Les Cahiers du GERAD ISSN: 0711–2440

Deep-learning-based partial pricing in a branch-and-price algorithm for personalized crew rostering

F. Quesnel, A. Wu, G. Desaulniers, F. Soumis

G–2020–72

December 2020

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2020
– Bibliothèque et Archives Canada, 2020

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2020
– Library and Archives Canada, 2020

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

Deep-learning-based partial pricing in a branch-and-price algorithm for personalized crew rostering

Frédéric Quesnel [a]
Alice Wu [a]
Guy Desaulniers [a]
François Soumis [a]

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

frederic.quesnel@gerad.ca
alice.wu@gerad.ca
guy.desaulniers@gerad.ca
francois.soumis@gerad.ca

December 2020
Les Cahiers du GERAD
G–2020–72

Abstract : The personalized crew rostering problem (CRP) consists of assigning pairings (sequences of flights, deadheads, connections, and rests, forming one or several days of work) to individual crew members to create a feasible roster that maximizes crew satisfaction. This problem is often solved using a branch-and-price algorithm. In this paper, we propose a partial pricing scheme for the CRP in which the column generation subproblem of each crew member only contains the pairings that are likely to be selected in an optimal or near-optimal solution. The task of selecting which pairings to include in each network is performed by a deep neural network trained on historical data. We test the proposed method on several large instances. Our results show that our method finds solutions of similar quality as that of the classical branch-and-price algorithm in less than half of the computational time.

Acknowledgements: This work was supported financially by the Natural Sciences and Engineering Research Council of Canada and AD OPT under grant no. CRDPJ – 477127–14. The authors are grateful for this support.

1 Introduction

Creating good-quality schedules is of great importance for airlines. Although the main focus of airlines is to minimize employee-related costs, they usually put significant effort in creating schedules that are satisfactory for their crew members, as employee satisfaction has a positive impact on service quality and lowers the turnover rate. However, creating such a schedule is challenging from an optimization standpoint because the instances are generally large (containing several hundreds of crew members and thousands of flights) and because the problems are highly combinatorial since each task can be performed by a large number of crew members. For these reasons, crew schedule optimization is still an active research topic.

Aircrew schedules are usually created according to a two-step procedure: crew pairing followed by crew rostering. The goal of the crew pairing problem (CPP) is to find a set of pairings that covers a set of flights at minimum cost. A pairing is a sequence of flights, deadheads (flights that crew members take as passengers to be relocated), connections, and rests, forming one or multiple days of work for a crew member. A pairing must start and end at the same crew base (airport where crew members live) and comply with airline regulations as well as collective agreements. The crew rostering problem (CRP) use those pairings to create a set of feasible schedules. A crew schedule is a sequence of activities (pairings, vacations, trainings, ...) for a crew member and a given period (typically a month). The set of all crew member schedules form a roster. Rosters must satisfy a set of so-called horizontal and vertical rules (Kohl and Karisch 2004). Horizontal rules are constraints on individual schedules and include rules such as the minimum number of days off and the maximum amount of work allowed in a valid schedule. Vertical rules are constraints on the whole roster. Examples of vertical rules are rules regarding the crew composition (number of crew members and their roles) of each flight, language and qualification constraints, etc.

Schedule feasibility rules are very different for cockpit and cabin crews. For instance, cabin crews are typically qualified to work on multiple aircraft types of the same family whereas pilots and copilots can only operate one aircraft type at any given time. For this reason, crew rostering for pilots and copilots is usually performed independently from cabin crew rostering.

This paper tackles the CRP for cabin crews which is typically solved by a (possibly heuristic) branch-and-price algorithm. Our goal is to speed up this algorithm by proposing a partial pricing strategy that relies on deep learning to reduce the size of the networks underlying the column generation pricing subproblems.

1.1 Related works

Many crew rostering approaches are employed by airlines. In the *bidline* approach, the objective is to create anonymous schedules, called bidlines, that are later assigned to crew members through a bidding system. Examples of papers on the bidline problem can be found in Weir and Johnson (2004), Boubaker et al. (2010), Saddoune et al. (2012). The objective of the bidline problem is usually to minimize the number of bidlines or to create a set of bidlines with a low variance on key performance indicators, such as the total work time and the number of days off, in order to achieve a certain degree of fairness.

Conversely, the goal of *personalized* approaches is to directly assign a feasible schedule to individual crew members. The main advantage of this approach is that crew characteristics such as their availabilities and preferences can be taken into account when building their schedules. For instance, the model of Kasirzadeh et al. (2017) takes into account crew preferences towards specific flights and vacation days to maximize crew satisfaction. Similarly, Zeighami and Soumis (2019) use a similar model in their formulation of the integrated crew pairing and rostering problem. Quesnel et al. (2020) propose a sequential approach that takes into account crew languages in the CPP as well as in the CRP. As for the bidline approach, the objective of the personalized approach can also be to create fair schedules. For instance, the models proposed by Doi et al. (2018) and Sasaki and Nishi (2015) penalize schedules whose working time deviates from the average.

Because there are many CRP variants, several solution methods have been proposed to solve them. Several of those solution methods are population-based metaheuristics (see, for example, Ozdemir and Mohan 2001, Maenhout and Vanhoucke 2010, Deng and Lin 2011, Azadeh et al. 2013). However, except for Maenhout and Vanhoucke (2010) those methods were only tested on very small instances containing fewer than two hundred flights. Furthermore, it is often difficult to evaluate the optimality gap of the solutions due to the nature of metaheuristics. Many other optimization methods have been used to solve variants of the CRP, such as constraint programming (Dawid et al. 2001, for a CRP variant where the only goal is to find a feasible solution), a greedy heuristic (Chen et al. 2011), a multi-start randomized heuristic (de Armas et al. 2017), and a multi-commodity flow problem (Cappanera and Gallo 2004). However, those methods are often tailored to the specific needs of the airline and are unlikely to perform well for different CRP variants.

Finally, the CRP is often solved using column-generation-based methods. The CRP is formulated as either a set-partitioning problem (for cockpit crews) or a generalized set-partitioning problem (for cabin crews) in which each set-partitioning variable corresponds to a feasible schedule variable. The column generation algorithm solves the linear relaxation by iteratively solving a restricted master problem (RMP) and one or several pricing subproblems (SP). The RMP is the linear relaxation restricted to a limited set of schedule variables. The goal of the SPs is to find negative reduced cost schedules to add to the RMP. Those schedules are also called columns because each schedule variable corresponds to a column in the constraint matrix of the CRP. The column generation algorithm iteratively solves the RMP and the SPs until no new schedules are generated by the SPs, at which point the current solution to the linear relaxation is optimal. For large-scale instances, a branching heuristic is then applied to obtain an integer solution. Gamache et al. (1999) propose a branch-and-bound heuristic that fixes entire schedules at each branching node and returns the first integer solution found, without performing backtracking. They are able to obtain good-quality solutions for instances containing up to 5000 pairings in less than 5 hours. Xu et al. (2018) solve the CRP linear relaxation using column generation, and then apply an exact branching procedure to find an integer solution using only the columns generated at the root node. Borndörfer et al. (2006) embed the column generation algorithm into a heuristic branching strategy, thus generating new columns at each branch-and-bound node, to yield a so-called branch-and-price heuristic. Their method finds near-optimal solutions for instances containing up to 14 000 tasks in less than 8 hours. A similar branch-and-price method is used by many authors, such as Kasirzadeh et al. (2017), Zeighami and Soumis (2019), Quesnel et al. (2020). Column-generation-based heuristics offer many advantages compared to other approaches. They are able to tackle larger problems than metaheuristics and typically find solutions with very low optimality gaps in reasonable times. They also easily integrate most horizontal and vertical rules, making these algorithms very versatile.

In a branch-and-price heuristic, there is one SP per crew member in personalized approaches. For the bidline approach, only one SP is necessary because all crew members are considered identical. In both cases, the SPs are formulated as shortest path problems with resource constraints (SPPRC). Typically, solving the SPs accounts for a large fraction of the total computing time. One of the reasons for this is that the SPs are very large because each one considers all pairings. However, each schedule only contains a few pairings (usually less than 10). In many cases, it is possible to identify pairings that are unlikely to be assigned to a given crew member, such as when a pairing conflicts with a crew member's preferred off-period. Conversely, some pairing/crew member combinations can be identified as desirable, for instance when a crew member speaks a language that is required on multiple flights of a pairing.

