solving the linear relaxation of the MILP problems similar to Khalil et al. (2016) and Gasse et al. (2019). The second set of features includes two other vectors, denoted X_3 and X_4, that rely on the notation:

$$\nu^j_{\tilde{p}(h)+l-1} = \max_{e \in E^j, s \in S^j} \{z^{R(i)}_{s,e,j} | a_s = \tilde{p}(h) + l - 1\}, \qquad \forall l \in \{1, 2, ..., L\}, \tag{12}$$

$$\mu^j_{\tilde{p}(h)+l-1} = \max_{e \in E^j, s \in S^j} \{z^{R(i)}_{s,e,j} | b_s = \tilde{p}(h) + l - 1\}, \qquad \forall l \in \{1, 2, ..., L\}, \tag{13}$$

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

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

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

4.3 Learning strategy

We frame the problem of predicting the optimal start-of-shift and end-of-shift periods as a supervised multi-label classification problem. Our input space is $\mathcal{X} = \mathbb{R}^{L \times m}$, with either $m = 2$ for simple MILP parameter features or $m = 4$ for the augmented features, as described in Section 4.2. Our output space is $\mathcal{Y} = \{0,1\}^{L \times 2}$, with binary labels that indicate if each time period is start-of-shift and/or end-of-shift. What we aim for is then a mapping from features to labels, $f : \mathcal{X} \to \mathcal{Y}$ (a.k.a., a classifier). Given a distribution of MILP instances $p(X, Y)$ and a distance function $Dist : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, the goal of supervised learning is to find an optimal mapping $f^\star$, which makes the most accurate predictions on average,

$$f^\star = \arg\min_f \mathbb{E}_{X,Y} \left[Dist(Y, f(X)) \right]. \tag{14}$$

In this work we use the popular Hamming loss as our distance function, which is minimized using a simple binary relevance scheme (Zhang et al., 2018). We employ parametric probabilistic models $q_\theta : \mathcal{X} \to [0,1]^{L \times 2}$ to output the probability for each label to take value one, with θ the model parameters that are to be learned. Our training objective is then the binary cross-entropy (BCE) loss between the probabilistic predictions and the true labels. Finally, we achieve (14) via empirical risk minimization from a collection of training data points $\mathcal{D} = \{(X^{(i)}, Y^{(i)})\}_{i=1}^N$, as described in Section 4.1, leading to the following learning problem,

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{N} BCE(Y^{(i)}, q_\theta(X^{(i)})). \tag{15}$$

The individual classification accuracy of each label is then obtained using the binary predictions $f_{\theta^\star}(X) = \lceil q_{\theta^\star}(X) \rceil$. In practice we solve (15) via stochastic gradient descent on the training data set, with early stopping on a validation set to prevent overfitting.

4.4 Machine learning models

We experiment three different model architectures for f_θ, a fully-connected neural network (FCN), a convolutional neural network (CNN) (Fawaz et al., 2019), and a U-shaped encoder-decoder convolutional neural network (UNet) (Ronneberger et al., 2015), inspired from the problem of image segmentation in computer vision (Asgari Taghanaki et al., 2021).

Our FCN network, described in Table 5, simply employs 6 densely connected layers with batch normalization, and a decreasing number of hidden units.

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

Our UNet network, described in Table 7, consists of 4 stacked one-dimensional convolutional encoders, with an increasing number of kernels and down-sampling between each encoder (implemented

using a 2-max-pooling), followed by 4 stacked one-dimensional convolutional decoders with a decreasing number of kernels and up-sampling between each decoder (implemented as transposed convolutions), and with skip-connections between each encoder and decoder of the same level (implemented as layer concatenation).

