Dynamic evaluation of airline Crew's flight requests using a neural network

Beulen, M.; Scherp, L.; Santos, B. F.

DOI
10.1016/j.ejtl.2020.100018

Publication date
2020

Document Version
Final published version

Published in
EURO Journal on Transportation and Logistics

EURO Journal on Transportation and Logistics 9 (2020) 100018

Contents lists available at ScienceDirect

EURO Journal on Transportation and Logistics

journal homepage: www.journals.elsevier.com/euro-journal-on-transportation-and-logistics

Dynamic evaluation of airline Crew's flight requests using a neural network

M. Beulen, L. Scherp, B.F. Santos *

Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology, the Netherlands

A B S T R A C T

In airline crew rostering, pilots' requests to operate specific flights need to be evaluated efficiently to avoid inefficient schedules. Despite the relevance of correctly assessing and granting crew requests, this topic has received very little attention in the literature. In this paper, we address the case this process is a dynamic problem, in which flight requests are submitted while others have already been granted and pre-assigned. This is the first work to dynamically model flight requests during the crew rostering process. We propose a simulation-trained neural-network algorithm to evaluate flight requests, providing a systematic way of assessing flight requests and supporting the definition of a cost-efficient request granting policy. To train and test this algorithm, we developed an innovative rolling rostering framework that captures the dynamic process in practice. The framework relies on an integer linear programming crew rostering model solved with the help of a column-generation algorithm. The neural-network algorithm is trained and tested in a case study with a major European airline. The results show that the algorithm is more effective than the current practice at the airline, granting 22% more requests while using the same workforce to operate the flight schedule.

1. Introduction

Airline crew planning is a complex process that is often divided into two phases: *crew pairing* and *crew rostering* (or *assignment*) ((Kasirzadeh et al., 2017)). The crew pairing phase concerns the development of a set of *pairings* composed by a sequence of *flight segments*, such that costs are minimized and each flight in the schedule is covered by only one pairing. In the crew rostering phase, individual monthly *rosters* (or schedules) are created and assigned to specific crew members. Since the crew rostering problem is concerned with individual crew members, roster construction must take into account the pre-assignment of other activities to specific crew members. These activities are reserve duties, training sessions, off-duty blocks or granted flight requests that may have been assigned in an earlier scheduling stage.

At most European airlines, crew members can express their preference to operate specific pairings a few weeks before the rostering phase. For the crew members, these requests are relevant, especially for those operating long-haul flights. These flights are covered by pairings that may have a time span of over a week, meaning that the crew will be away from their crew base for multiple consecutive days. This can be irregular and challenging from a personal planning point of view and therefore, the possibility to request flights can be highly valued by crew members. For the airline, allowing flight requests is one of the possible means for airlines to accommodate for crew satisfaction. In practice, airlines decide which request to grant based on estimations of the airline future needs and rules defined in the collective labour agreements. The majority of the airlines assess the flight request from all crew members simultaneously, at the time the monthly rosters are produced. However, there is an increasing number of airlines that are adopting a dynamic crew request approach, in which feedback is given to the crew member shortly after the request is expressed. This approach necessarily raises challenges in terms of the future feasibility and productivity of the rosters. However, there are multiple advantages from this method. Most benefits are on the crew side. With this dynamic method, crew members have more control over their rosters, receiving a fast response to their requests and fixing some of the desired activities in their roster several weeks before operations. For airlines, this method is a way of improving the social quality of the rosters and increase crew satisfaction.

To properly assess a-priori the impact of each request is a challenging problem, given that requests usually refer to pairings that start weeks or even months after the current rostering planning horizon and implications cannot yet be determined. Wrong decisions will necessarily lead to costs to operate the flight schedule. To further illustrate this, Fig. 1 shows a Gantt chart representation of three example crew schedules for a period of 10 weeks. Week 1 is the current week of operations. The schedules consist of blocks of multiple-day pairing activities. Three consecutive scheduling horizons are distinguished in this example. The first is week 1 till 4 in which the schedule has already been finalized and published to the crew members. Although still prone to disruptions and mutations, this part of the schedule is typically considered within the disruption management practice of an airline. The second horizon is the moment when the crew rostering problem is solved and adapted before publishing it (the thick vertical line at Week 4). The goal is to ensure that all activities in the schedule are covered with the required crew members in a

* Corresponding author.
E-mail address: b.f.santos@tudelft.nl (B.F. Santos).

https://doi.org/10.1016/j.ejtl.2020.100018
Received 12 January 2020; Received in revised form 30 June 2020; Accepted 16 July 2020

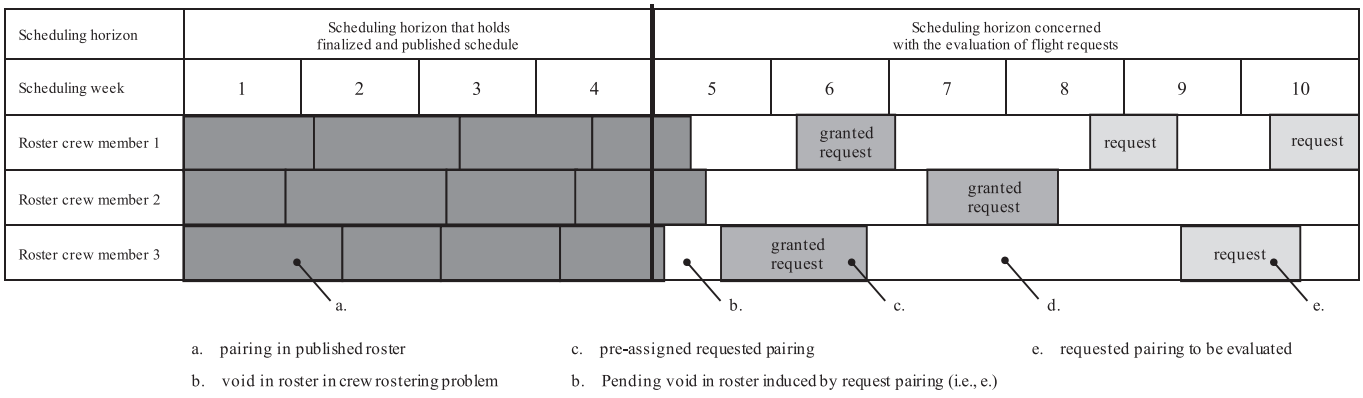

Fig. 1. Three example crew schedules for a period of 10 weeks from an airline schedule controller perspective indicating the challenge of evaluating flight requests.

cost-efficient way. The third horizon is the period in which crew members can submit their requests (*e* in Fig. 1). That is, in this case, crew may submit their request up to 10 weeks before operations and before the roster is published. Depending on the collaborative labour agreement, the airline has a couple of days after receiving the request to take the decision of grant or not to grant it. The granting of requests can be considered as pre-assignment decisions (*c* in Fig. 1) that can only be revoked with high costs. These pre-assignments may lead to voids in the roster (*b* in Fig. 1) when solving the crew rostering problem. These voids may indicate an inefficient use of resources with respect to a scenario in which pre-assigned activities are not considered.

This paper presents one of the first works addressing this problem of dynamic evaluating flight requests. We propose a neural network classification algorithm to assess each flight request, based on features from the request, flight and current status of the schedule. To train and test the neural network algorithm, we developed a dynamic rostering framework that evaluates flight requests over time. The framework combines a mixed integer linear programming (MILP) rostering optimization model with the neural-network algorithm. The rostering model is designed to simulate and test the dynamic crew rostering process, assessing the effect of granting requests that can reduce the efficiency of future rosters. These results are used to train the neural-network algorithm, such that future requests can be classified in terms of their potential impact on the rostering efficiency.

The contribution of this work is two-fold. First, this is the first work to directly model and assess crew requests when building crew schedules. Although having received little or no attention so far in the literature, the dynamic evaluation of crew requests is current practice for several airlines. Second, it proposes a machine-learning approach to evaluate flight requests, assessing each request based on the potential impact that it may have on the crew rostering costs at a later stage. This is an essential step as it allows for an adaptive evaluation method that is applicable, and can be retrained, for changing rostering scenarios. The machine learning approach can be used by airlines as it can give a score per request, given support to the decision of the crew schedulers when evaluating crew requests.

The remainder of the paper is structured as follows: Section 2 presents an overview of existing literature and discusses the context of the flight request evaluation problem. Section 3 presents the methodology. The experiments and results are presented in Section 4. Section 5 concludes this paper with a summary of the main findings and considers future research directions based on this work.

2. Literature and research context

2.1. Crew rostering literature

Kasirzadeh et al. (2017) provided a comprehensive literature overview and description of the crew scheduling problem, involving crew pairing (i.e., the problem of generating unassigned pairings), the and crew rostering (i.e., the problem of allocating pairings to individual crews). In particular, the authors describe the commonly used approaches in research and the industry to solve the airline crew rostering problem. Approaches to handling crew preferences vary per airline but almost always there is a process in place that incorporates preferences into crew rostering.