1.2 Contributions

In this paper, we propose a new branch-and-price algorithm for the personalized CRP in which a set of reduced SPs is solved instead of the traditional SPs. Each reduced SP contains a fraction of all pairings – those that are the most desirable for the corresponding crew member. Thus, using the reduced SPs speeds up the branch-and-price algorithm while sacrificing very little in terms of solution quality.

As mentioned above, it is relatively easy to determine which features indicate good pairing/crew member pairs. Unfortunately, this is not sufficient to create a suitable decision rule that chooses which pairings to include in each crew member's SP. One would also have to determine the relative weight of each feature as well as identify potentially complex interactions between them. For instance, is it better to prioritize languages spoken by relatively few crew members over more widespread languages? What should be done if a pairing is a good match for a crew member with regards to the vertical constraints, but has features disliked by that crew member? Furthermore, some pairings may be undesirable for all crew members, but must still be included in some subproblems.

To overcome this problem, we build a machine learning (ML) model that acts as an expert to determine which pairings are better-suited for each crew member. It is a deep neural network that predicts the likelyhood that a pairing will be assigned to a crew member in an optimal or near-optimal solution. We train this neural network in a supervised fashion, using historical data.

The two main contributions of this paper can, therefore, be summarized as :

- Developing a machine learning model that determines the likelyhood that a pairing will be assigned to a given crew member.
- Proposing a branch-and-price algorithm that leverages this model to rapidly create near-optimal crew schedules.

We test the proposed method on several instances of the personalized CRP for cabin crews, derived from datasets published by Kasirzadeh et al. (2017). Finally, we show that our method is more than 50% faster than the traditional branch-and-price algorithm, and produces solutions of similar quality. We also show that our ML model is more accurate than simple rules designed by an expert.

The remainder of this paper is structured as follows. We present the CRP variant considered in Section 2. The proposed branch-and-price algorithm with partial pricing is presented in Section 3. This algorithm relies on the predictions of the ML model described in Section 4. Computational results are presented in Section 5 and Section 6 briefly concludes.

2 The Crew Rostering Problem

In this section, we present the CRP variant considered in this paper. We first describe the CRP in Section 2.1, and give a mathematical formulation of the problem in Section 2.2

2.1 Problem description

Let $\mathcal{M}$ be a set of crew members, $\mathcal{F}$ a set of flights, and Γ a set of pairings that covers each flight exactly once. Let $\mathcal{F}_p \subseteq \mathcal{F}$ denote the set of flights in pairing $p \in \Gamma$. The CRP for cabin crews is usually separable by aircraft family because cabin crews are qualified to serve on only one aircraft family at any given time and must undergo several weeks of training to change to a new one. Therefore, we assume that aircrafts of the same aircraft family are used for all pairings in Γ. The goal of the CRP is to assign a legal schedule to each crew member, forming a valid roster. We consider the following schedule legality (horizontal) rules :

- A schedule must contain at most $\bar{T}^{SCHED}$ minutes of work.
- There must be at least $\bar{T}^{INTER}$ minutes between two consecutive pairings.
- There is at most $\bar{D}^{CONSEC}$ consecutive days of work.
- A schedule must contain at least $\bar{D}^{OFF}$ days off.
- If a crew member has assigned activities, such as scheduled vacations or training days, those activities must be included in his or her schedule.

In our tests we used $\bar{T}^{SCHED} = 5100$, $\bar{T}^{INTER} = 720$, $\bar{D}^{CONSEC} = 6$, and $\bar{D}^{OFF} = 10$.

We consider two types of vertical rules. The first type requires that an appropriate number of crews must be assigned to each flight, and hence, to each pairing. Let a_p denote the number of crews required to work on pairing p (in our tests, $a_p = 5\ \forall p \in \Gamma$). It is sometimes impossible to assign a sufficient number of crews on all pairings. In that case, the uncovered pairings are left as *open time* to be manually assigned to reserve crews at a later date by the planner. Let ρ_p^P denote the penalty incurred for leaving pairing $p \in \Gamma$ as open time.

The second type of vertical rules consists of language constraints, requiring the crew composition of some flights to include at least one crew with some language qualification. For instance, a flight going to Barcelona might require at least one Spanish-speaking crew. Language constraints are important for airlines because they help improve customer satisfaction. Some language constraints are also imposed by law in some countries. Let $\mathcal{L}$ be the set of languages, and let $\mathcal{L}_f^F$ be the set of language qualifications required among the crew operating on flight $f \in \mathcal{F}$. Since each flight appears in exactly one pairing the flight language constraints can be aggregated into pairing language constraints. Let $\mathcal{L}_p^P = \bigcup_{f \in \mathcal{F}_p} \mathcal{L}_f^F$ denote the set of language requirements for pairing $p \in \Omega$.

It is often impossible to satisfy all language constraints. In that case, the planner would either manually adjust the language requirement of some problematic flights, or assign those flights to crew members who possess the required language qualifications without having officially declared them (sometimes, crew members choose to withhold language qualifications to have more varied schedules). For this reason, language constraints are modeled as *soft* constraints, that can be violated by incurring a penalty in the objective function. Let ρ_{lp}^L denotes the penalty for violating the language constraint for language $l \in \mathcal{L}_p^P$ and pairing $p \in \Gamma$.

The CRP aims at maximizing the total crew satisfaction. We assume crew preferences are known beforehand. Crew members can usually express their preferences in an online portal, by putting weights on a number of schedule features. We consider two types of preferences: pairing preferences and off-period preferences. Pairing preferences include all features related to pairing themselves, such as their destinations, length, date of beginning/end, ... An off-period preference is a preference towards an off-period consisting of one or several consecutive days off. Let s_{pm}^P denote the total weight crew member $m \in \mathcal{M}$ put on pairing $p \in \Gamma$. This weight can be zero if m expresses no preference towards p. Let $\mathcal{O}_m$ be the set of preferred off-periods for m, and let s_i^O be the weight m puts on off-period $i \in \mathcal{O}_m$.

Let Ω be the set of all feasible schedules and let $\Omega_m \subseteq \Omega$ be the set of all feasible schedules for crew member $m \in \mathcal{M}$. Let $\Gamma_s \subseteq \Gamma$ be the set of pairings in schedule $s \in \Omega$. Let u_{ps} be a constant taking value 1 if schedule $s \in \Omega$ contains pairing $p \in \Gamma$. Similarly, let v_{os} be a constant taking value 1 if schedule $s \in \Omega_m, m \in \mathcal{M}$ contains off-period preference $o \in \mathcal{O}_m$. The satisfaction crew member $m \in \mathcal{M}$ derives from schedule $s \in \Omega_m$, denoted c_s, is given by :

$$c_s = \sum_{p \in \Gamma} u_{ps} s_{pm}^P + \sum_{i \in \mathcal{O}_m} v_{os} s_i^O$$

2.2 Mathematical formulation

Let x_s be a variable equal to 1 if schedule $s \in \Omega$ is selected and 0 otherwise. Let $\epsilon_{lp}^L, l \in \mathcal{L}_p$ be a slack variable for the language constraint for language l and pairing p and let variable ϵ_p^P indicates the number of missing crew members for pairing p. Let $m_s \in \mathcal{M}$ and $\Gamma_s \subseteq \Gamma$ denote the crew member and the set of pairings associated with schedule s, respectively. Let $\mathcal{L}_m^M$ be the languages spoken by crew member $m \in \mathcal{M}$. The CRP is formulated as :

$$\max_{s \in \Omega} \sum_{s \in \Omega} c_s x_s - \sum_{p \in \Gamma} \sum_{l \in \mathcal{L}_p} \rho_{lp}^L \epsilon_{lp}^L - \sum_{p \in \Gamma} \rho_p^P \epsilon_p^P \tag{1}$$

s.t.

$$\sum_{s \in \Omega_m} x_s = 1 \qquad \forall m \in \mathcal{M} \tag{2}$$

$$\sum_{s\in\Omega} u_{ps}x_s + \epsilon_p^P = a_p \qquad \forall p \in \Gamma \tag{3}$$

$$\sum_{s\in\Omega|p\in\Gamma_s, l\in\mathcal{L}_{m_s}^M} x_s + \epsilon_{lp}^L = 1 \qquad \forall p \in \Gamma, l \in \mathcal{L}_p \tag{4}$$

$$\epsilon_p^P \in \mathbb{Z}^* \qquad \forall p \in \Gamma \tag{5}$$

$$\epsilon_{lp}^L \in \{0,1\} \qquad \forall p \in \Gamma, l \in \mathcal{L}_p \tag{6}$$

$$x_s \in \{0,1\} \qquad \forall s \in \Omega \tag{7}$$

Objective function (1) maximizes the total crew satisfaction minus the sum of the penalties for violating constraints (3) and (4). Constraints (2) ensure exactly one schedule is selected for each crew member and constraints (3) ensure that each pairing is covered by a sufficient number of crew members. Constraints (4) are the language constraints. Constraints (5)–(7) impose a binary value on the x and ϵ^L variables, and a nonnegative integer value on the ϵ^P variables.