Table 5: FCN architecture

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

Table 6: CNN architecture

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

4.5 Optimization heuristic pipelines

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

1. For each pair of job $j \in J$ and day $h \in H$, compute $\underline{P}_{j,h}$ and $\overline{P}_{j,h}$ using the trained ML model;
2. Build the MILP model (1)–(11) considering only personalized shift variables $z_{s,e,j}$ such that $s \in S^j \cap S^h$ for some $j \in J$ and $h \in H$, $a_s \in \underline{P}_{j,h}$, and $b_s \in \overline{P}_{j,h}$;
3. Solve the resulting MILP model using a commercial MILP solver.

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

1. Build the MILP model (1)–(11) considering all personalized shift variables $z_{s,e,j}$;
2. Solve the linear relaxation of this model to collect the features in vectors X_3 and X_4;
3. For each pair of job $j \in J$ and day $h \in H$, compute $\underline{P}_{j,h}$ and $\overline{P}_{j,h}$ using the trained ML model;
4. From the MILP model (1)–(11), filter out all personalized shift variables $z_{s,e,j}$ such that $s \in S^j \cap S^h$ for some $j \in J$ and $h \in H$, and at least one of the following two conditions hold: $a_s \notin \underline{P}_{j,h}$ or $b_s \notin \overline{P}_{j,h}$;
5. Solve the resulting MILP model using a commercial MILP solver.

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

Table 7: UNet architecture

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

5 Computational experiments

In this section, we perform computational experiments to evaluate the efficiency of the proposed approaches in various instances. In Section 5.1, we describe the computational environment and the instances used for benchmarking. In Section 5.2, we evaluate the performance of the classifiers. In Section 5.3, we compare the performance of the basic and feature-enhanced optimization heuristics between them and against other heuristics used in practice and in the literature. Finally, in Section 5.4, we assess how robust our approaches are against instances from different distributions.

5.1 Experimental setup

The three proposed ML models (FCN, CNN, UNet) described in Section 4 were implemented in Python 3 using the scikit-learn (Pedregosa et al., 2011) ML library. The libraries of Keras were used to design deep learning models. IBM ILOG CPLEX 20.1.0 was used as the MILP solver. The code responsible for loading the input data, querying the ML models, and constructing the MILP models was also written in Python 3. All MILP problems were solved on a Linux machine equipped with a 12-core (2 threads per core) Intel Core i7 processor clocked at 3.4 GHz with 16 GB RAM.

As mentioned in Section 4.1, we have access to 14 real-life instances (see Table 4) that we kept for our main tests. From these instances, we have generated 1,400 artificial instances that form the training and validation datasets. Moreover, using the same procedure described in Section 4.1, we generated from each original instance 10 additional instances to be used during the testing phase. Finally, to test the robustness of our best solution approaches to larger input data variations, we also generated 10 other artificial instances based on a similar but different distribution. We describe how these instances were generated in Section 5.4.

5.2 Evaluation of the predictors

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

In this section, we evaluate the performance of these three ML models on the test dataset (composed of 140 artificial instances). For the evaluation metric, we use the area under the Receiver Operating Characteristic (ROC) curve (Fawcett, 2006). Figure 1 shows the average of the curves over the 96 periods, for each predictor. A ROC curve depicts the true and false positive rates on the y and x axes, respectively. The point in the upper left corner of the ROC space is where the ROC curve of a perfect predictor would touch. A random predictor would yield a diagonal line from the bottom left corner to the top right corner as shown in Figure 1. To compare the performance of multiple predictors, one can, thus, compare the areas under their ROC curves. Larger areas (close to 1) indicate a better predictor. Thus, we observe in Figure 1 that UNet*, UNet, and CNN* predictors have the highest areas under the ROC curves (approximately 0.947, 0.942, and 0.941), indicating that the other models are outperformed by them.