Two general approaches to airline crew rostering can be taken. Firstly, in the *bidline schedules*, complete anonymous lines of work (called bidlines) are created. The bidlines are presented to the crew members who can express preferences in the form of a bidding process. The bidlines are then assigned to individual crew members by optimizing for these bids. Examples of methods for the bidline approach were presented by Beasley and Cao (1996), Christou et al. (1999), Boubaker et al. (2010).

Secondly, in the *personalized schedules* approach, schedules are created for individual crew members, while considering pre-assigned activities, such as training periods or holidays, and crew members individual requirements. Crew members can express their preferences by referring to particular roster attributes (Caprara et al. (1998)). Examples of roster attributes are duty start times, duty end times, pairing length, pairing destination, specific pairings with fixed start and end date, multiple day leave of absence, flying East- or Westbound. These attributes are not always mutually exclusive. In the context of some airlines, crews can be more specific and indicate preferences for particular activities, such as flight pairings or a weekend off (Gamache (1998)). In this personalized schedules approach, preferences and crew satisfaction can be either considered at the global level (the *rostering* problem) or by given priority to the satisfaction of senior crew members (the *seniority-based* or *preferential bidding* problem). A definition of the rostering problem as a set-covering optimisation is provided by Kohl and Karisch (2004). The authors are affiliated with the Carmen Crew Rostering system which is used at major European airlines such as Alitalia, British Airways, KLM as well as the German state railway company Deutsche Bahn. Other examples of the rostering problem in the literature are the works from Day and Ryan (1997), which considered an integer programming approach to address the short-haul operations of Air New Zealand, Gamache (1999), which compared the results from a proposed column-generation algorithm with the rostering solution at Air France, and Maenhout and Vanhoucke (2010) that used a Dantzig–Wolfe decomposition and a scatter search method. The preferential bidding problem is less present in the literature, because most airlines prefer to adopt the rostering problem (Eltoukhy et al. (2017)). Gamache (1998) presented a model for assigning pilots using the preferential bidding approach for the case of Air Canada. In this model, a problem is solved for each crew member sequentially and in seniority order aiming to maximize the bidding score for each roster. Achour et al. (2007) presented an exact solution method to solve the preferential bidding approach where the assignment of a

roster with maximum bidding score to a more senior crew is delayed until only one feasible roster is left for that crew member. The problem definition and optimisation approaches defined by Gamache (1998) and Achour et al. (2007) are currently implemented in the crew scheduling systems of several North-American airlines.

Recent trends in the crew rostering literature focus on the efficiency of the solution techniques used and on the integration of the problem with the crew pairing problem. In the solution process, literature has focused on the effect of using different types of heuristics to improve roster quality in limited computation time (e.g., Zeghal and Minoux (2006), Saddoune et al. (2011), Azadeh et al. (2013)). The crew pairing and crew rostering problem is commonly called crew scheduling problem in the literature and it has been addressed by, e.g., Guo et al. (2006), Zeghal and Minoux (2006), Saddoune et al. (2012), Azadeh et al. (2013), Kasirzadeh et al. (2017), Quesnel et al. (2019), Zeighami and Soumis (2019).

2.2. Dynamic crew preferences

Most of the previous works, in particular the ones dealing with the personalized schedules approach, consider the existence of pre-assigned activities when solving the rostering problem. However, none of them address the evaluation and definition of those pre-assignments. The pre-assignments are assumed as input. The problem of defining or granting pre-assignments is, nevertheless, a challenge one. Decisions have to be made while assuming that the roster will be finalized at a later stage. Assignment decisions affect the state of the roster in a later planning stage causing inefficiencies in the crew schedule.

In practice, most pre-assignment decisions are made by schedulers that follow basic airline-imposed rules or collaborative agreements. A systematic and better informed strategy could support the decision process, by evaluating the impact of assignment some activities in advance. In particular, for this work, such a systematic strategy could help to analyse crew requests and determine in advance relevant characteristics of the expected rosters. Although Kohl and Karisch (2004) mentioned the existence of preferential bidding systems on the market for fast feedback on crew requests during the bidding phase, this problem was never addressed in the literature.

2.3. Evaluation mechanisms

Within the domain of nurse rostering, Smet et al. (2014) address that it is common in the academic body that schedule periods are isolated when modeling them and that this does not conform to real-world requirements. Many of the problem constraints relate to the following or previous schedule periods and to continuity in general. This part is often not taken into account in the evaluation metrics of a roster. Within the airline crew rostering domain, Biskup and Simons (2004) acknowledges this and states that very few models consider learning effects in scheduling. The author distinguishes between autonomous learning and induced learning methods. Because of rapid technological progress in terms of computing, the importance of considering learning effects as a competitive advantage in mathematical models is self-evident. The authors use learning effects in the determination of due dates in a scheduling environment. An example of induced learning is presented by Suraweera et al. (2013), who proposes a method that is able to infer constraints from historical crew schedules based on a set of user provided template outlining the general structure of important constraints. Complex multivariate constraints can be induced by the algorithm. In a study on the operational airline crew scheduling problem, Stojković et al. (2009) argue that a very relevant asset to research would be the day-to-day decision data of schedulers that make decisions on the scheduling of disturbances. It is then possible to compare results obtained by airline operators with automated decision models. In terms of modeling crew preferences, a validation with operational data is often missing. Only one author (Maenhout and Vanhoucke, 2010) has used one month of airline crew preference data as validation for modeling crew preferences in airline crew rostering. Contrary to this work, our research had access to the historical rostering and flight request data (on both submitted requests and corresponding decisions) from a major European airline.

2.4. Research goal

The goal of this is to address the identified literature gap proposing the first crew's request evaluation method. We propose a data-driven approach, since learning methods are expected to provide a suitable methodology for evaluating flight requests. The goal of supervised machine learning is to reason from externally supplied class labels in a data set what the class label of a future instance might be. Translating that to crew preference management, it can be tested whether the granting or rejecting of individual crew preferences can be evaluated using information about the current roster status and the attributes of the request itself. Supervised learning can be divided into two categories: classification and regression. Since an individual crew request decision is a yes-or-no decision, classification is expected to suit the problem specifics more accurately. An overview of supervised learning algorithms is presented in a review paper by Kotsiantis (2007).

Classification algorithms have to be trained with data that reflect good and bad decisions. This is essential to teach the algorithm how to make the decisions based on the context of the request. The dependency on data increases when we increase the number of features considered to describe the context or when we aim for more accurate decisions. In the context of crew's flight request, there is not much data available that could be used to train such an algorithm. Therefore, we propose a dynamic rostering framework to be used as a simulator and training gym for our classification algorithm. This dynamic framework is described in the next section of the paper.

3. Dynamic rostering framework

For an airline, it is desired to grant as many requests as possible with the least possible required crew members to cover all the flight pairings. Granting flight requests serves as a mean to promote crew satisfaction, while the need for additional crew members represents higher operating costs. The trade-off between these two factors is hard to model in a concrete way – i.e., it is hard to determine what should be the maximum costs to grant a pairing request. Therefore, we propose a dynamic rostering framework that makes use of a supervised machine learning requests evaluation algorithm to decide whether to grant or reject flight pairing requests based on previous experience. This way, no trade-off or cost threshold is defined to determine which requests to grant. The methodology proposed was developed considering the problem of developing personalized roster for the long-haul cockpit crew from an airline operating according to personalized rostering with pre-assigned activities.

The dynamic rostering framework is summarized in Fig. 2, in which the process blocks are labeled referring to the dynamic rostering framework or DRF. The framework is based on a rolling roster modeling approach. In short, the dynamic framework loops over time, solving a rostering model for every time step, while taking into account pairing assignments or granted requests decided in previous time steps. The framework is used to simulate the rostering process for multiple weeks. Several simulations can be run in which the rostering problem with flight request evaluation can be solved for multiple consecutive time periods.

In this section, we explain the framework in detail. We start by justifying the concept of time-space networks that we followed to represent the rostering problem (**DRF.3**, in Fig. 2). In Subsection 3.2 we discuss the shortest path algorithm used to generate cost-efficient rosters (**DRF.4**). Then, we present the formulation of the rostering model used at each time step (**DRF.5**), followed by the column generation algorithm used to solve the rostering problem (**DRF.6–.9**). In Subsection 3.5 we