3 Solution method

We solve the CRP using a heuristic branch-and-price method that uses a set of reduced SPs to converge rapidly towards a near-optimal solution. We describe the branch-and-price method in Section 3.1 and the reduced SPs in Section 3.2. The particularity of the reduced SPs is that each one contains only a subset of all pairings. The procedure used to select which pairings are included in each SP is described in Section 3.3.

3.1 Branch-and-price

We use a branch-and-price heuristic to solve problem (1)–(7). The linear relaxation of the problem is solved using column generation. The RMP corresponds to (1)–(4) (plus nonnegativity constraints on all variables) in which Ω is replaced by $\Omega' \subseteq \Omega$, the set of schedules generated since the beginning of the column generation algorithm. The RMP is solved using a standard linear programming technique. There is one SP for each crew member. The goal of the SP for crew member $m \in \mathcal{M}$ is to find the least reduced-cost schedule for m. If this schedule has a negative reduced cost, it is added to Ω'. Solving a subproblem often yields several suboptimal negative reduced-cost columns, which can also be added to Ω'. It would be inefficient to solve all SPs at each column generation. Instead, SPs are solved sequentially until SP^{MIN} of them are successful (i.e. they return at least one negative reduced-cost column). Note that all SPs are solved in the last column generation iteration to prove that no negative reduced-cost column remains.

The solution returned by the column generation algorithm is usually fractional. We apply a heuristic branching procedure to obtain an integer solution. At each node of the branch-and-bound tree, a linear relaxation is solved using the column generation algorithm. The branching decisions consist of assigning one or several pairings to the schedules of one or several crew members. Let (p, m) denote the assignment of pairing $p \in \Gamma$ to the schedule of crew member $m \in \mathcal{M}$. Let x_s^{*n} denote the value of x_s in the linear relaxation optimal solution at node n and let

$$\xi_{pm}^n = \sum_{s\in\Omega^m} u_{ps}x_s^{*n}$$

be the value of assignment (p, m) in this solution. Let Ξ^n be the set of assignments (p, m) for which ξ_{pm}^n is fractional. Each branching decision consists of imposing one or several assignments of Ξ^n. Let $A_n \subseteq \Xi^n$ denote the selected assignments in the branching decision at node n. Those assignments are selected by decreasing value of ξ_{pm}^n, until either :

- $|A_n| = ASSIGN\_MAX$.

  or

- $\sum_{(p,m)\in A_n} \left(1 - \xi^n_{pm}\right) \geq SUM\_COMP\_MAX$

Where $ASSIGN\_MAX$ denotes the maximum number of assignments that may be included in any branching decision and SUM_COMP_MAX is the maximum value of the *sum of complements* of the fixed assignment. Assignments with lower fractional values are more likely to lead to poor branching decisions, and hence to poor solutions. The sum of complement criterion, therefore, limits the amount of risk taken in any branching decision. Note that some branching decisions may lead to an infeasible solution, for instance, if two conflicting pairings are assigned to the same crew member. It is straightforward to filter out these assignments. The branch-and-bound tree is explored in a diving fashion, and no backtracking is performed. The algorithm stops when solving the linear relaxation at a node yields an integer solution, and returns this solution.

3.2 Subproblems

In traditional branch-and-price algorithms, the SP for crew member $m \in \mathcal{M}$ considers all pairings when searching for a negative reduced-cost schedule. This is inefficient because many of those pairings can be identified as undesirable for m *a priori*. By contrast, our method includes only the pairings most likely to be assigned to m in the final roster. Let $\Gamma_{mn} \subseteq \Gamma$ be the set of pairings included in m's SP at node n of the branch-and-bound tree. We explain the procedure used to select the pairings of Γ_{mn} in Section 3.3. We formulate the SPs as SPPRCs on acyclic networks. The network corresponding to the SP of crew member m and node n is depicted in Figure 1. It contains 5 types of nodes and 7 types of arcs. It contains the following nodes :

- A *source* node and a *sink* node.
- For each pairing in Γ_{mn}, a *beginning_of_pairing* node and an *end_of_pairing* node.
- A *midnight* node for each day of the horizon. Let $\mathcal{D}$ be the set of days in the horizon, numbered from 1 to —D—. The midnight node for day $d \in \mathcal{D}$ represents the first minute of that day (00:00).

A *beginning_of_schedule* arc connects the source to the midnight node of the first day. Similarly, an *end_of_schedule* arc connects the midnight node of the last day to the sink. For each pairing of Γ_{mn} ending on the last day of the horizon, an end_of_schedule arc also connects the pairing's end_of_pairing node to the sink. A *day_off* arc connects the midnight node of day d to the midnight node of day $d+1$ if and only if $d, d+1 \in \mathcal{D}$. A *pairing arc* connects each beginning_of_pairing node to its corresponding end_of_pairing node. Let p be a pairing beginning on day d_1 and ending on day d_2. A *beginning_of_pairing* arc connects midnight node d_1 to the beginning_of_pairing node of p and an *end_of_pairing* arc connects the end_of_pairing node of p to midnight node $d_2 + 2$. Note that the end_of_pairing arc includes a day off following the end_of_pairing p. A *direct_pairing_connection* arc connects the end_of_pairing node of pairing p to the beginning node of pairing q if the interval between p and q is longer than $\bar{T}^{INTER}$ but does not contain a full day off (from midnight to midnight).

Addidional arcs are used to represent preferred off periods. Let $[d_1, d_2] \in \mathcal{O}^m$, be a preferred off-period for m. A *preferred_off_period* arc connects midnight node d_1 to midnight node $d_2 + 1$. Let p be a pairing ending on day $d_1 - 1$. An end_of_pairing_followed_by_preferred_off_periodarc connects the end of pairing node of p to midnight node $d_2 + 1$ or to the sink if d_2 is the last day of the horizon.

Resource constraints are used to enforce the horizontal rules that cannot directly be modeled by arcs. A resource is a commodity that is consumed on arcs and bounded on nodes. For instance, we enforce the maximum-schedule-work-time rule using the *maximum schedule work time* resource. Le δ_p denote the duration of pairing $p \in \Gamma$, in minutes. The maximum schedule work time resource is initialized at 0 at the source and δ_p units of it are consumed when traversing the pairing arc of p. The value of the resource has an upper bound of $\bar{T}^{SCHED}$ at each node of the network. Similarly, one resource is used to enforce the maximum-consecutive-days-of-work rule and another resource enforces the minimum-number-of-days-off rule.

The SPs are solved using a labeling algorithm (see Irnich and Desaulniers 2005). This algorithm iteratively extends *labels* (arrays containing the reduced cost and resource consumptions of partial paths, that are associated with a node) from the source throughout the network. Dominance rules are applied to remove inefficient (non-Pareto optimal) labels at each node.