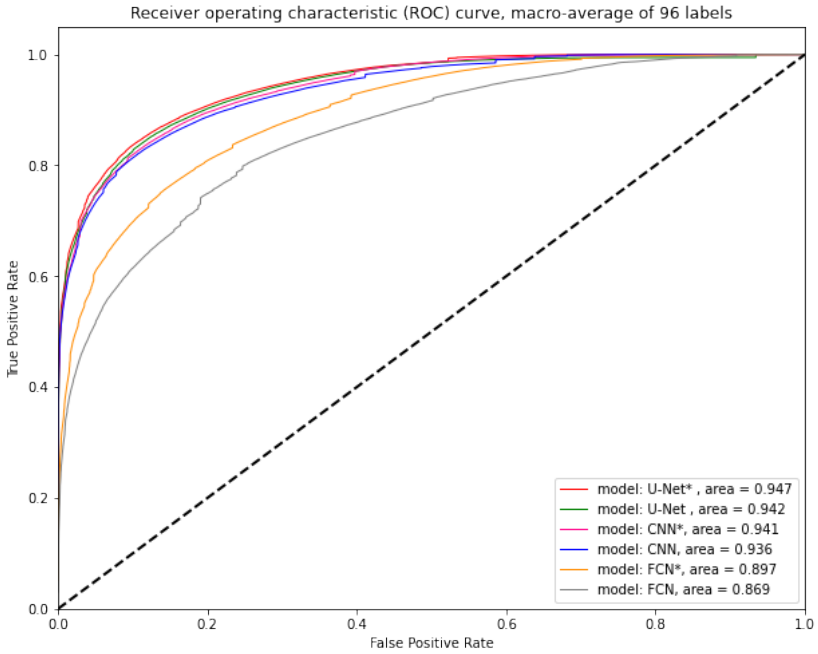
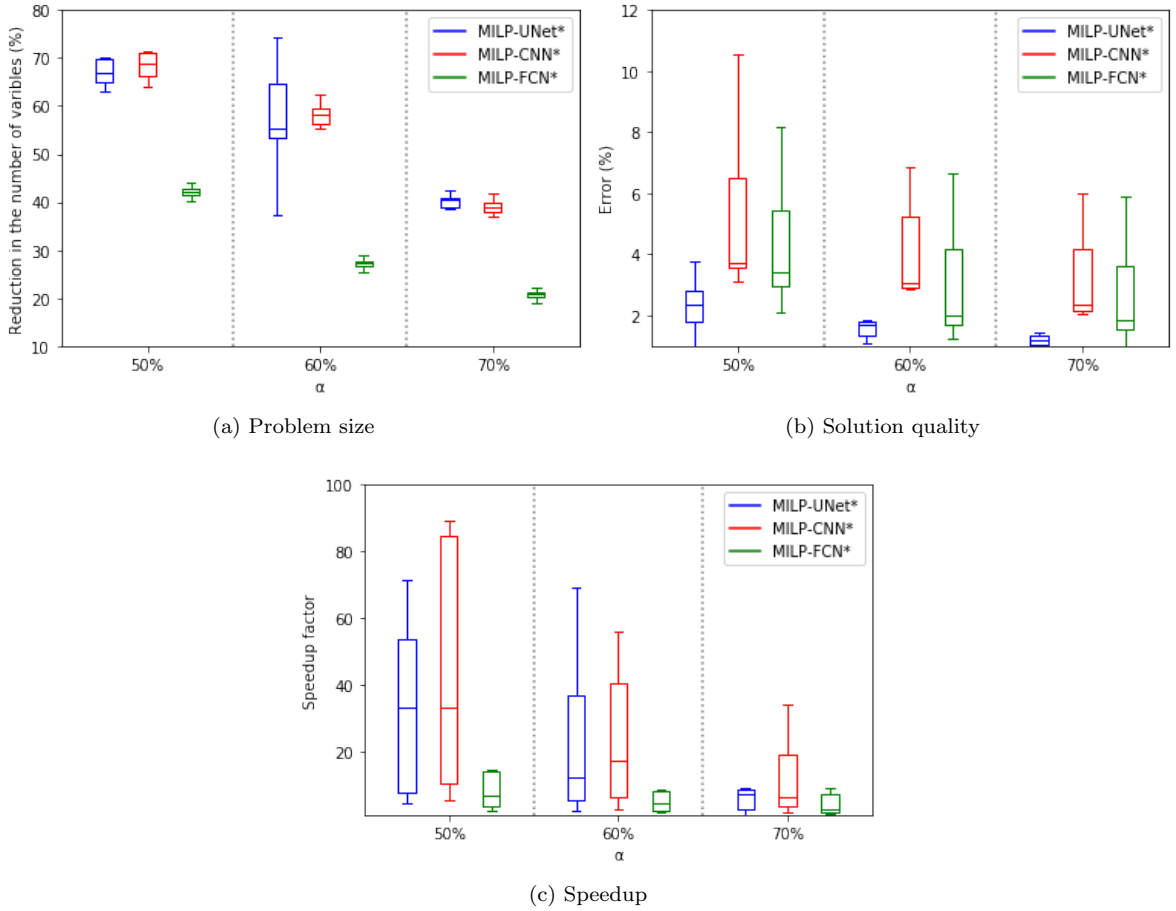