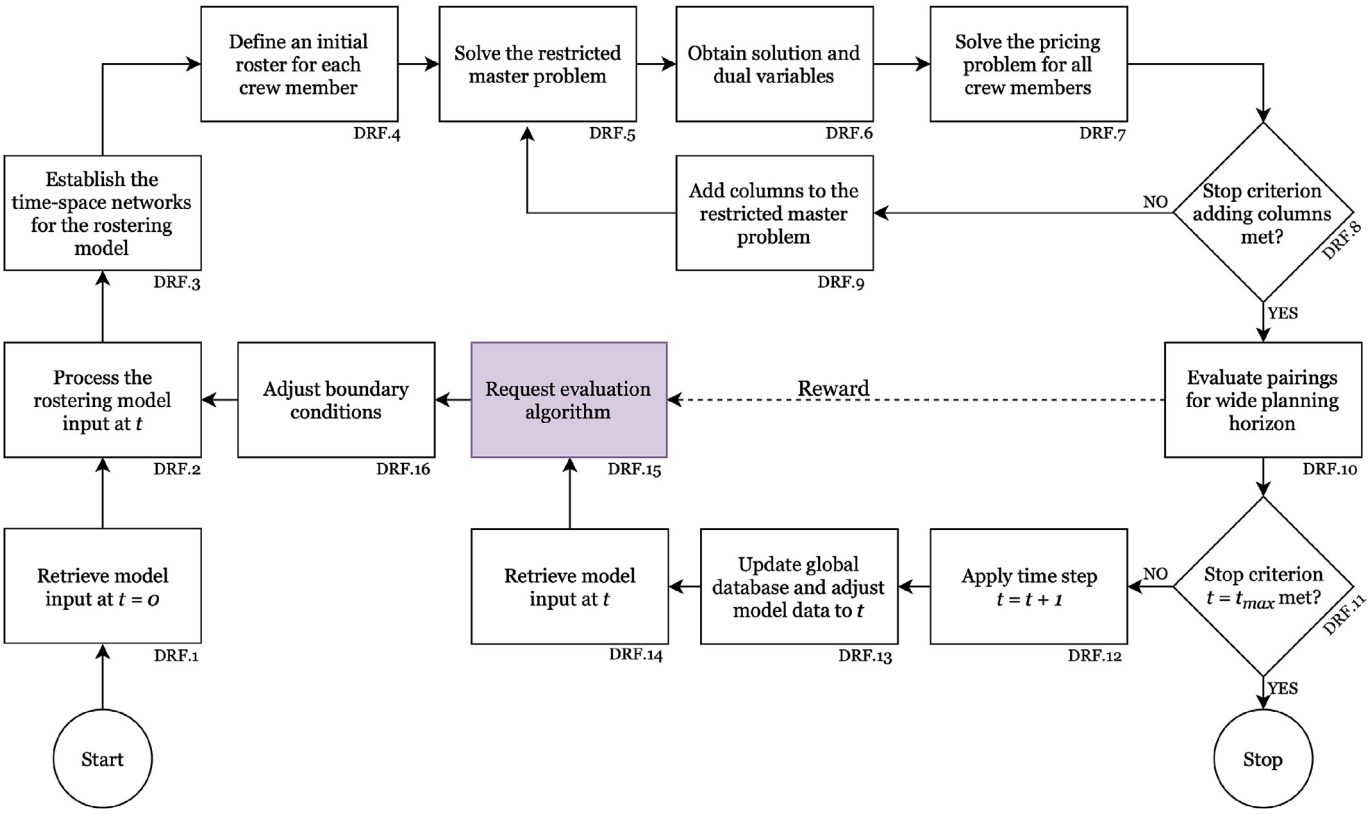

Fig. 2. Flow diagram of the dynamic rostering framework (DRF).

explain how resulting rosters are evaluated (**DRF.10**). Then, we explain how we generated new pairing requests for each time step. In Subsection 3.4, the simulation-trained neural network algorithm and other benchmark algorithms that are used as request evaluation algorithms (**DRF.15**) are introduced. We conclude this section by explaining the process of creating and adjusting boundary conditions in the rosters (**DRF.16**).

3.1. Time-space network

Time-space networks are defined for each time the rostering model is required to construct a framework for feasible sequencing of activities in terms of time and space (**DRF.3**, in Fig. 2). Such a network graph is composed of nodes that are connected by arcs. The nodes represent activities (e.g., flights or layovers) to time and space and the arcs represent the possible connections between the nodes.

To give a visual impression, a 7-day time-space network is presented in Fig. 3. The figure shows an example with three different pairings - Pairing A that is repeated every other day; Pairing B that is repeated twice per week; and Pairing C that is operated only once per week. Pairings are assumed to be an input to the dynamic rostering framework. Note that the pairings represented in this figure are simplified. In practice, they can have a duration of 10 or more days and they include flights, rest periods and other activities, such as trainings and office days. In this figure, the horizontal and vertical axes indicate the time and space component of the network, respectively. A cost value can be assigned to an arc which indicates the costs of operating such pairing or being at the airline base airport off-duty.

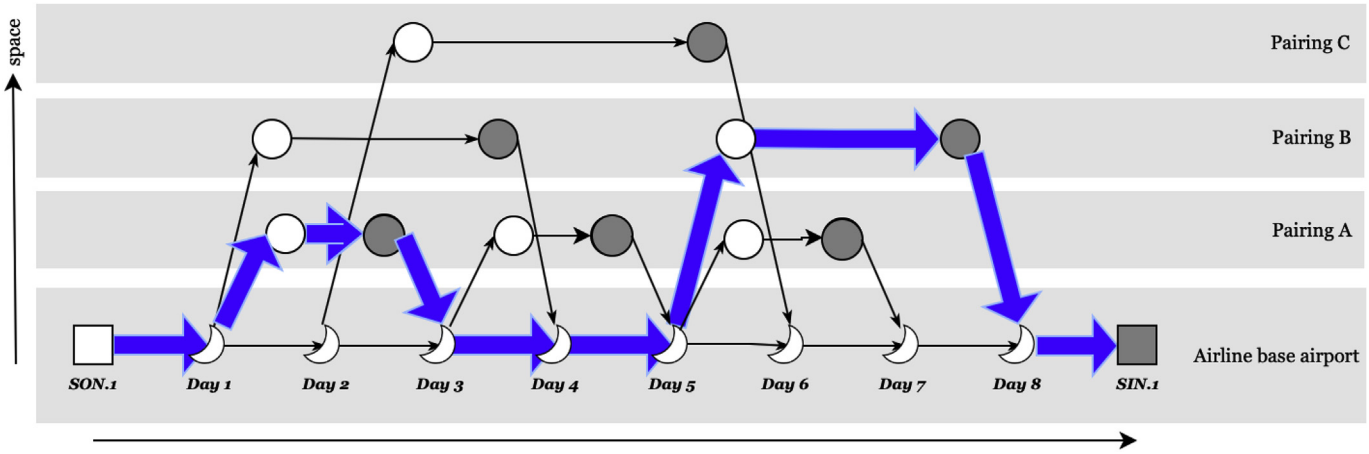

Fig. 3. Example of a 7-day time-space network with three different pairings. In thick arrows the shortest path between the roster source node *SON.1* and the roster sink node *SIN.1*.

3.2. Shortest path algorithm

Every possible path between the source node *SON.1* to the sink node *SIN.1* in Fig. 3 is a different roster for a pilot. Even for this very simple example, there are 15 possible 7-day rosters. For a practical case with thousands of pilots and hundreds of flights operated per day, it would be very computationally demanding to compute all possible paths for all pilots, compare them and determine the optimal distribution of pairings among pilots.

Therefore, we make use of the time-space network and of a shortest path algorithm to generate cost-effective sequences of activities (rosters) for crew members (**DRF.4**). The shortest paths are retrieved using Dijkstra's algorithm (Ahuja et al., 1990). An example of a roster is provided in blue thick arrows in Fig. 3, in which the pilot would have to start operating Pairing A on the first day, have to rest days at the airline base airport and then operate Pairing B.

Before running the rostering problem, the shortest path algorithm is used to compute a single initial roster per crew member (**DRF.4**). This way, the initial rostering problem only contains a very small amount of the feasible rosters – as many as the crew members being considered. Additional rosters are latter added using the column generation algorithm explained in sub-section 3.4. During this rostering generation process, a null cost is given to pairings that were assigned or granted to the specific crew member in the previous time step.

3.3. Rostering problem

The rostering problem is formulated as a set covering problem that ensures that in the final solution all pairings are covered by one crew member. The resulting model is defined as an integer linear programming model.

To formulate this problem, we analysed the rostering and request evaluation process at one airline. We observed that granting request makes the crew schedule less efficient, creating voids in the schedule and requiring more crew members to operate the scheduled flights. The voids in a crew roster are sometimes hard to fill in with other activities. It depends on the length of the voids. If the void is smaller than the length of the pairings used by the airline, it would mean that the void could only be used for other activities than not flying. However, the demand for these other activities depends on the context of the crew member. If, for instance, no training, medical checks, office duties or holidays can be scheduled in these voids, these days will be *lost days* for the airline. This means that the crew member will have a paid day off.

Following these observations, we have considered a few assumptions when formulating the rostering problem:

- The objective of the rostering problem is to minimize the number of necessary crew members needed to operate the schedule. However, as this is not known at the time the requests are evaluated, a proxy can be obtained by computing the past crew costs associated with voids of different lengths. These costs include the costs of additional crew to cover the schedule and the salary associated with lost days.
- The airline can always use additional crew to cover the schedule. That means that, besides the normal number of crew members needed to operate the flight schedule before granting requests, it will be possible to call more crew. These additional crew members will be more costly (a compensation has to be given, either in extra days off or a wage bonus).
- A crew member is equivalent to one full-time equivalent (FTE) employee. Although several crew members work only part-time, we work with the sum of the equivalent FTE's of the airline crew.