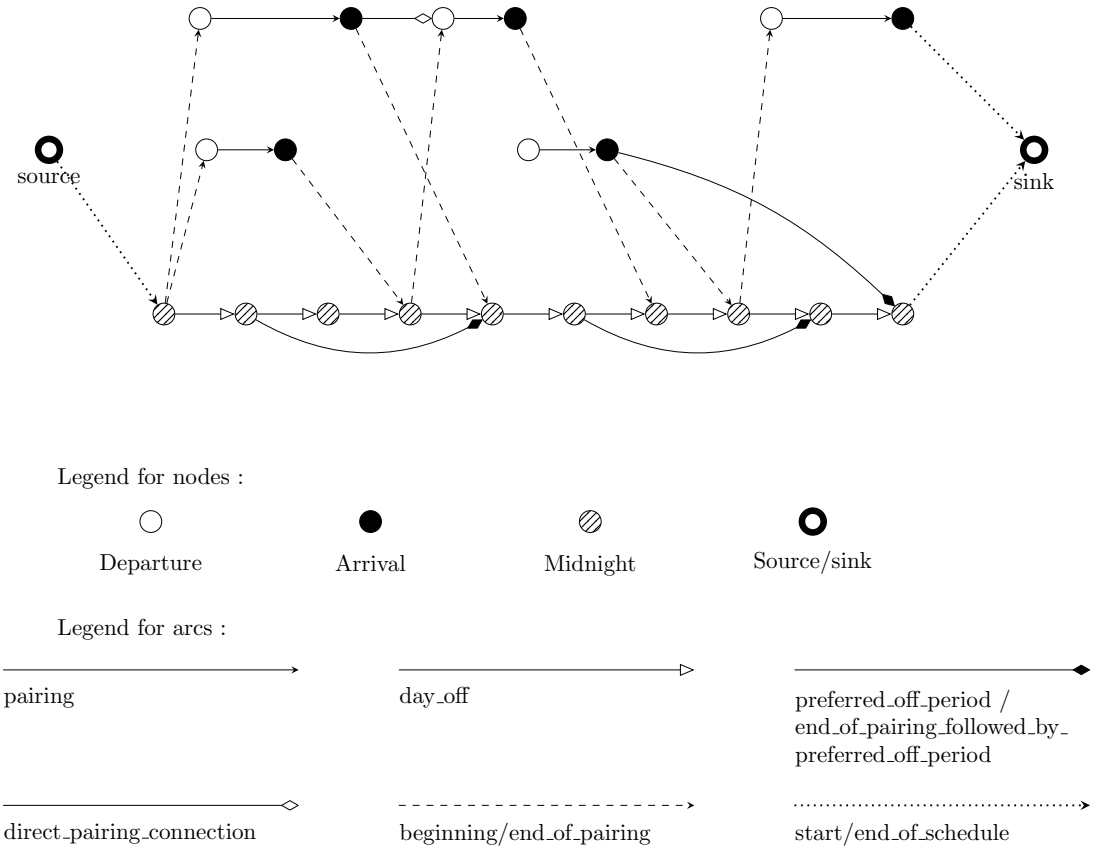

Figure 1: Network for the subproblems

3.3 Selecting Γ_{mn}

Let Γ^m be the set of feasible pairings for crew member $m \in \mathcal{M}$ (i.e. those that do not conflict with any of m's scheduled activities). As mentioned above we restrict the set of pairings used in the SPs to those in $\Gamma_{mn} \subseteq \Gamma^m$, the pairings most likely to be selected in a near-optimal solution. We abusively refer to $|\Gamma_{mn}|$ as the *size* of the SP of m at branch-and-bound node n because it is directly linked to the number of arcs in the SP's network. Let the parameter $S^{MAX} \in [0, 1]$ define the maximum subproblem size. Specifically, Γ_{mn} contains at most $S^{MAX}|\Gamma|$ pairings.

Set Γ_{mn} is constructed as follows. Let P_{mn} be the set of pairings assigned to m in branching decisions before node n. Those pairings are always included in Γ_{mn}, otherwise m's SP would become infeasible. Conversely, let C_{mn} be the set of pairings conflicting with at least one pairing of P_{mn}. Those pairings are always excluded from Γ_{mn} to enforce the branching decisions.

The remaining pairings of Γ_{mn} are selected as follows. Let $\Theta_m = (p_1^m, p_2^m, ..., p_{|\Gamma^m|}^m)$ be an ordering of the pairings of Γ^m such that $i < j$ indicates that $p_i^m \in \Gamma^m$ is more likely to be assigned to m than $p_j^m \in \Gamma^m$. Θ_m is obtained using the ML model described in Section 4. Let $\Theta_{mn} = \Theta_m \backslash \{P_{mn} \cup C_{mn}\}$, with the ordering preserved. The first $\min\{S^{MAX}|\Gamma| - |P_{mn}|, |\Theta_{mn}|\}$ pairings of Θ_{mn} are added to Γ_{mn}.

As more pairings are fixed for m, the cardinalities of P_{mn} and C_{mn} increase. $|C_{mn}|$ typically increases faster than $|P_{mn}|$ because each pairing generally conflicts with several other pairings. Each

time a branching decision assigns a pairing to m's schedule, pairings with a higher rank in Θ_{mn} are included in Γ_{mn}. This allows the algorithm to recover from ranking mistakes made by the ML algorithm.

A possible variant of the proposed method would be to dynamically adjust the size of the SPs. Starting with small SPs, their size would be increased when they can no longer find new columns. We tested several versions of such a method in a preliminary phase but found them less efficient than the one presented here. This is because the benefits of solving smaller SPs at the beginning of the algorithms are outweighed by the disadvantages of solving the same SPs at multiple sizes in the later stages.

An improvement of the method would be to exclude pairing p from all SPs if it has already been assigned to a_p crew members. However, this improvement is independent of the contributions of this paper, so we chose not to implement it in order to fairly compare our algorithm with the standard branch-and-price algorithm. Note that the speed-up resulting from this improvement would likely be negligible because the pairing exclusion condition occurs rarely and near the end of the algorithm.

4 Machine learning model

As mentioned in the previous section, we use a machine learning model to create Θ_m. Our ML model is a deep neural network trained in a supervised training framework using data from several CRP solutions. We describe the data in Section 4.1 and the features of the model in Section 4.2. We present the architecture of the neural network in Section 4.3.

4.1 Data

We train our model on CRP solutions of instances derived from the seven datasets published by Kasirzadeh et al. (2017). Each dataset contains flight data from a major North-American airline for one month and a single aircraft type. The flights of each dataset cover 3 bases and between 26 and 54 airports.

We added artificial language constraints to those datasets. Each instance contains 15 or 16 languages and each language is linked to one or several airports or bases. The airports for each language were chosen to closely imitate the language distribution of an international airline. Let $\mathcal{L}_a^A$ be the set of languages for airport or base a. For a flight $f \in \mathcal{F}$ going from airport or base a to airport or base b, we add language constraints requiring that each language of $\mathcal{L}_a^A \cup \mathcal{L}_b^A$ is spoken by at least one member of the crew operating f.

Crew data was generated using a random procedure similar to the one used in Quesnel et al. (2020). Each crew member is given the following attributes :

- A set of spoken languages (between 1 and 4).
- A set of weighted pairing preferences.
- A set of weighted off-period preferences. Each off-period preference is composed of 3 consecutive days off.

A small fraction of the crews (between 5% and 15%) are also given a scheduled vacation of seven consecutive days. Our procedure ensures that the distribution of languages among crew members closely matches that of an international airline based in North America. One language is spoken by all employees and required on all flights (i.e. English) and another language is spoken by a majority of crew members ($\approx$ 65%). A few other languages are spoken by a significant fraction of the crews (10–25%). The other languages are relatively rare among the crew members and are required for a small number of flights. We also ensure the number of bilingual, trilingual, and quadrilingual crews is similar to that of real-life instances.

In practice, airlines have access to several historical CRP instances and solutions. Indeed, the process of creating the next month's roster often involves solving the CRP multiple times, as the planner adjusts the parameters of the software, fixes parts of the solution, etc. Airlines also have access to the CRP solution for the past few months. Those instances are usually similar because the flight schedule and the crew members do not change a lot from one month to another. We imitate this by generating several instances for each dataset and several solutions for each instance. Different instances of the same dataset have the same flights and language constraints, but different sets of crew members. Note that our instances are more dissimilar than they typically are in airlines because each of our instances has a different crew set, whereas crew turnover is relatively slow in airlines. For this reason, the prediction problem is harder in our case than it would be in real life.