Figure 1: The ROC curves of the six predictors

We have observed that the performance of each ML model may vary greatly depending on the period of the day. Indeed, it is easier to predict when shifts should start at the beginning of a day when the demand is increasing, than in the middle of the day when demand is steady but some shifts must start because previous shifts are ending. On the contrary, for ending periods, the predictors have

a better performance at the end of the day when demand is decreasing than in the middle of the day. To account for this instability in the prediction performance, we propose to not forward the output of the ML models directly to the shift enumeration procedure. Instead, similar to the approach of Xavier et al. (2020), for each period, the output of a predictor is accepted only if both the recall and precision metrics of this predictor on the validation dataset surpass a predetermined threshold (α). Hence, if the performance of a predictor for a specific period is not acceptable, that period is considered a potential shift starting/ending period.

To show how the value of α impacts the performance of the optimization heuristics, we conducted a sensitivity analysis. First, we solved all 140 artificial test instances using an (almost) exact algorithm, i.e., using the MILP model with fully enumerated shifts and an optimality gap tolerance of 0.01%. Then, we solved again these instances with MILP-CNN*, MILP-FCN*, and MILP-UNet* using different values of α, ranging between 50% and 70%. The results of these experiments are summarized in Figure 2. The box plots in Figure 2a provide the percentage of the reduction in the number of variables achieved by the predictors compared to the full model (the higher the better). Figure 2b reports the difference in percentage between the values of the solutions computed by the heuristics and the exact algorithm, called the error (the lower the better). Finally, Figure 2c specifies the speedup in computational time achieved by the heuristics (the higher the better).



(a) Problem size

(b) Solution quality

(c) Speedup

Figure 2: Sensitivity to α

These results show that setting a larger threshold value for the acceptance of the predictor's output in the optimization heuristics yields, as expected, a better solution quality but at the expense of a smaller computational time speedup. MILP-FCN* reduces fewer variables which indicate the recall

or precision of the classifier is less than α for most of the periods and consequently its output is not accepted. More reduction in the number of variables and less error of MILP-UNet* shows its better performance than the other two heuristics, which is consistent with the performance of the predictors observed in Figure 1. We conducted the remaining tests using MILP-UNet* and MILP-UNet. Furthermore, to obtain a good speedup, we set $\alpha = 50\%$.

5.3 Evaluation of the optimization heuristics

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

1. Exact: Exact algorithm where the MILP model (1)–(11) is solved considering all possible shifts, a tolerance on the optimality gap of 0.01%, and a time limit of 20,000 seconds;
2. Tol-1%: Same as Exact except that the optimality gap tolerance is set to 1%;
3. Tol-5%: Same as Exact except that the optimality gap tolerance is set to 5%;
4. LPF: A heuristic that i) solves the linear relaxation of the MILP model with the complete set of shifts, ii) fixes to 0 all $z_{s,e,j}$ variables taking value 0 in the linear relaxation solution, and iii) solves the resulting model with an optimality gap tolerance of 1%;
5. PSD: The parallel stimulation of disruptions heuristic of Hassani et al. (2021);
6. MILP-UNet: The proposed heuristic described in Section 4.5 that filters out shifts by applying a UNet ML model considering only the X_1 and X_2 feature vectors;
7. MILP-UNet*: Same as MILP-UNet except that the X_3 and X_4 feature vectors are also considered in the UNet model;
8. LPF-UNet*: Same as MILP-UNet* except that all $z_{s,e,j}$ variables that take a positive value in the linear relaxation solution cannot be filtered out.