The sets, variables and parameters used are summarized next, followed by the formulation of the optimisation problem for a given crew type at a specific time step *t*. Since we do not work with all possible rosters and that new roster are added per iteration of the column generation algorithm, we called this optimisation problem as a restricted master problem (**DRF.5**).

Sets

$L(t)$: set of crew members of the crew type considered at time step t, including additional crew.

$P(t)$: set of pairings to be covered by the crew type considered at time step t.

$P_l(t)$: set of pairings $p \in P(t)$ that were previously assigned to crew member$l \in L(t)$ in the previous time steps and that are either already published or granted.

$R(t)$: set of personalized rosters for all crew members at time step t.

$R_l(t)$: set of personalized rosters for crew member $l \in L(t)$ at time step t.

Decision variables

$x_r^l(t) = 1$: if personalized roster $r \in R_l$(t) is chosen for standard crew member $l \in L(t)$ at time step t; 0: otherwise.

s_p^- : slack variable for pairing p if it is not included in any of the personalized rosters in the initial iteration.

s_p^+ : slack variable for pairing p if it is included in more than one personalized roster in the initial iteration.

Parameters

$(n_v^i)_r^l(t)$: number of voids v in personalized roster $r \in R_l(t)$ for crew member $l \in L(t)$ with a void length of i number of days.

c_v^i : cost of void in the roster with a void length of i number of days.

$c_p^l(t)$: cost of choosing roster $r \in R_l(t)$ to crew member. $l \in L(t)$

M: a large constant.

$e_p^{r,l}(t) = 1$: if pairing $p \in P(t)$ is part of roster $r \in R_l(t)$ for $l \in L(t)$ at time step t; 0: otherwise

M: maximum duration of a void in any of the rosters. $r \in R(t)$

Optimisation problem

$$\text{Minimize} \quad \sum_{l \in L(t)} \sum_{r \in R_l(t)} \left[\sum_{i=1}^{m} (n_v^i)_r^l(t) \cdot c_v^i + c_p^l(t) \right] \cdot x_r^l(t) + M \times \sum_{p \in P(t)} \left(s_p^- + \times s_p^+ \right) \tag{1}$$

Subject to:

$$\sum_{l \in L(t)} \sum_{r \in R_l(t)} e_p^{r,l}(t) \cdot x_r^l(t) + s_p^+ - s_p^- = 1 \quad , \quad \forall p \in P(t) \tag{2}$$

$$\sum_{r \in R_l(t)} x_r^l(t) = 1 \quad , \quad \forall l \in L(t) \tag{3}$$

$$\sum_{r \in R_l(t)} e_p^{r,l}(t) \cdot x_r^l(t) = 1 \quad , \quad \forall p \in P_l(t), \forall l \in L(t) \tag{4}$$

$$x_r^l(t) \in \{0, 1\} \quad , \quad \forall l \in L(t), \forall r \in R_l(t) \tag{5}$$

$$s_p^+, s_p^- \in \mathbf{Z} \quad , \quad \forall p \in P(t) \tag{6}$$

The objective function (1) is the minimization of the sum of the costs of the rosters for all crew members (first term). The formulation of these costs is case-specific and needs to be adapted to the airline crew rostering policy. In Equation (1), we given an example of how to formulate these costs by adopting the policy followed by the reference airline considered in the case study. According to their policy, crew costs can be divided into two types of costs: the cost of leaving voids in the personalized roster of a crew member and the cost of assigning a personalized roster to a crew member. The first type of costs is computed by analysing historical airline data. Such analysis can be used to estimate the effective average loss days

associated with each different void length and the respective cost. The second type of costs is considered to penalise the use of additional crew and to capture some airline operational requirements or collective agreements. For instance, these costs can be used to differentiate salary costs associated with the allocation of pairings to different crews, to define request priorities among different crew members, to include costs of cancelling other predefined crew activities besides flights (e.g., training sessions), and to include fairness considerations when granting requests. That is, when considering crew rankings and fairness, crew members with higher request priority or with fewer requests granted in the past have a lower roster costs than other crews.

A second term is added to the objective function to consider the use of slack variables in the initial iteration(s) of the column generation algorithm. Given that we initiate the rostering problem to only work with a single roster per crew member, not all pairings are necessarily covered in the initial iterations while other pairings are selected more than once. Therefore, slack variables are added to constraints (2); these constraints force each pairing to be included in one and only one personalized roster. The slack variables guarantee that the problem is feasible in the initial iteration. Note that in the implementation only slack variables for the pairings that indeed to be fixed are added to the model.

Constraints (3) ensure that each crew member has in the end a roster (even if it is an empty roster for most additional crew members), while constraints (4) import previous defined assignments of pairings to crew members. The latter constraints are the ones that capture the dynamic sequence of solving the rostering problem sequentially in several time steps, guaranteeing the consistency with previous decisions. Constraints (5) and (6) define the domains of the variables.

This rostering is solved for each time step t in the rostering planning horizon. For the traditional monthly crew rostering strategy (illustrated in Fig. 1), this would mean that the time steps would be separated by 4 weeks and that in each time step we would look into a time horizon of 4 or more weeks, depending on the last end date of the pairings starting in the 4 weeks under analysis.

3.4. Column generation algorithm

The optimisation of the integer linear programming problem in the rostering model is achieved using a commercial optimisation solver (Gurobi Optimization, 2018). To guarantee the global optimal solution, the full set of all the possible rosters for each crew member should be available. In larger-scale rostering problems, hundreds or even thousands of possible rosters may exist for each crew member. Computationally, it is very demanding or even impractical to compute all rosters. Furthermore, since only one roster can be chosen for each crew member, most non-basic decision variables would have a value of zero. A better approach is to use a column generation algorithm (Desaulniers et al., 2005). Following this algorithm, we start with a manageable set of rosters, obtain a feasible problem, and from that point onward determine additional rosters to add in order to improve the incumbent solution. Only rosters (i.e., columns) that have the potential to improve the objective function are added to the rostering model. This algorithm has proven to be very efficient solution technique to solve scheduling problems, in particular airline crew scheduling problems (e.g., Gamache (1999), Cordeau et al. (2001), Kasirzadeh et al. (2017)).

The steps of the column generation algorithm adopted are:

- **DRF.5** – Solve a relaxed version of a feasible restricted master problem – This means in solving the restricted master problem (RMP) assuming that all decision variables are continuous. The feasibility of RMP is guaranteed by computing an initial roster for each crew member (DRF.4) and working with the slack variables in the rostering model.
- **DRF.6** – Obtain the solution and dual variables – From the solution to the relaxed restricted master problem, extract the decision variables values and the values of the decision variables to the dual problem.
- **DRF.7** – Solve the pricing problem for all crew members – The pricing problem represents the generation of new rosters using the shortest path algorithm described above (sub-section 3.2). The costs at the arcs of the time-space network are modified by subtracting the (negative) dual variables associated with constraints (2). A new roster is generated for all crew members.
- **DRF.8** – Check if the stop criterion for adding columns is met – If the cost of any of the newly generated rosters is lower than the cost of the current roster for the crew member, then the roster (column) is added to the restricted master problem and a new iteration as to be followed. Otherwise, if no more negative reduced columns are found, the algorithm stops.
- **DRF.9** – Add columns to the restricted master problem – Modify the restricted master problem by adding the rosters (columns) that priced-out the current rosters. Eliminate slack variables (columns) that are no longer in use.

3.5. Pairings evaluation

Requests are either granted or rejected in the weeks before the rosters are finalized. The correctness of granting a request can only be established after the roster is created to be able to measure its effects. Consequently, in this dynamic rostering framework the evaluation of the pairing request decisions are done a few time steps later.

To measure the effects of all these granted requests, a KPI is introduced that evaluates all granted requests in a given rostering week. The idea is that a good resulting roster is the one that grants a large number of flight requests, while using none or very few additional crew members. Therefore, we used the ratio represented in Equation (7).

$$KPI = \frac{(\#\ \text{granted flight requests})^2}{(\#\ \text{required additional FTE} + 1)^2} \tag{7}$$

The KPI contains two variables: granted flight requests and required additional FTE. The granted flight requests are in the numerator, as a higher amount means a better KPI. Additionally, the required additional FTE is in the denominator as a higher amount means a worse KPI. The value of 1 is added to the equation as the required additional crew members could be 0. A quadratic value of these variables is used to stress the fact that relation between granted flight requests and additional FTE's is not linear – to have an additional granted request for the same amount of additional FTE's is better than increasing the KPI by one unit.

This KPI is given as a score to all requests performed in a time step (assumed to be one week). The isolated effect of an individual granted request on the need for additional FTE is hard to measure. Therefore, all granted requests and additional FTE in a given time step are considered at once. All requests granted in that week are labelled with the resulting KPI value.

3.6. Generate new pairing requests

In the case this dynamic rostering framework is applied in practice, new pairing requests can be retrieved from the airline system. However, to train and test the evaluation algorithm, pairing requests need to be generated for each time step (**DRF.14**). This is a random process that, to better represent the real problem, should respect the flight request conditions followed at the airline and should capture historical trends.