We applied the following procedure to generate several CRP solutions for each instance. We first solved the CPP for each instance using the branch-and-price algorithm proposed by Quesnel et al. (2020). From this CPP solution, we extracted the pairing associated with the base containing the largest number of crew members and solved the CRP for this base (recall that the CRP is separable by base, whereas the CPP is not), using a branch-and-price algorithm equivalent to the one described in Section 3, with $S^{MAX} = 1$ (i.e. all pairings are included in all SPs). Note that the CPP often assigns all flights requiring a given language to pairings associated with bases other than the selected one. For this reason, instances of the same dataset may have different numbers of languages. We solved the CRP multiple times for each instance to obtain multiple solutions for each instance. In order to obtain different solutions, we shuffled the order in which the employees appear in the input file before each run, which changes the order in which the rows are added to the RMP and the initial order in which the SPs are solved. Preliminary tests showed that this is sufficient to obtain largely different solutions.

The instance characteristics for each dataset are given in Table 1. The instances of datasets 1–3 are significantly smaller than that of datasets 4–7, and are therefore called small instances. The instances of datasets 4–7 are called large instances. Note that we created more instances and more solutions for datasets 1 and 3. Those instances were used in a preliminary phase when the amount of data required to obtain accurate predictions was still unknown.

Table 1: Characteristics of the instances

Dataset	Nb. of instances	Nb. solutions per instance	Avg. nb. of languages	Nb. crew members per instance	Avg. nb. of pairings
1	30	30	11.37	69	108
2	15	5	5.67	51	171
3	30	30	11.63	123	268
4	15	5	11.80	274	479
5	15	5	10.13	526	464
6	15	5	12.60	675	717
7	15	5	11.07	735	823

4.2 Features

Labels and features are extracted from the solutions of each instance. We create one entry for each valid pairing/crew member pair. Let E_{mp} denote the entry for crew member m and pairing p. E_{mp} contains the following features :

1. For each language $l \in \mathcal{L}$:
   a. A binary feature indicating whether p requires an l-speaking crew member, denoted $f^{(a.)}_{lpm}$.
   b. A binary feature indicating whether m speaks l, denoted $f^{(b.)}_{lpm}$.
   c. A numerical feature corresponding to the frequency of l amongst pairings, denoted $f^{(c.)}_{lpm}$.
   d. A numerical feature corresponding to the frequency of l amongst crew members, $f^{(d.)}_{lpm}$.

e. A numerical feature corresponding to the *language compatibility score* between p and m for language l, defined as $f^{(e.)}_{lpm} = (1 - f^{(c.)}_{lpm}) \times (1 - f^{(d.)}_{lpm})$ if $l \in \mathcal{L}^M_m$ and $l \in \mathcal{L}^P_p$, and 0 otherwise. Note that this feature has a higher value for scarcer language.

2. A numerical feature called *pairing satisfaction score*, given by

$$f^{(2)}_{pm} = \frac{s^P_{pm}}{\sum_{p' \in \Gamma} s^P_{p'm}}$$

This feature indicates the fraction of m's theoretical maximum pairing satisfaction that is achieved by assigning him or her pairing p. We use $f^{(2)}_{pm}$ rather than s^P_{pm} so that the value of the feature is in the interval $[0, 1]$.

3. A numerical feature called *preferred off-period conflict score*. Let c_{op} be a constant equal to 1 if preferred vacation $o \in \mathcal{O}^m$ conflicts with p. The preferred off-period conflict score is given by

$$f^{(3)}_{pm} = \frac{\sum_{o \in \mathcal{O}^m} c_{op} s^O_o}{\sum_{o \in \mathcal{O}^m} s^O_o}$$

This feature indicates the fraction of m's theoretical maximum off-period satisfaction that is precluded if pairing p is assigned to him or her.

4. A binary feature indicating whether a preferred off-period of m ends the day before the beginning of p, denoted $f^{(4)}_{pm}$.

5. A binary feature indicating whether a preferred off-period of m begins the day after the end of p, denoted $f^{(5)}_{pm}$.
The last two features are relevant because assigning m a pairing p just before or after a preferred off-period is very efficient.

Entry E_{mp} is given a label λ_{mp}, whose value corresponds to the frequency at which p is assigned to m in the solutions. For instance, if p is assigned to m in 3 out of 5 solutions, $\lambda_{mp} = 3/5 = 0.6$. This label definition is based on the assumption that pairings that are more advantageous for m are more frequently assigned to him or her. However, it is likely that some advantageous entries are wrongly given a zero label because not enough solutions were created to identify all such advantageous entries. Larger instances are more likely to suffer from this problem because, whereas the number of entries for an instance is approximately $|\mathcal{M}| \times |\Gamma|$, each CRP solution affects the value of at most $\max_{p \in \Gamma}\{a_p\} \times |\Gamma|$ nonzero labels. Thus, for large instances, a very large number of solutions would be required to obtain accurate labels. We address this issue in the next subsection.

4.3 Network architecture

We train a deep neural network for each dataset on the task of predicting the label of a new entry. Let λ^P_{pm} denote the predicted label of entry E_{mp}. We obtain Θ_m by ordering the pairings of Γ^m in decreasing order of λ^P_{pm}.

The instances of each dataset are split into three disjoint subsets: a training set, a validation set, and a test set. The number of instances and entries in each set for each of these subsets is given in Table 2. We randomly choose the test (and validation and training) instances among all the instances, but we cannot mix the entries of training instances with the entries of testing instances because we want to evaluate the performance of our branch-and-price algorithm on complete testing instances.

The standard way to build those sets would be to randomly split the entries of all instances. However, because we wish to evaluate the performance of our branch-and-price algorithm as well as that of the ML model, it is necessary to reserve instances for testing purposes.

Table 2: Number of instances in the training, validation and test sets

Dataset	Nb. training instances	Nb. validation instances	Nb. test instances
1	20	5	5
2	8	2	5
3	20	5	5
4	8	2	5
5	8	2	5
6	8	2	5
7	8	2	5

The neural network is a feedforward fully connected deep neural network. Conforming with standard practices, the number of neurons in any hidden layer is less than or equal to that of the previous layer. All neurons except those of the input and output layers are rectilinear (ReLU) units. The output layer is composed of a single sigmoid unit so the output is in the interval $[0, 1]$.

We use different metrics to evaluate the training and validation performances of the model. The validation performance should indicate how good our model is for its intended purpose. In our case, the ML model is used to build Θ_m, which is in turn used to select the pairings of Γ_{mn}. At the beginning of the branch-and-price algorithm, only the first $k = S^{MAX}|\Gamma^m|$ pairings of Θ_m are included in the SP of crew member m. Ideally, the k first pairings of Θ_m would contain all entries E_{mp} such that $\lambda_{mp} > 0$ (i.e. all desirable entries). A good performance indicator of the ML model can be obtained by taking the sum of the "true" labels in the k first pairings of Θ_m for all crew members. We measure the performance of our ML model by computing the *label sum ratio in the top k*, defined as

$$TOP_k = \frac{\sum_{m\in\mathcal{M}} \sum_{i=1}^{k} \lambda_{mp_i^m}}{\sum_{m\in\mathcal{M}} \sum_{p\in\Gamma^m} \lambda_{mp_i^m}}$$

Thus, TOP_k is the sum of the true labels of the first k entries in Θ_m for each crew member $m \in \mathcal{M}$, over the total label sum. Note that the denominator can be omitted because $\sum_{m\in\mathcal{M}} \sum_{p\in\Gamma^m} \lambda_{mp_i^m}$ is independent of the model's predictions. Nevertheless, this definition is more practical when comparing the performance of the neural network for different datasets.

TOP_k is not a suitable loss function for training because it is not a differentiable function of the neural network's output. Instead, the neural network is trained by minimizing binary cross-entropy (BCE) between the true labels and the predicted labels. Let $\mathcal{E}$ be the set of entries. The BCE function is given by

$$H = -\frac{1}{|\mathcal{E}|} \sum_{E_{mp}\in\mathcal{E}} \lambda_{mp} \log(\lambda^P_{mp}) + (1 - \lambda_{mp}) \log(1 - \lambda^P_{mp})$$

Although the BCE loss function is usually used for binary classification problems, it can also be used when label values are in the interval [0, 1] (see Alain et al. 2014, Creswell et al. 2017). It is easy to show that the BCE first-order optimality condition ($\nabla_{\lambda^P} \cdot H = \vec{0}$) implies that $\lambda^P_{mp} = \lambda_{mp}\ \forall(m, p)|E_{mp} \in \mathcal{E}$.