The 14 real-life instances of the reference dataset were solved using each of these algorithms. We report the results of these experiments in Tables 8 to 10. For each instance and each algorithm except the Exact algorithm, Table 8 provides the increase (in percentage) of the cost of the computed solution with respect to the cost of the solution obtained by the Exact algorithm. For each instance and each algorithm, Table 9 indicates the computational time (in seconds) required by each algorithm. For each instance, Table 10 specifies in the second and third columns the total numbers of variables and constraints in the MILP model (1)–(11), when all shifts are considered in the algorithms Exact, Tol-1%, and Tol-5%. The last four columns give the percentage of variables that are removed by the corresponding heuristic. Note that no such statistic is reported for the PSD heuristic as it builds the shifts dynamically and does not rely on a MILP model. In all these tables, averages over all instances are reported in the last row and the best result for each instance and on average is highlighted in bold.

From these results, we make the following observations. The computational times required by the Exact algorithm clearly show the need to develop a more sophisticated exact solution algorithm or to resort to a heuristic for solving the instances in acceptable computational times (say, less than one hour for the largest instances). In particular, we observe that the 20,000-second time limit is reached for 4 of the 14 test instances. A simple heuristic consists in using the same MILP model (with all shift variables) and running the same algorithm but increasing the optimality gap tolerance as in algorithms Tol-1% and Tol-5%. With these heuristics, the average computational time decreases but remains large for some of these instances. Furthermore, the solution costs become less controllable without a larger tolerance, yielding an average cost increase of 1.87% for heuristic Tol-5%.

Table 8: Cost increase (in %) with respect to best solution cost

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

Table 9: Computational times (in seconds)

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

Table 10: MILP model size

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

The heuristic LPF removes the largest number of variables in all instances. It is the fastest for most instances but requires a relatively large amount of time for instance 13. Furthermore, it yields

the largest average cost increase (2.72%). As for PSD, the average quality of its solutions is similar to that of those produced by LPF, but with a larger variance. It is, however, faster on average, especially because it terminates very rapidly for instance 13 with the worst solution among all heuristics.

The last three heuristics apply ML to filter out variables. In particular, the heuristics MILP-UNet and MILP-UNet* are identical except that the latter considers additional features collected from the linear relaxation solution. We observe that this is highly profitable given that it allows removing more variables on average (74.3% versus 71.2%) yielding faster average computational times (174 versus 262 seconds). Furthermore, the average cost increase is reduced by 26%. Finally, LPF-UNet* retains only a few more variables than MILP-UNet* but yields much higher quality solutions (with an average cost increase of only 0.37%) in a larger average computational time (674 versus 174 seconds). This much larger average time is again due to the effort spent solving instance 13.

To summarize, Figure 3 represents the performance of each heuristic with a point in a Cartesian coordinate system where the (broken) x-axis and the y-axis correspond to the average computational time and average cost increase, respectively. This plot allows us to clearly identify two Pareto-optimal algorithms. MILP-UNet* yields the smallest average computational time (174 seconds) and an average cost increase of 1.28%. On the other hand, LPF-UNet* produces the smallest average cost increase (0.37%) in an average computational time of 674 seconds. All other heuristics are dominated by these two algorithms. Therefore, if short computational times are sought, the heuristics MILP-UNet* and LPF-UNet* proposed in this paper and relying on an ML model to reduce the number of variables seem to be the most appropriate.