In this study, we have defined two sets of parameters that were used in the generation of new pairings:

- Pairing request probability, $p_{request}$ – Probability of a certain pairing from the set of pairings in a given planning horizon (i.e., week 7 up to and including week 12) being requested. This probability reflects the fact that there are pairings that are much more requested than others.
- Number of pairing requests per crew member, $nl_{request}$ – Number of pairings within the requested planning horizon that are requested by

each crew member in each time stage. This number reflects the fact that there are groups of crew members that submit more requests than others.

These parameters can be estimated using airline historical data. Requests are generated by randomly selecting a crew member with still less requests than $nl_{request}$. The pairing being requested is selected by generating a random value and comparing it with the pairing request probabilities $p_{request}$. In the case that duplicate pairing requests occurs, the process is repeated. A crew member that achieves the number of pairings $nl_{request}$ is excluded from being selected. The process ends when no more crew members have requests to make.

3.7. Flight request evaluation method

In the dynamic framework of Fig. 2, flight requests are evaluated in element **DRF.15**. We propose a simulation-trained Neural Network (NN) algorithm for this evaluation (Schmidhuber, 2015). This is a supervised machine learning technique that is used in the dynamic rostering framework to classify each request. The ones that get a high score should be granted, while requests associated with a low score potential cause high costs to the airline and should be rejected.

In this subsection, we present the NN algorithm adopted followed by the description of the other evaluation methods used as a benchmark. Three alternative methods are used: a random assessment algorithm, a rule-based assessment process, and a static version of the optimisation model. Regardless of the method used, the assessment process follows four steps, represented in Fig. 4:

- **RA.1** – A feasibility check that verifies
  - whether granting a flight request is feasible in the current roster without having to capture the full planning horizon in a time-space network. This feasibility check may guarantee, for example, that not too many requests are granted in a given week.
  - If the crew member is legible for the request submitted or if the request does not violate any collective agreement. For fairness reasons, crew members may be subject to a maximum number of requests submitted and granted per time period (e.g., per year).
- **RA.2** – A feature extraction step that collects data from the state of the roster at the moment of assessing the request.
- **RA.3** – The assessment method that is used to decide whether to grant or decline the flight request.
- **RA.4** – Updating the model for the decision that was made in the flight request assessment.

3.7.1. Neural network algorithm

One of the main contributions of this work is the development of a classification algorithm based on a NN algorithm to classify flight pairing requests in the context of the dynamic rostering framework. This NN is used to learn from previously made decisions on whether to grant or reject a request. By keeping track of specific features when the decision is made and analyzing the final result of the impact on the rosters, the NN could improve the decision-making process compared to the current airline practice. To apply a NN algorithm to the problem, the Scikit Learn MLPClassifier package is used. After several experiments, the best performance was obtained while using a network of 2 hidden layers both with 100 neurons. The ReLU activation function was used and adopted the ADAM learning algorithm, using a log loss function. In addition, five features were selected to be included in the algorithm. These are visualized in Fig. 5 and explained below. In this figure, it can be seen that three of the features focus on the distance between the previously granted request before the new requested pairing, one on the distance after the new requested pairing and one at the requested pairing itself.

- Two features based on the void length between the requested pairing and the previously assigned pairing in the roster for the given pilot. The void length can range from 0 to infinite days, but given the planning horizon we capped this value to a maximum of 44 days. These two features could also be used after the requested pairing and subsequent pairing, but it was found during the experiments that these did not increase the performance of the NN algorithm.
  - **Avg7**: the amount of blocks of 7 days that fit in this void length. The average pairing length is 7 days. For the maximum of 44 days of void length, it means that this feature ranges from 0 to 6 blocks of 7 days.
  - **Rest7**: the rest value of the difference between the void length and blocks of 7 days. For example, for a void length of 20 days, 2 blocks of 7 days fit and there are 6 rest days. Therefore, this feature ranges from 0 to 6.
- Additionally, two features are related to the amount of pairings that would fit before and after the requested pairing if granted.
  - **#PairB**: The log of the amount of pairings that are available and would fit between the last assigned pairing and before the requested pairing. The log value is taken as the value for this amount ranges from 0 to almost 300 pairings (due to 71 pairings per weeks and multiple weeks of requests) and it is more important to go from for example 10 to 20 pairings than from 100 to 150.
  - **#PairA**: The log of the amount of pairings that would fit after the requested pairing and previously assigned pairings up to week 13. This value is even higher than the one above and can get up to 500 pairings as more often there are no pairings assigned yet after the requested pairing. Again, the log is taken to highlight the effect of low values for this amount.
    - Lastly, the length of the requested pairing is used as a fifth feature.

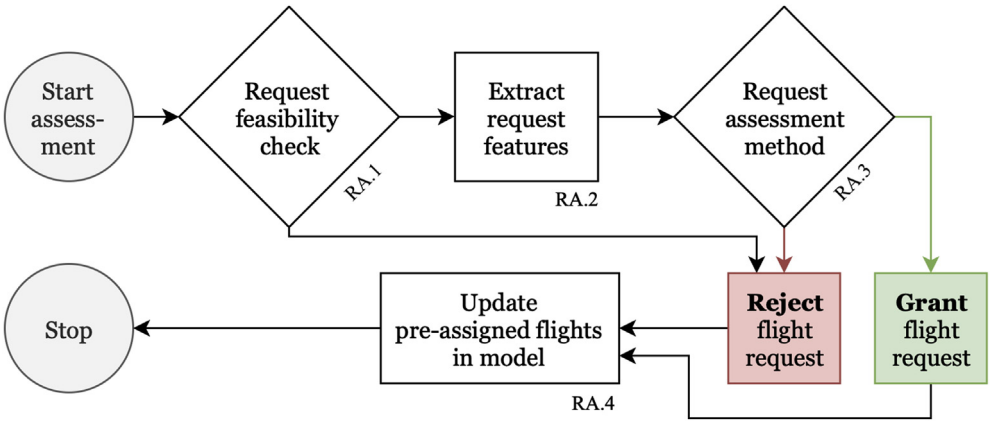

Fig. 4. Flight request assessment algorithm (RA = Request Assessment).

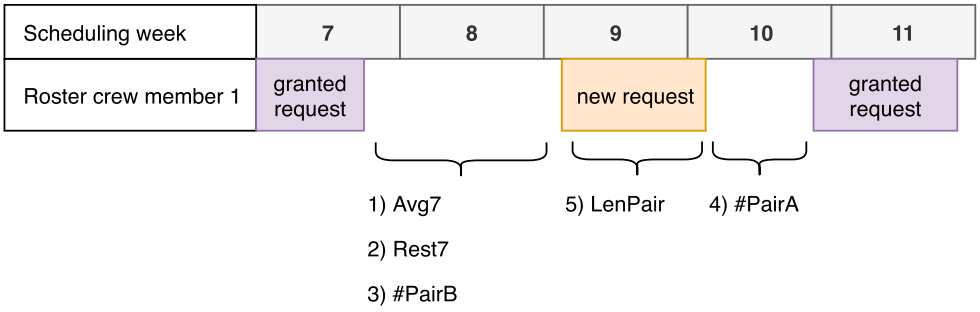

Fig. 5. Illustration of the pairing request features in the context of the rostering problem.

– **LenPair**: The length of the requested pairing. This feature ranges from 5 to 12 days.

Using the features and the KPI presented in subsection 3.5, the NN can be trained. The NN algorithm is trained with the data collected by running the simulation model for many times, with different random generated instances. As a warm-up, the NN is initially trained using data generated by the random algorithm. For the first 500 weeks of rosters generated using the random algorithm, a set of features is stored when decisions are made to grant or reject rosters. Afterwards, the random algorithm is replaced by the NN algorithm, so the network learns from its own decisions.

For each flight request under consideration, the NN provides a score between 0 and 1. Decisions have to be made based on a granting threshold that defines when to grant a request. For example, a granting threshold of 0.5 means that all requests that receive a score above 0.5 will be granted. This threshold is used in the experiments to alter the number of requests granted.

3.7.2. Benchmark methods

Random assessment. The random assessment of requests is based on a probability of granting or not granting the request. Similar to the granting threshold from the NN, this probability defines what should be the value of a randomly generated value to grant a given request. No features of the request are taken into account for this. This probability is also used in the experiments to compare the different algorithms. The random assessment serves as a lower-bound reference when comparing the different algorithms.

Rule-based (airline practice). The rule-based algorithm is based on the method used by the airline from the experiments. The airline uses fixed rules to determine whether to grant or reject a request. These rules are based on certain features of the request. First of all, the length of the void between the requested pairing and the pairings already in the schedule is used. If this length is a multiple of the average pairing length, a request should be granted. There is flexibility to deviate from this length. For example, if the average pairing length is 7 and a flexibility of 10% is adopted, requested pairings can be granted if they that have a void length of 21 ± 2.1 days. This flexibility is a setting of the rule-based algorithm and is used in the experiments. The second feature that is used in this algorithm is the maximum amount of granted requests per day. If this maximum is reached, no more requests are granted.