Predicting entries with nonzero labels should be the priority for our application. To see why, suppose the ML model wrongly predicts $\lambda^P_{mp} = 1$ for entry E_{mp}, whose true label is $\lambda_{mp} = 0$. This is of little consequence because even though p has a low rank in Θ_m, the branch-and-price algorithm can still choose not to assign p to m. Conversely, if the ML model wrongly predicts $\lambda^P_{mp} = 0$ for an entry E_{mp} whose true label is $\lambda_{mp} = 1$, it is likely that a pairing conflicting with p gets assigned to m before p is included in m's SP, potentially resulting in a poor solution. Another reason why correctly classifying nonzero labels should be prioritized is that, as explained above, the value of many labels may be underestimated due to the relatively small number of solutions. The information in zero labels is, therefore, less reliable.

Most labels have a zero value, so the binary cross-entropy loss function is likely to favor ML models that accurately predict those labels. We test two strategies to correct this imbalance. The first one consists of creating multiple copies of randomly-selected positive-valued labels. This technique is known as *random oversampling* (Chawla et al. 2011). In our implementation of random oversampling, a hyperparameter controls the ratio between the number of positive-valued and zero-valued labels. The other strategy, called *weight loss*, consists of assigning a higher weight to nonzero labels in the loss function. In our implementation of weight loss, a hyperparameter controls the relative total weight of the positive-valued labels vs. that of the zero-valued labels. The weight of a positive-valued label is proportional to its value (e.g. the weight associated with label $\lambda^P_{mp} = 0.3$ is three times higher than that of label $\lambda^P_{m'p'} = 0.1$). The balancing strategy used (random oversampling, weight loss, or none) is taken as a hyperparameter of the model.

The neural network and the training framework are implemented in Python using the PyTorch library. Training is performed in a supervised fashion using either the stochastic gradient descent (SGD) (Bottou 2010) or the Adam algorithm (Kingma and Ba 2015). Several strategies are used to prevent overfitting. The neurons have a dropout probability between 0 and 0.4. Also, the validation performance (TOP_k) is computed every 10 epochs and the training algorithm is stopped when it degrades twice in a row, or after a fixed number of epochs.

The hyperparameters and their possible values are given in Table 3. We determine the appropriate hyperparameters of the neural network and the training algorithm by performing a random grid search over the hyperparameter space. We select the model that achieves the best validation (TOP_k) performance.

Table 3: Hyperparameters of the neural network

Hyperparameter	Range
Number of hidden layers	{2,3}
Number of neurons in the first hidden layer	[200, 1600]
Number of neurons in the second hidden layer	[10, 800]
Number of neurons in the third hidden layer	[10, 300]
Training algorithm	{SGD, Adam}
Balancing strategy	{None, weightloss, random over-sampling}
Weightloss ratio (if applicable)	[0.4, 0.75]
Random over-sampling ratio (if applicable)	[0.7, 2.5]
Dropout probability	[0, 0.4]
Apply batch normalization (Ioffe and Szegedy 2015)	{yes, no}
Learning rate	[0.01, 0.1]

5 Computational experiments

We now present computational results for the proposed branch-and-price algorithm. All experiments are performed on the test instances described in Section 4.1, using a Linux computer with an Intel i7-8700 CPU clocked at 3.20GHz. First, we evaluate the performance of the ML models in Section 5.1. We then compare the proposed method with several branch-and-price algorithms in Section 5.2.

5.1 Performance of the ML model

We computed a performance profile of each ML model by computing the geometric mean of the TOP_k over the test instances of each dataset, for various values of k. Those performance profiles are displayed in Table 4, in the rows identified by "ML". We also report the average fraction of nonzero labels in each test dataset. The values of k are reported as a fraction of Γ to compare instances of different sizes.

Training a neural network took less than 1 hour for the small datasets and less than 4 hours for the large ones. We do not report those training times beacause not much effort was put into optimizing the training framework, so those training times could likely be significantly improved. Note that high

training times are not a big concern for a real-world application because each ML model would be trained once and then used for several months or years. Furthermore, in the crew sceduling service of an airline, the computers are very busy one week per month and relatively free the others weeks. The training can be done in the weeks where the computers are not busy.

We compare the performance profiles of the ML models with those of a prediction rule designed by an expert (designated by "EX" in Table 4). It is a linear combination of some of the features presented in Section 4.2, with the weight of each feature adjusted manually. The expert prediction rule for pairing p and crew member m is

$$\lambda_{pm}^{EX} = \sum_{l \in \mathcal{L}} f_{lpm}^{(e.)} + 2 f_{pm}^{(2)} - f_{pm}^{(3)} + f_{pm}^{(4)} + f_{pm}^{(5)}. \tag{8}$$

Equation (8) thus balances two objectives: ensuring that pairings that require scarce languages appear in the subproblem of crew members who speak those languages and ensuring that the subproblem of crew member m contains pairings he or she prefers. The first term of (8) promotes the first objective by increasing λ_{pm}^{EX} if some of the languages that are required by pairing p are compatible with those spoken by crew member m. Furthermore, scarce languages have a higher weight. The other terms of (8) promote the second objective. We put a weight of 2 for feature (2) because there is a high correlation between $f_{pm}^{(2)}$ and the probability that p is given to m. A weight of -1 is put on feature (3) to penalize entries conflicting with preferred off-periods. Finally, we put a weight of 1 on features (4) and (5) because we observe crew members are likely to be granted a preferred off-period if they are assigned a pairing that begins (ends) just after (before) that off-period. Some of the features are not used in (8) because including them did not improve the performance profiles.

Note that the result of this prediction function is not necessarily on the same scale as the labels (i.e. in the interval $[0, 1]$). This is not a problem because TOP_k is computed using the pairing orderings rather than the difference between the predicted labels and the real ones.

Table 4: Average TOP_k for the test instances

Dataset	Avg. frac. nonzero labels (%)	Average TOP_k, with k as a percentage of $\|\Gamma\|$								
		$k =$	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	26.2	ML	48.5	65.9	76.7	85.9	91.7	95.5	97.8	99.2	99.8
		EX	43.9	61.4	71.4	80.3	86.2	92.1	96.3	98.6	99.8
2	19.9	ML	33.8	48.0	58.9	69.9	79.6	87.4	93.1	97.5	99.6
		EX	27.7	43.6	56.9	67.4	77.2	86.5	92.5	97.0	99.6
3	28.0	ML	48.4	63.9	74.4	83.3	90.3	94.6	97.3	99.2	99.8
		EX	37.8	57.0	69.7	78.2	86.5	93.0	96.4	98.6	99.8
4	5.3	ML	63.2	80.9	89.0	95.4	98.3	99.1	99.6	99.9	100.0
		EX	52.1	72.0	82.5	89.8	95.5	98.0	99.0	99.7	99.9
5	2.8	ML	74.4	87.3	94.0	97.6	98.8	99.4	99.7	99.9	100.0
		EX	62.4	79.4	86.9	92.6	96.8	98.4	99.1	99.6	99.9
6	3.0	ML	57.6	73.6	83.4	91.9	95.5	97.0	98.1	98.9	99.5
		EX	48.3	66.8	77.5	86.5	93.5	96.4	97.7	98.7	99.5
7	2.9	ML	53.1	66.6	75.1	82.2	86.9	90.4	93.5	96.2	98.3
		EX	42.4	59.5	68.8	76.6	84.1	89.0	92.4	95.5	98.1

Comparing the performance profiles of the ML model for different datasets, we observe that the small instances (from datasets 1 to 3) have, on average, smaller TOP_k values than the large instances.