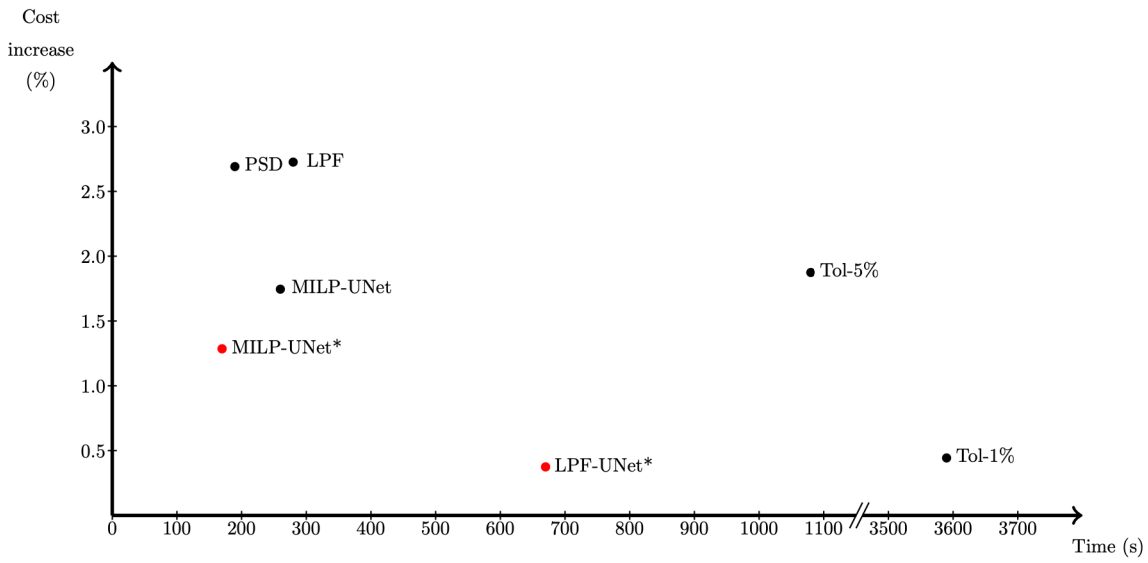


Figure 3: Heuristic performance summary

5.4 Out of distribution results

So far, we performed computational experiments on instances that share the same distribution with the training dataset. For evaluating our best heuristics on problem instances that arise from a different distribution, we have generated a new set of instances (10 instances for each of the 14 original instances) using the procedure described in Section 4.1 except for the following three aspects:

1. Instead of perturbing the demand on only 10% to 20% of the periods, demand noises are applied to all periods (when the job is active);
2. A larger variance is selected for the Gaussian noise, i.e., $\sigma = 0.4 \times d_p^j$ instead of $\sigma = 0.2 \times d_p^j$;

3. A positive noise is always added as follows: $d_p^j \leftarrow d_p^j + |\lceil \mathtt{x} \rceil|$, where $\mathtt{x}$ is sampled from a random distribution $\mathcal{N}(0, \sigma^2)$. This yields an increased demand;
4. To perturb the employee availability, we pick $\lceil 2\% \times |E| \rceil$ employees randomly (instead of $\lceil 1\% \times |E| \rceil$) and make them unavailable throughout the week.

The mean and standard deviation of the demand per job and period (excluding the overnight periods when a job is inactive) in this new dataset and the training dataset is $(2.88, 2.75)$ and $(2.37, 2.07)$, respectively. To determine whether these means are equal or significantly different, we implemented a two-sample Student's t-test, where null hypothesis was "the two means are equal" and the alternative hypothesis was "the two means are different". The p-value obtained from this test is 1.4e-06. Assuming a significance level of 0.01, this p-value indicates that the means are statistically different. Therefore, we conclude that the demand distribution of the new dataset is significantly different from that of the training dataset.