Static rostering model. The previous methods are compared with a static version of the dynamic rostering framework that eliminates the dynamic element of time in this process. The solution obtained with this method is the optimal situation for the situation that all requests are submitted at the same time and not time-phased. Because we defined the rostering problem as a minimization problem and to grant a request will increase the costs for the airline, the following constraint was added to the model (1)–(3) & (5)–(6):

$$\sum_{l \in L(t)} \sum_{r \in R_l(t)} \sum_{p \in P(t)} e_p^{r,l}(t) \cdot g_p^l(t) \cdot x_r^l(t) \geq q \tag{8}$$

where $g_p^l(t)$ is equal to 1 if pairing $p \in P(t)$ is requested by crew member $l \in L(t)$ at time step t, and 0 otherwise; and q is the minimum desired number of requests to be granted. Note that the constraints capturing the consistency of the dynamic decisions (constraints 4) are not considered in this static model. The static rostering model is used as an upper-bound reference when comparing the different algorithms.

3.8. Boundary conditions

To account for continuity throughout the simulations, the overlap of activities into the current rostering planning horizon is considered using a set of boundary conditions. These boundary conditions are updated at the end of each time step (**DRF.16**) and are reflected in the rostering model (1)–(6) via the set $P_l(t)$, the set of pairings that were previously assigned to a crew member in previous time steps. This set considers the overlap of both the pairings already published (elements **a** in Fig. 1) and the flight pairings request granted in previous time steps (elements **c** in Fig. 1).

Furthermore, the boundary conditions are also considered in the establishment of the individual time-space networks (**DRF.3**). Constraints (4) enforce that previously assigned pairings are kept in the rosters of the crew members. This makes the pricing problem (**DRF.7**) intractable when having more than one pre-assigned pairing per crew member, as discussed in Barnhat et al. (2000) for the multicommodity flow problem. Thus, our approach was to adapt the time-space network per crew member. We have eliminated from the network all parallel pairings in conflict with the pre-assigned pairings. This guarantees that the boundary conditions are respected when generating new paths and it improves the efficiency of the solution technique.

4. Experiments and results

This section describes the experimental set-up, which focuses on the input and settings of the different algorithms used. Additionally, the results of the experiments performed are given and discussed.

4.1. Experimental set-up

The dynamic rostering framework was tested with data from a major European airline. This reference airline has a particular rostering and flight request evaluation process. Requests can be submitted up to 12 weeks before the operation and the rostering problem is solved every week (Fig. 6). The rosters are produced and adapted on weeks 5 and 6 before operations and are published before Week 4, so a rolling monthly roster is provided to each crew member. The airline does not consider prioritisation of the requests according to the seniority of the crew

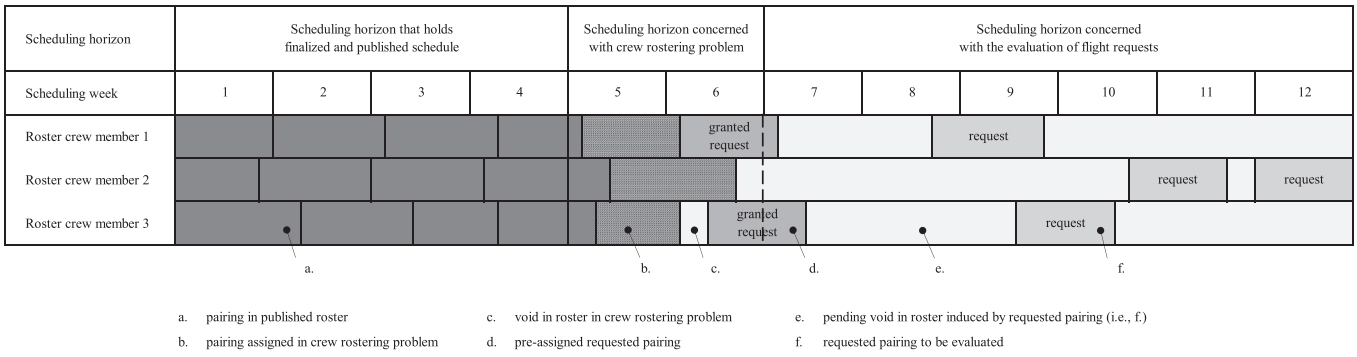

Fig. 6. Crew rostering and flight request evaluation processes from the reference airline.

members. It follows a first-come-first-served priority logic. Fairness is considered by establishing quotas regarding the number of requests requested or granted per crew member. We have adopted a similar strategy in our experiments.

We had access to the cockpit crew scheduling data, including flight requests data, for their long-haul fleet. We used this information to determine the pairings per week and to generate pairing requests probabilities. The pairings generated were composed by the flights, the layover between the flights and the days off after the last flight, according to collaborative agreements. In total, 71 flight pairings and 72 FTE are used. The same flight pairings are repeated in the dynamic rostering context for all the weeks, which is according to the airline practice. The probabilities that flight pairings are requested is determined based on a season of airline request data. In every iteration of the dynamic model a sample is created of pairings that are requested by each crew member.

The rule-based technique based on the airline was set-up so that it would match the performance as measured at the airline. The performance of the flight request process is measured at the airline by comparing the number of requests that are granted per FTE. It is found that the airline uses around 75 FTE per week to cover the flight schedule and, on average per week, 0.219 requests are granted per FTE at the airline. The airline technique should, therefore, be set-up accordingly so that a similar amount of granted requests per FTE are obtained. The right setting for the rule-based technique is to adopt a flexibility of 11%.

The granting probability of the random assessment and granting threshold of the NN algorithm assessment are then also set at various levels to be able to compare the classifier with the other techniques. The random assessment method required a probability of 0.13 to match the same amount of FTE's and a probability of 0.185 to match the same number of average granted requests per week, when compared with the airline. The NN algorithm required a threshold of 0.488 and 0.511, respectively to match the same amount of FTE's and average granted requests per week.

The three algorithms are tested for varying granting thresholds (NN algorithm), granting probabilities (random assessment) and flexibility setting (rule-base). Due to the dynamic nature of the problem, especially the carry-in effects of the rostering problem, the tests are performed for multiple weeks. In total 5 repeats are performed of 115 weeks, from which the last 100 weeks are selected for comparison. The first 15 weeks are used to initialize the model and allow for correct carry-in in the first measured week. This means that 500 weeks are run for each different algorithm setting.

4.1.1. Set-up of the neural network

The NN algorithm was trained for 3500 weeks. Each of these weeks is based on the same flight schedule, pairings and request probabilities. These runs led to 600000 request granting decisions. From these 600000 requests performed, 50000 were granted by the model. These 50000 granted requests were classified using the KPI. Experiments were performed to determine the best threshold to determine whether a request is correctly or incorrectly granted. It was found that using the best 20000 requests based on the KPI as correctly granted and the worst 20000 as incorrectly granted performed best. These 40000 requests are labelled accordingly and provided to the NN. The set of 40000 requests is split in a training set and a test set with 36000 and 4000 requests respectively. After training the NN algorithm with these requests and their accompanying features, the performance of the NN is measured with respect to the test set. The results are given in Table 1. It can be observed that for the test set of 4000 requests an F1-score and accuracy of 60% is obtained. Additionally, the converge of the loss function during NN training can be observed in Fig. 7.

The relative importance of each of the features used in the NN in the classification of a request is determined using Garson's algorithm (Garson, 1991). This algorithm partitions the weights of the hidden layer and output neurons into components of each input feature. The results of this algorithm are given below.

- **Avg7**: 8.6%
- **Rest7**: 39.8%
- **#PairB**: 7.3%
- **#PairA**: 8.5%
- **LenPair**: 35.8%

The **Rest7** and **LenPair** have the highest relative importance according to this method, while the other three features to have lower importance. The **Rest7** importance validates the current rule-based approach followed by the airline, where the main rule is to check if the length of the void is multiple of seven. The importance of the **LenPair** suggests that the number of days blocked when granting a request is relevant for the flexibility of the rostering process – long pairings may congest the roster too much, while small pairings may create long voids.

4.2. Results of experiments

A summary of the results of the experiments can be seen in Fig. 8. Four lines can be observed: static, random, rule-based and NN algorithm. On the y-axis of the graph the average number of granted requests per week is given, while on the x-axis the average required FTEs per week for the 500 weeks is given. For the given schedule under analysis with 0 requests granted a minimum of 72 FTE are required. The static solution increases to 30.5 granted requests for 72 FTE and from that point onward it is found that on average 1.65 additional FTE are required for each additional granted request. The full range for the three algorithms is

Table 1
Performance of the neural network.