This is because the small instances have a higher proportion of nonzero labels than the large ones, making it harder to achieve high TOP_k values. For example, the ML model for dataset 5 achieves a $TOP_{30\%} = 94\%$, but most entries in the predicted top 30% have a zero-valued label. To achieve a similar TOP_k value for dataset 3, the predicted 30% would have to be almost exclusively composed of entries with nonzero labels, which leave very little place for prediction mistakes. Those results indicate that in order to have a sufficient number of *good* pairings in each subproblem, it is necessary to use a larger value of S^{MAX} for the small instances than for the large ones. The TOP_k for the instances of dataset 2 is lower than that of the other small instances, for all values of k. This is likely because fewer instances were used to train the neural network of dataset 2.

We observe very large variations in the TOP_k performance for the ML model among the large instances. Unfortunately, we are unable to identify the cause of these differences. Likely, the low $TOP_{10\%}$ performance observed for the instances of dataset 4 is linked to the relatively high percentage of nonzero labels in that instance, as explained above. Anecdotal evidence also suggests that TOP_k performance might be worse for datasets with a high pairings-to-crew-members ratio, corresponding to CPP solutions containing a large number of pairings of short length. In those cases, some of the learning features may be less indicative of the fitness of individual entries. For example, the two features indicating the proximity of a pairing to a preferred off-period may be weakly correlated with the fitness of the entries for those instances because the flexibility arising from the large number of pairings makes it easier to grant those off-period preferences. We did not further investigate this matter because those TOP_k variations for the large instances do not appear to affect the performance of our solution method.

Finally, we compare the performance profiles of the ML model to those of the expert prediction function. The expert predictions are far better than a random prediction function, for which the expected value of $TOP_{x\%}$ is x. However, for all datasets and all values of k, the TOP_k performance of the ML model is higher than that of the expert prediction function. In fact, the $TOP_{40\%}$ for the ML model is at least 5 percentage points higher than that of EX, except for dataset 2 for which the $TOP_{40\%}$ is 2.5 percentage points higher than that of EX. This indicates that although it is possible for an expert to design a prediction function that is somewhat accurate, a ML model can be significantly more accurate.

5.2 Results on the crew rostering problem

Let Alg^{ML} denote the branch-and-price algorithm described in Section 3. We compare Alg^{ML} with the following algorithms :

- Alg^{STD} : A standard branch-and-price method similar to those used in airlines. It is equivalent to Alg^{ML} with $S^{MAX} = 1$, i.e. the SP of crew member $m \in \mathcal{M}$ contains all pairings that are feasible for m.
- Alg^{RF} : Alg^{ML} with Θ_m generated randomly.
- Alg^{RV} : Alg^{ML} with Γ_{mn} selected randomly each time SP m is solved.
- Alg^{EX} : Alg^{ML} with Θ_m generated according to the expert prediction function (8).

Results are presented in Table 5. Each row contains the results for one dataset, averaged over the test instances of that dataset. We report the average CPU time and objective value for Alg^{STD} as well as the average relative difference in computing time and objective value between each algorithm and Alg^{STD}. For all algorithms except Alg^{STD}, we use $S^{MAX} = 0.6$ for datasets 1–3 and $S^{MAX} = 0.2$ for datasets 4–7. Those values were determined in preliminary tests. It is necessary to use a large value of S^{MAX} in small instances so that enough pairings are included in each SP.

We first compare Alg^{STD} with Alg^{ML}. For all datasets, Alg^{ML} finds solutions in less than 50% of the CPU time of Alg^{STD}, and in less than 10% of the CPU time for dataset 4. The solutions produced by Alg^{ML} are, on average, slightly inferior to those computed by Alg^{STD}, with average differences in the objective value less than or equal to 0.55% for all datasets except for datasets 2 and 4. This was

expected because our ML model sometimes makes mistakes which causes some advantageous pairings to be wrongly excluded from some SPs. Nevertheless, the optimality gaps obtained at the root node are relatively small (close to the negative of the value difference with Alg^{STD} and less than 1% except for datasets 2 and 4), indicating that such mistakes are relatively rare. This shows that in most cases, Γ_{m0} (the set of pairings in the SP of crew member $m \in \mathcal{M}$ at the root node) is sufficient to create a near-optimal linear relaxation solution to the CRP.

The only difference between Alg^{RF}, Alg^{EX} and Alg^{ML} is the accuracy of the predictions used to create Θ_m: the predictions of Alg^{ML} are more accurate than that of Alg^{Ex}, which, in turn, are more accurate than that of Alg^{RF}. We observe similar computing times for all algorithms. However, the solutions produced by Alg^{RF} have a significantly lower objective value, on average, than that of all other algorithms tested. The values of the solutions of Alg^{EX} are larger than those of Alg^{RF} but much smaller than those of Alg^{ML}. This shows that accurate predictions are necessary for our method to produce good-quality solutions. The scoring function designed by an expert is not accurate enough to be useful in the proposed method.

Alg^{RV} produces solutions with a larger objective value than those produced by Alg^{ML}, except for datasets 5 and 6. This is because by selecting a different set of pairings each time a SP is solved, the pairings of any good schedule for $m \in \mathcal{M}$ are likely to be included together at least once in m's SP. However, Alg^{RV} is comparatively slow: its computing times are similar to those of Alg^{STD}. Although each SP of Alg^{RV} is faster to solve due to their smaller SP sizes, the algorithm performs many more iterations. A typical behavior of column generation is a tailing-off effect, where the last column generation iterations generate few useful columns. In Alg^{RV}, it may take several iterations before a given negative reduced-cost column can be generated in a SP because every pairing of the column must be included in the SP, which has the effect of extending the tailing-off effect for that algorithm. We do not observe this behavior in Alg^{ML} because the set of pairings included in each SP is fixed at each node of the branch-and-bound tree.

Table 5: Comparaison of Alg^{STD} with Alg^{RF}, Alg^{RV}, Alg^{EX}, and Alg^{ML}

Dataset	CPU	CPU Diff (%) vs Alg^{STD}				Value	Value Diff (%) vs Alg^{STD}			
	Alg^{STD}	Alg^{RF}	Alg^{RV}	Alg^{EX}	Alg^{ML}	Alg^{STD}	Alg^{RF}	Alg^{RV}	Alg^{EX}	Alg^{ML}
1	79	-69	-12	-57	-50	39563	-26.00	-0.32	-10.05	-0.63
2	832	-63	20	-70	-67	33572	-38.09	-0.33	-14.67	-5.73
3	1884	-65	-21	-70	-64	84273	-13.88	-0.05	-6.98	-0.28
4	39778	-71	-9	-94	-92	195922	-6.53	-0.36	-5.55	-1.64
5	1425	-60	-13	-62	-60	374579	-35.49	-0.33	-33.30	-0.15
6	6973	-82	-13	-78	-51	501447	-17.32	-0.05	-3.50	-0.02
7	64650	-83	-46	-89	-80	508663	33.01	0.36	-10.65	-0.55
Average	**16505**	**-61**	**-12**	**-66**	**-59**	**41326**	**-20.62**	**-0.11**	**-10.66**	**-1.19**

We now perform a further analysis of Alg^{ML} for some datasets of interest. As discussed above, Alg^{ML} produces unsatisfactory solutions for dataset 2 with $S^{MAX} = 0.6$ and for dataset 4 with $S^{MAX} = 0.2$. In those cases, increasing the value of S^{MAX} may yield solutions with a larger objective value. Conversely, Alg^{ML} finds solutions with an objective value very close to that of Alg^{STD} for dataset 6 and $S^{MAX} = 0.2$. In that case, it may be possible to use a smaller value of S^{MAX} to speed up the algorithm, and still get solutions with large values. We tested Alg^{ML} with a different value of S^{MAX} for datasets 2, 4 and 6. Results are presented in Table 6. This table also reproduces the results of Table 5 for Alg^{STD} and Alg^{ML} for easy comparaison.