These 140 out-of-distribution test instances are grouped into 14 sets of 10 instances, one for each original instance. They are denoted O1, O2, ..., O14. Each of these instances was solved using three algorithms: Exact, MILP-UNet*, and LPF-UNet*. The results of these experiments are reported in Table 11. Each line indicates the average results obtained for an instance set, namely, the average time (in seconds) required by each algorithm as well as the average increase (in percentage) of the solution cost obtained by the two heuristics compared to the optimal value computed by the Exact algorithm. The last row reports averages over all instances.

From these results, we first observe that these instances are much easier to solve than the original ones. Indeed, the Exact algorithm can solve all of them within the time limit and in an average time of 239 seconds, which is much less than the 8,247 seconds achieved for the original instances. This is due to an increase in the number of under-coverings (as the demand is larger and the employee availability slightly less) that induces the largest proportion of the solution cost. It then becomes easier to find a feasible solution within the 1% tolerance on the optimality gap. Next, we observe that both MILP-UNet* and LPF-UNet* substantially reduce the computational times with respect to the Exact algorithm, by 91% and 88%, respectively. Like with the original instances, LPF-UNet* requires larger computational times than MILP-UNet* but produces better quality solutions. Overall, we can say that the performance of these two heuristics is maintained in these out-of-distribution instances.

Table 11: Results on the out-of-distribution instances

	Exact	MILP-UNet*		LPF-UNet*	
Inst. set	Time (s)	Time (s)	Cost increase (%)	Time (s)	Cost increase (%)
O1	22	4	2.7	6	0.9
O2	13	5	0.9	5	0.1
O3	25	7	1.3	9	0.2
O4	29	8	1.2	10	0.3
O5	46	14	2.0	19	0.2
O6	63	11	2.0	11	0.7
O7	62	17	1.4	24	0.6
O8	353	47	1.1	28	0.9
O9	51	8	1.4	9	0.3
O10	118	14	0.9	28	0.7
O11	144	16	0.9	23	0.4
O12	460	37	1.5	54	0.1
O13	266	39	1.9	62	0.9
O14	1,689	79	2.3	101	0.6
Avg.	239	22	1.54	28	0.49

6 Conclusion

In this work, we have studied a flexible personnel scheduling problem that involves a large number of potential work shifts. To efficiently solve this problem, we have proposed three ML models to predict the starting and ending times of the shifts and use these predictions to filter out from a MILP model shift variables associated with shifts that have a low probability of being selected. Different heuristics based on this approach were devised and two of them provided Pareto-optimal results on 14 real-world test instances. Indeed, the MILP-UNet* heuristic was the fastest with an average speedup factor of 47 compared to an Exact algorithm but yielded an average cost increase of 1.28%. Even when compared to the state-of-the-art neighborhood-based heuristic PSD, MILP-UNet* shows a much more robust behavior being capable of finding better quality solutions in similar computing times. On the other hand, the LPF-UNet* heuristic was able to compute high-quality solutions with an average cost increase of 0.37% but was slower with a speedup factor of 12. These results clearly show the potential to design efficient solution approaches by combining state-of-the-art techniques from ML and mathematical programming.

For future research, we envision using ML to select from the history (previous weeks/months) a small subset of shifts or complete schedules for some employees (e.g., the regular employees) and to restrict the optimization model to the selected shifts/schedules for these employees. This seems an interesting strategy to further speed up the solution process without losing on solution quality.

References

Babak Abbasi, Toktam Babaei, Zahra Hosseinifard, Kate Smith-Miles, and Maryam Dehghani. Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management. Computers & Operations Research, 119:104941, 2020.

Saeid Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen, Julien Cohen-Adad, and Ghassan Hamarneh. Deep semantic segmentation of natural and medical images: a review. Artificial Intelligence Review, 54(1): 137–178, 2021.

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

Jonathan F Bard and Hadi W Purnomo. A column generation-based approach to solve the preference scheduling problem for nurses with downgrading. Socio-Economic Planning Sciences, 39(3):193–213, 2005a.