True Positives	False Positives	False Negatives	True Negatives	F1-score	Accuracy
1040	987	616	1357	0.60	0.60

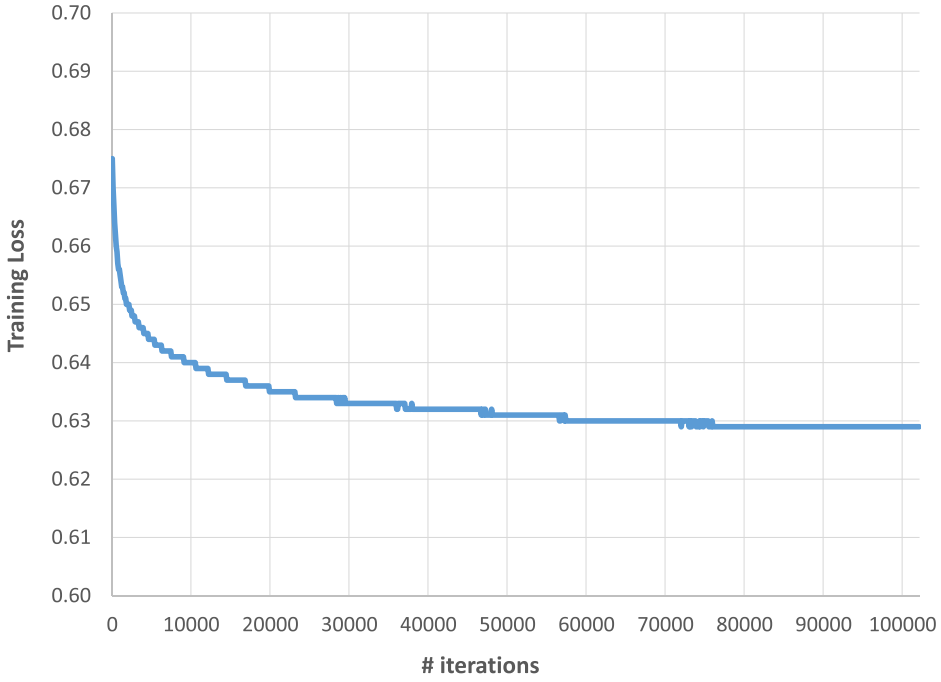

Fig. 7. Loss function for neural network training.

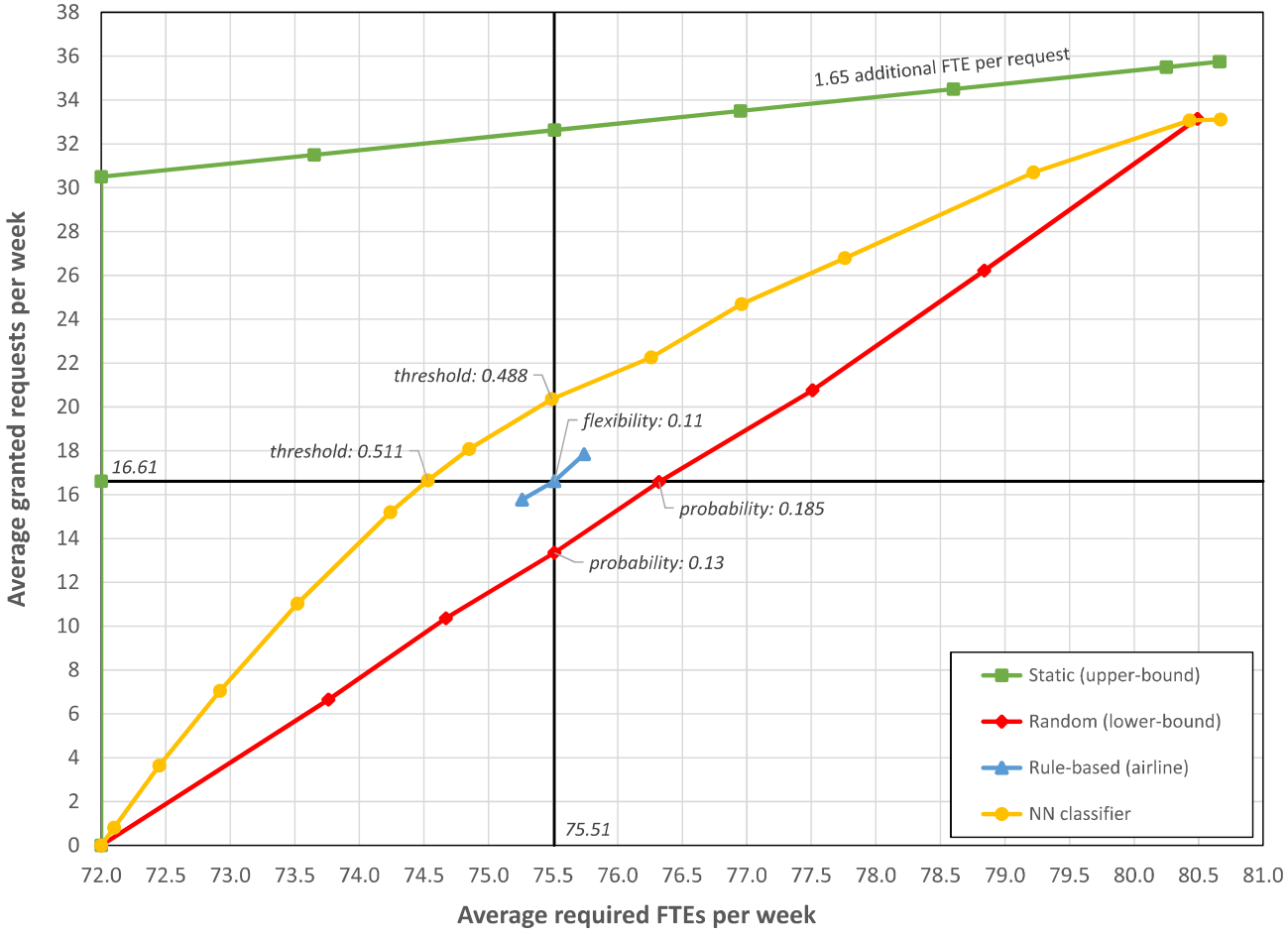

Fig. 8. Results of the three evaluation techniques and static solution.

provided, which means a granting threshold between 0 and 1 for the NN algorithm, a granting probability between 0 and 1 for the random algorithm and no flexibility and maximum flexibility in the airline rule-based method.

To validate the rule-based method, the results are compared with the average amount of granted requests per FTE. This is obtained with a value of 75.51 required FTEs per week and 16.61 granted requests per week. This leads to 0.220 granted requests per FTE. To compare the three algorithms and the static solution, the settings are adjusted so that a similar amount of required FTEs per week or similar amount of granted requests per week is obtained. Having a similar result in terms of required FTEs or granted requests enables the focus on just one of these variables rather than them combined. The settings that lead to these results can be seen in the Figure but also in Table 2. In this table, there are two sets of results displayed. The first set is the results for a similar number of required FTEs, while the second set has a similar amount of granted requests. From both the figure and table, it can be observed that for the same amount of required FTEs the static solution as expected provides the best result with 32.63 granted requests on average per week. The NN classifier leads to 20.36 granted requests, while the airline solution and random solution lead to 16.61 and 13.35 granted requests respectively.

For a similar amount of granted requests, the required FTEs are 72.00 for the static solution. This solution does not need any additional FTEs to be able to grant 16.61 requests. The NN algorithm needs 74.53 FTE, the airline rule-based solution 75.51 and the random solution 76.32 FTE. This means a similar order as with the requested FTEs, with the static solution performing best followed by the NN algorithm, airline and random.

Besides the comparison for similar levels of granted requests and requested FTEs, the trend for different settings of the algorithms can be observed from Fig. 8. The static solution rises from 0 to 30.5 granted requests without needing additional FTEs on top of the 72 FTE. After that, it increases with 1.65 FTE per additional granted request. The NN classifier ranges from a granting threshold of 1.00 (0 granted requests and 72.00 FTE) to 0.00 (33.11 granted requests and 80.67 FTE). It can be observed that with the granting threshold of 0.00 the model is saturated, as the point before is with a granting threshold of 0.10 and leads to a similar amount of granted requests. The performance of the random solution is similar which demonstrates that at that far end of the range the solutions are random due to saturation. For increasing granting threshold, the model needs increasingly less additional FTE for each granted request until it reaches a threshold of 1.00. The airline technique shows a much smaller range in solutions as it has much lower flexibility than the NN algorithm due to the rule-based nature of the technique. Finally, the random solution shows a similar trend as the NN algorithm with a similar start and end, but in terms of performance is below the NN algorithm.

4.3. Discussion of experiments

The results demonstrate the applicability of a NN classification algorithm to determine whether to grant or reject requests. It is found that for similar amounts of required FTEs the NN algorithm performs better than the airline and random techniques in terms of granted requests per week. In a similar fashion, the NN also performs better than the airline rule-based and the random selection methods in terms of the number of additional FTEs required to operate the weekly schedules with a similar amount of granted requests. This better performance is also indicated in Fig. 8, where the performance line of the NN algorithm is always higher than the airline and the random methods, with the exception of the 0 granted requests point.

Table 2
Results with standard deviation of the techniques.