For dataset 2, Alg^{ML} with $S^{MAX} = 0.8$ finds solutions with objective values close to that of Alg^{STD}. As expected, we observe longer computing times than Alg^{ML} with $S^{MAX} = 0.6$, but those computing times remain significantly smaller than those of Alg^{STD}. We observe similar results for the

Table 6: Alg^{ML} with different values of S^{MAX} for datasets 2, 4, and 6

Dataset	Algorithm	CPU (s)	vs Alg^{STD} (%)	Objective	vs Alg^{STD} (%)
2	Alg^{STD}	832		-33572	
	Alg^{ML} $S^{MAX} = 0.6$	272	-67	-31649	-5.73
	Alg^{ML} $S^{MAX} = 0.8$	506.3	-39	-33459	-0.34
4	Alg^{STD}	39778		195922	
	Alg^{ML} $S^{MAX} = 0.2$	3064	-92.30	192704	-1.64
	Alg^{ML} $S^{MAX} = 0.3$	9187	-77	194863	-0.54
6	Alg^{STD}	6973		-501447	
	Alg^{ML} $S^{MAX} = 0.2$	3441	-51	-501366	-0.02
	Alg^{ML} $S^{MAX} = 0.1$	1171	-83	-500383	-0.21

solutions of the dataset 4 instances obtained using Alg^{ML} with $S^{MAX} = 0.3$. Solving the dataset 6 instances using Alg^{ML} with $S^{MAX} = 0.1$ we obtain solutions with an objective value 0.21% smaller on average than those obtained using Alg^{STD}, in 17% of the computing time. This shows that our algorithm can sometimes be sped up without sacrificing much in terms of objective value.

6 Conclusion

In this paper, we propose a branch-and-price algorithm for the personalized CRP that performs partial pricing by limiting the size of the SPs. Our method creates a personalized ordering of the pairings for each crew member according to their desirability, and only the most desirable pairings are included in each of the crew member's SP. This ordering is created using a ML model that is a deep neural network trained on historical data. Rather than trying to predict the *true* desirability of pairings, this ML model acts as an expert that predicts which pairing assignments are more likely to be chosen by the optimizer. The features used to train the model are simple and do not require a deep understanding of the problem. The proposed method could be adapted to the needs of real-life airlines with minimal efforts.

The proposed approach creates near-optimal CRP solutions in less than half of the time of a standard branch-and-price method. We show that the predictions of our ML model are more accurate than that of an expert and that this level of accuracy is necessary for our method to produce high-quality solutions. In a broader context, this paper contributes to showing that information derived from an offline ML model can be of great use to speed up optimization algorithms.

References

Guillaume Alain, Yoshua Bengio, Aaron Courville, Rob Fergus, and Christopher Manning. What Regularized Auto-Encoders Learn from the Data-Generating Distribution. Technical report, 2014.

A. Azadeh, M. Hosseinabadi Farahani, H. Eivazy, S. Nazari-Shirkouhi, and G. Asadipour. A Hybrid Meta-heuristic Algorithm for Optimization of Crew Scheduling. Applied Soft Computing, 13(1):158–164, 2013. doi: 10.1016/j.asoc.2012.08.012.

Ralf Borndörfer, Uwe Schelten, Thomas Schlechte, and Steffen Weider. A Column Generation Approach to Airline Crew Scheduling, pages 343–348. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. doi: 10.1007/3-540-32539-5_54.

Léon Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Proceedings of COMPSTAT'2010, pages 177–186. Physica-Verlag HD, 2010. doi: 10.1007/978-3-7908-2604-3_16.

Khaled Boubaker, Guy Desaulniers, and Issmail Elhallaoui. Bidline scheduling with equity by heuristic dynamic constraint aggregation. Transportation Research Part B: Methodological, 44(1):50–61, jan 2010. doi: 10.1016/j.trb.2009.06.003.

Paola Cappanera and Giorgio Gallo. A multicommodity flow approach to the crew rostering problem. Operations Research, 52(4):583–596, jul 2004. doi: 10.1287/opre.1040.0110.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16:321–357, jun 2011. doi: 10.1613/jair.953.

Qiao Chen, Andrew Lim, and Wenbin Zhu. A greedy heuristic for airline crew rostering: Unique challenges in a large airline in China. In Lecture Notes in Computer Science, volume 6704 LNAI, pages 237–245. Springer, Berlin, Heidelberg, 2011. ISBN 9783642218262. doi: 10.1007/978-3-642-21827-9_24.

Antonia Creswell, Kai Arulkumaran, and Anil A Bharath. On denoising autoencoders trained to minimise binary cross-entropy. CoRR, abs/1708.0, aug 2017. URL http://arxiv.org/abs/1708.08487.

Herbert Dawid, Johannes König, and Christine Strauss. An enhanced rostering model for airline crews. Computers and Operations Research, 28(7):671–688, jun 2001. doi: 10.1016/S0305-0548(00)00002-2.

Jesica de Armas, Luis Cadarso, Angel A. Juan, and Javier Faulin. A multi-start randomized heuristic for real-life crew rostering problems in airlines with work-balancing goals. Annals of Operations Research, 258(2):825–848, nov 2017. doi: 10.1007/s10479-016-2260-y.

Guang Feng Deng and Woo Tsong Lin. Ant colony optimization-based algorithm for airline crew scheduling problem. Expert Systems with Applications, 38(5):5787–5793, may 2011. doi: 10.1016/j.eswa.2010.10.053.

Tsubasa Doi, Tatsushi Nishi, and Stefan Voß. Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time. European Journal of Operational Research, 267(2):428–438, jun 2018. doi: 10.1016/j.ejor.2017.11.046.

Michel Gamache, François Soumis, Gérald Marquis, and Jacques Desrosiers. A Column Generation Approach for Large-scale Aircrew Rostering Problems. Operations Research, 47(2):247–263, 1999.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In 32nd International Conference on Machine Learning, ICML 2015, volume 1, pages 448–456. International Machine Learning Society (IMLS), feb 2015. ISBN 9781510810587.

Stefan Irnich and Guy Desaulniers. Shortest Path Problems with Resource Constraints. In Column generation, pages 33–65. Springer, 2005.

Atoosa Kasirzadeh, Mohammed Saddoune, and François Soumis. Airline crew scheduling: models, algorithms, and data sets. EURO Journal on Transportation and Logistics, 6(2):111–137, jun 2017. doi: 10.1007/s13676-015-0080-x.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. International Conference on Learning Representations, ICLR, dec 2015.

Niklas Kohl and Stefan E. Karisch. Airline Crew Rostering: Problem Types, Modeling, and Optimization. Annals of Operations Research, 127(1-4):223–257, mar 2004. doi: 10.1023/B:ANOR.0000019091.54417.ca.

Broos Maenhout and Mario Vanhoucke. A hybrid scatter search heuristic for personalized crew rostering in the airline industry. Innovative Applications of O.R, 206(1):155–167, oct 2010. doi: 10.1016/j.ejor.2010.01.040.

H. Timucin Ozdemir and Chilukuri K. Mohan. Flight graph based genetic algorithm for crew scheduling in airlines. Information Sciences, 133(3–4):165–173, apr 2001. doi: 10.1016/S0020-0255(01)00083-4.

Frédéric Quesnel, Guy Desaulniers, and François Soumis. A branch-and-price heuristic for the crew pairing problem with language constraints. European Journal of Operational Research, 283(3):1040–1054, 2020. doi: 10.1016/j.ejor.2019.11.043.

Mohammed Saddoune, Guy Desaulniers, Issmail Elhallaoui, and François Soumis. Integrated Airline Crew Pairing and Crew Assignment by Dynamic Constraint Aggregation. Transportation Science, 46(1):39–55, feb 2012. doi: 10.1287/trsc.1110.0379.

D Sasaki and T Nishi. Combinatorial Auction Algorithm with Price-Adjustment Mechanism for Airline Crew Scheduling Problems. In Proceedings of 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pages 1183–1187, 2015. doi: 10.1109/IEEM.2015.7385834.

Jeffery D Weir and Ellis L Johnson. A Three-Phase Approach to Solving the Bidline Problem. Annals of Operations Research, 127(1):283–308, mar 2004. doi: 10.1023/B:ANOR.0000019093.93633.bc.

Nan Xu, Mingyu Zhao, Yangfan Liu, and Fan Yang. A column generation based algorithm for airline cabin crew rostering problem. In 2018 IEEE 4th International Conference on Computer and Communications, ICCC 2018, pages 2511–2515, dec 2018. ISBN 9781538683392. doi: 10.1109/CompComm.2018.8780853.

Vahid Zeighami and François Soumis. Combining Benders decomposition and column generation for integrated crew pairing and personalized crew assignment problems. Transportation Science, 53(5):1479–1499, 2019.