Jonathan F Bard and Hadi W Purnomo. Preference scheduling for nurses using column generation. European Journal of Operational Research, 164(2):510–534, 2005b.

Stephen E. Bechtold and Michael J. Brusco. Working set generation methods for labor tour scheduling. European Journal of Operational Research, 74(3):540–551, 1994.

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

Vincent Boyer, Bernard Gendron, and Louis-Martin Rousseau. A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem. Journal of Scheduling, 17(2):185–197, 2014.

Peter Brucker, Edmund K Burke, Tim Curtois, Rong Qu, and Greet Vanden Berghe. A shift sequence based approach for nurse scheduling and a new benchmark dataset. Journal of Heuristics, 16(4):559–573, 2010.

Jens O Brunner and Raik Stolletz. Stabilized branch and price with dynamic parameter updating for discontinuous tour scheduling. Computers & Operations Research, 44:137–145, 2014.

Michael J. Brusco and Larry W. Jacobs. Starting-time decisions in labor tour scheduling: An experimental analysis and case study. European Journal of Operational Research, 131:459–475, 2001.

Reinhard Bürgy, Hélène Michon-Lacaze, and Guy Desaulniers. Employee scheduling with short demand perturbations and extensible shifts. Omega, 89:177–192, 2019.

Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. Journal of Scheduling, 7(6):441–499, 2004.

Edmund K. Burke, Timothy Curtois, Gerhard Post, Rong Qu, and Bart Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. European Journal of Operational Research, 188(2):330–341, 2008.

Edmund K. Burke, Jingpeng Li, and Rong Qu. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. European Journal of Operational Research, 203(2):484–493, 2010.

Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Grammar-based integer programming models for multiactivity shift scheduling. Management Science, 57(1):151–163, 2011.

Sana Dahmen and Monia Rekik. Solving multi-activity multi-day shift scheduling problems with a hybrid heuristic. Journal of Scheduling, 18(2):207–223, 2015.

Sana Dahmen, Monia Rekik, François Soumis, and Guy Desaulniers. A two-stage solution approach for personalized multi-department multi-day shift scheduling. European Journal of Operational Research, 280(3): 1051–1063, 2020.

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

Leslie C. Edie. Traffic delays at toll booths. Journal of the Operations Research Society of America, 2(2): 107–138, 1954.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 33(4):917–963, 2019.

Tom Fawcett. An introduction to ROC analysis. Pattern Recognition Letters, 27(8):861–874, 2006.

Martina Fischetti and Marco Fraccaro. Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. Computers & Operations Research, 106:289–297, 2019.

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

Rachid Hassani, Guy Desaulniers, and Issmail El Hallaoui. Parallel stimulation of disruptions for personnel scheduling in a flexible working environment. pages 1–23, January 2021. URL https://www.gerad.ca/en/papers/G-2021-01.

Willie B. Henderson and William L. Berry. Heuristic methods for telephone operator shift scheduling: An experimental analysis. Management Science, 22(12):1372, 1976.

Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

Markus Kruber, Marco E. Lübbecke, and Axel Parmentier. Learning when to use a decomposition. In Integration of AI and OR Techniques in Constraint Programming, pages 202–210. Springer International Publishing, 2017. ISBN 978-3-319-59776-8.

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

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

Vincent A. Mabert and Charles A. Watts. A simulation analysis of tour-shift construction procedures. Management Science, 28(5):520–532, 1982.

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

Emir Hüseyin Özder, Evrencan Özcan, Tamer Eren, et al. A systematic literature review for personnel scheduling problems. International Journal of Information Technology & Decision Making, 19(6):1695–1735, 2020.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 12(85):2825–2830, 2011.

Claude-Guy Quimper and Louis-Martin Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. Journal of Heuristics, 16(3):373–392, 2010.

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

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.

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

Gary M Thompson. Improved implicit optimal modeling of the labor shift scheduling problem. Management Science, 41(4):595–607, 1995.

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

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

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

Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. Frontiers of Computer Science, 12(2):191–202, 2018.