Technique	Required FTEs	Granted requests
Static	75.51	32.63
NN algorithm	75.49 ± 0.05	20.36 ± 0.22
Rule-based	75.51 ± 0.12	16.61 ± 0.17
Random	75.51 ± 0.05	13.35 ± 0.08
Static	72.00	16.61
NN algorithm	74.53 ± 0.15	16.65 ± 0.34
Rule-based	75.51 ± 0.12	16.61 ± 0.17
Random	76.32 ± 0.10	16.58 ± 0.10

The results can also be used to highlight the complexity of the problem. The upper-bound solution shows 32.63 granted requests on average per week for 75.51 required FTEs, following a static representation of the problem at hand. The NN algorithm and the rule-based solution are only able to grant 20.36 and 16.61 requests respectively for a similar amount of required FTEs. In other words, the NN algorithm and airline method are able to grant 62% and 51% of the requests respectively compared to the static solution. This shows that while the NN algorithm performs better than the current airline rule-based method, there is still a large difference between the NN algorithm and the static solution. Similarly, this can be concluded for the amount of required FTEs for 16.61 granted requests. The static solution does not need additional FTEs to solve the problem, while the NN algorithm and the rule-based method followed by the airline uses 2.53 and 3.51 extra FTEs respectively. It can be inferred that (part of) the differences result from the error introduced when simulating the personalized rostering problem with pre-assigned activities as a static problem.

The comparisons are performed around the point of 16.61 granted requests and 75.51 required FTEs, which is chosen as it is a similar performance as the airline's current practice. The results would be similar if this point is chosen differently, but the rule-based method based on the airline has limited flexibility due to the rule-based nature of the technique. For instance, a flexibility level of 100% would mean a random choice.

5. Conclusions

This paper presents the first study in which crew requests are modelled and assessed to build personalized rosters. This a challenging problem, in which the airline needs to make a decision whether to grant or reject the crew requests before the rostering process starts. The impact of the decisions in the resulting schedules is very hard to determine at the time of deciding. To solve this problem we proposed a dynamic rostering framework which combines an integer linear programming rostering model solved with the help of a column generation algorithm with a simulation-trained neural network (NN) algorithm. The NN algorithm classifies each request, providing a score that can help the airline to better select which requests to grant.

The need for a dynamic rostering model over a static model was confirmed in a case study with a major European airline. It was found that the NN algorithm performs better than the current airline practice in terms of both of these measures. On the one hand, for a similar amount of required FTEs, it is found that the NN algorithm is able to grant, on average, more than 22% requests than the rule-based method which follows airline practice. On the other hand, for a similar amount of granted requests, the airline method needs one FTE more than the NN classifier to operate the schedule with the same number of requests granted. It can, therefore, be concluded that the classification algorithm is a viable method for the evaluation of pairing requests.

Furthermore, we have also compared the results from the dynamic approach with the results from a static version of the rostering problem. As expected, our results suggest that more flexibility and higher crew satisfaction comes with a cost. The static formulation, in which all requests are assessed at the same time, may result in 3.5% fewer FTEs being required to operate a schedule with the same amount of requests granted by a dynamic approach. The implementation of a dynamic management solution to manage crew requests has to be a choice of the airline company, after weighting the disadvantages and benefits of such an

approach.

The practical value of the current dynamic rostering framework, and the potential of the NN algorithm, was well demonstrated with the case study of the European airline. However, this work can be further extended to be more suitable for another context. Namely, the framework was tested with data from a single airline and flight season. Although it should work for any other airline with an dynamic assessment of requests, it could be interesting to analyze the results for different scheduling practices and flight seasons, and considering requests not only for flight pairings but also for other activities (such as training sessions or holidays). Another natural extension of this work is to consider pairings with activities besides flights. These pairings are common in practice and they can have an influence on the assessment of the pairings requested. Furthermore, it would be interesting to adapt the framework to the context of short-haul cockpit crew or cabin crew contexts. Additional future work could include the potential use of the proposed framework as an on-line request assessment tool. The airline that supports this study is currently considering this on-line implementation.

Declaration of competing interest

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

Acknowledgements

The authors thank the reference airline for their support and the access to data. In addition, the authors thank to the two anonymous reviewers whose comments helped improve and clarify this manuscript.

References

Achour, H., Gamache, M., Soumis, F., Desaulniers, G., 2007. An exact solution approach for the preferential bidding system problem in the airline industry. Transport. Sci. 41 (3), 354–365.
Ahuja, R.K., Mehlhorn, K., Orlin, J., Tarjan, R.E., 1990. Faster algorithms for the shortest path problem. J. ACM 37 (2), 213–223.
Azadeh, A., Farahani, M.H., Eivazy, H., Nazari-Shirkouhi, S., Asadipour, G., 2013. A hybrid meta-heuristic algorithm for optimization of crew scheduling. Applied Soft Computing Journal 13 (1), 158–164.
Barnhat, C., Hane, C.A., Vance, P.H., 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Oper. Res. 48 (2), 318–326.
Beasley, J., Cao, B., 1996. A tree search algorithm for the crew scheduling problem. Eur. J. Oper. Res. 94 (3), 517–526.
Biskup, D., Simons, D., 2004. Common due date scheduling with autonomous and induced learning. Eur. J. Oper. Res. 159 (3), 606–616.
Boubaker, K., Desaulniers, G., Elhallaoui, I., 2010. Bidline scheduling with equity by heuristic dynamic constraint aggregation. Transp. Res. Part B Methodol. 44 (1), 50–61.
Caprara, A., Toth, P., Vigo, D., Fischetti, M., 1998. Modeling and solving the crew rostering problem. Oper. Res. 46 (6), 820–830.
Christou, I.T., Zakarian, A., Liu, J.-M., Carter, H., 1999. A two-phase genetic algorithm for large-scale bidline- generation problems at delta air lines. Interfaces 29 (5), 51–65.
Cordeau, J.-F., Stojković, G., Soumis, F., Desrosiers, J., 2001. Benders decomposition for simultaneous aircraft routing and crew scheduling. Transport. Sci. 35 (4), 375–388.
Day, P.R., Ryan, D.M., 1997. Flight attendant rostering for short-haul airline operations. Oper. Res. 45 (5), 649–661.
Desaulniers, G., Desrosiers, J., Solomon, M.M., 2005. Column Generation. Springer.
Eltoukhy, A.E.E., Chan, F.T.S., Chung, S.H., 2017. Airline schedule planning: a review and future directions. Ind. Manag. Data Syst. 117 (6), 1201–1243.
Gamache, M., 1998. The preferential bidding system at air Canada. Transport. Sci. 32 (3), 246–255.
Gamache, M., 1999. A column generation approach for large-scale Aircrew rostering problems. Oper. Res. 47 (2), 247–263.
Garson, G.D., 1991. Interpreting neural-network connection weights. AI Expet. 6 (4), 46–51.
Guo, Y., Mellouli, T., Suhl, L., Thiel, M.P., 2006. A partially integrated airline crew scheduling approach with time-dependent crew capacities and multiple home bases. Eur. J. Oper. Res. 171 (3), 1169–1181.
Gurobi Optimization, L.L.C., 2018. Gurobi Optimizer Reference Manual.
Kasirzadeh, A., Saddoune, M., Soumis, F., 2017. Airline crew scheduling: models, algorithms, and data sets. EURO Journal on Transportation and Logistics 6 (2), 111–137.
Kohl, N., Karisch, S.E., 2004. Airline crew rostering: problem types, modeling, and optimization. Ann. Oper. Res. 127, 223–257.
Kotsiantis, S.B., 2007. Supervised machine learning: a review of classification techniques. Informatica 31, 249–268.
Maenhout, B., Vanhoucke, M., 2010. A hybrid scatter search heuristic for personalized crew rostering in the airline industry. Eur. J. Oper. Res. 206 (1), 155–167.
Quesnel, F., Desaulniers, G., Soumis, F., 2019. Improving air crew rostering by considering crew preferences in the crew pairing problem. Transportation Science, Article in Advance 1–18. https://doi.org/10.1287/trsc.2019.0913, 0.
Saddoune, M., Desaulniers, G., Elhallaoui, I., Soumis, F., 2011. Integrated airline crew scheduling: a bi-dynamic constraint aggregation method using neighborhoods. Eur. J. Oper. Res. 212 (3), 445–454.
Saddoune, M., Desaulniers, G., Elhallaoui, I., Soumis, F., 2012. Integrated airline crew pairing and crew assignment by dynamic constraint aggregation. Transport. Sci. 46 (1), 39–55.
Schmidhuber, J., 2015. Deep learning in neural networks: an overview. Neural Network. 61, 85–117.
Smet, P., Bilgin, B., De Causmaecker, P., Berghe, G., 2014. Modelling and evaluation issues in nurse rostering. Ann. Oper. Res. 218, 303–326.
Stojković, M., Soumis, F., Desrosiers, J., 2009. The operational airline crew scheduling problem. Transport. Sci. 39. January:2014–2014.
Suraweera, P., Webb, G.I., Evans, I., Wallace, M., 2013. Learning crew scheduling constraints from historical schedules. Transport. Res. C Emerg. Technol. 26, 214–232.
Zeghal, F., Minoux, M., 2006. Modeling and solving a crew assignment problem in air transportation. Eur. J. Oper. Res. 175 (1), 187–209.
Zeighami, V., Soumis, F., 2019. Combining benders' decomposition and column generation for integrated crew pairing and personalized crew assignment problems. Transport. Sci. 53 (5), 1479–1499.