An aircraft and schedule integrated approach to crew scheduling for a point-to-point airline

Korte, Johanna P.; Yorke-Smith, Neil

DOI
10.1016/j.jairtraman.2025.102755

Publication date
2025

Document Version
Final published version

Published in
Journal of Air Transport Management

Journal of Air Transport Management 124 (2025) 102755

Contents lists available at ScienceDirect

Journal of Air Transport Management

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

An aircraft and schedule integrated approach to crew scheduling for a point-to-point airline

Johanna P. Korte[1], Neil Yorke-Smith [*]

STAR Lab, Delft University of Technology, The Netherlands

ARTICLE INFO

Keywords:
Crew scheduling
Aircraft routing
Optimisation
Branch-and-price
Point-to-point carriers

ABSTRACT

Crew costs make up the second largest expense for airlines, behind only fuel costs. This motivates a potential gain in improving crew efficiency within the bounds set by the law and collective labour agreements. Doing so requires to take into account aircraft routes and crew pairings, and the specifics of the airline's network. This work presents an integrated model for obtaining efficient crew pairings for airlines operating point-to-point networks, while also allowing for flight retiming. By considering simultaneously both crew pairing and constrained aircraft routing, better-performing solutions can be obtained. The greater complexity of the integrated model is addressed by means of a custom branch-and-price approach with a shortest path pricing sub-problem, in order to obtain exact solutions. The results of the integrated model are evaluated on a real-world case of an European low-cost carrier that operates a short-haul point-to-point network. Results show a reduction in crew duties of 10% and an increase in crew efficiency metrics by up to 1.5%, optimising the carrier's complete network of 926 flights over a full week.

1. Introduction

Airline scheduling consists of many different problems that are often solved independently and subsequently. Schedule optimisation must treat, on one hand, a competitive market with uncertain demand and exogenous shocks, and on the other hand, such a sequence of inter-related optimisation problems that connect over temporal and resource granularities: network design, schedule design, fleet assignment, aircraft routing, crew scheduling, and operational execution — besides rescheduling and handling operational dynamics.

The potential for integration of various of these problems is well recognised in the literature (Xu et al., 2023). The state of the art combines for example crew scheduling with one up- or down-stream problem (Sandhu and Klabjan, 2007), and often finds a decomposition or approximation approach necessary (Ben Ahmed et al., 2022; Khiabani et al., 2023).

The literature is less developed on integrated models for aircraft routing and crew pairing that also allow for flight retiming (Cacchiani and Salazar-González, 2020). Further, the literature more commonly treats the US market rather than the specifics of the European market (Erdogan et al., 2015), including its flight network topology and labour regulations. The flight schedules and aircraft and crew constraints for point-to-point airlines – such as operated by European low-cost carriers – exhibit differences from hub-and-spoke carriers (Cook and Goodwin, 2008). Hence the knowledge gap addressed by this article is how the crew pairing problem can be integrated with the aircraft routing problem and schedule optimisation, so as to improve the crew productivity of a carrier operating a point-to-point network.

We develop an integrated mixed-integer linear programming (MILP) model and show how it captures the integrated problem. We overcome the scale and computational complexity of solving the model by developing a branch-and-price decomposition approach. A key technical piece is a pairing graph, over which we can solve a shortest-path pricing sub-problem effectively. Even using a prototype implementation, on week-long schedules with 926 flights, we can reduce the number of crew duties by 10% and increase overall crew efficiency by 1.5%.

This article contributes to the literature as follows:

- A novel integrated model for point-to-point carriers, integrating a pair of cost-sensitive problems in schedule optimisation and allowing flight retiming.
- A custom exact decomposition approach to ensure computational feasibility while excluding approximation.
- Empirical evidence of the practical value of this approach on real data of the short-haul network of a European low-cost carrier.

The remainder of the article is structured as follows. Section 2 provides background on the problems addressed and related efforts

[*] Corresponding author.
E-mail address: n.yorke-smith@tudelft.nl (N. Yorke-Smith).
[1] Currently at KLM Royal Dutch Airlines.

https://doi.org/10.1016/j.jairtraman.2025.102755
Received 6 March 2024; Received in revised form 16 November 2024; Accepted 28 January 2025
Available online 14 February 2025

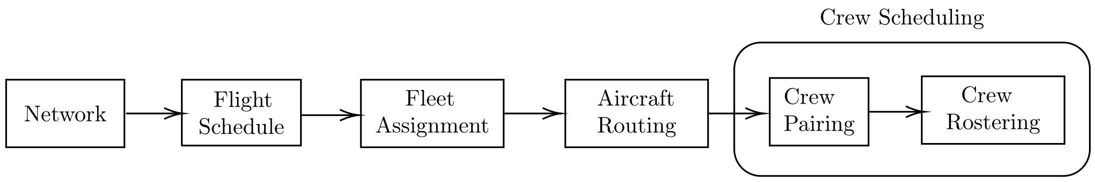


Fig. 1. Overview of the airline planning process.

in the literature. Section 3 presents the integrated model and the incremental solving approach. Section 4.3 reports an empirical study in the case of a European point-to-point carrier. Section 5 concludes with remarks for future work.

2. Background and related work

The first phase in airline scheduling is *network development*, where the desired origin–destination pairs and their corresponding frequencies are determined. It is then possible to construct a *flight schedule*, where each origin–destination pair gets scheduled at specific days and times. Subsequently, using demand forecasts, the *fleet assignment problem* is solved. In this problem, all flights in the schedule are assigned an aircraft type. The output of the fleet assignment problem will be the input for the fourth stage, the *aircraft routing problem*, where specific aircraft registrations are put on each flight, taking into account maintenance requirements. The last step of the process is scheduling crew such that all flights can be operated. This is called the *crew scheduling problem*. Because crew scheduling is NP-hard, it is usually solved in two phases: *crew pairing* and *crew rostering*. An overview of the entire airline scheduling process can be seen in Fig. 1. This article will focus primarily on the aircraft routing and crew pairing phases.

The crew pairing phase aims to generate a set of crew duties, called *pairings*, that will cover all flights in the schedule, and minimise the cost (discussed below) of the set of pairings. A pairing is a sequence of assignments that begins and ends at the same base, and respects the law and all labour agreements between the airline and its labour unions (Golden and Erne, 2022). Besides flights, also called *legs*, these assignments can also include positioning by car, taxi or aircraft, and hotel layovers. Positioning by aircraft is also called *dead-heading*, and means that a crew member will be transported as a passenger (Deveci and Demirel, 2018b). Additionally, all pairings that contain a flight leg will include a briefing and debriefing period. If two flight legs are scheduled immediately after each other, a minimum *turnaround time*, needed for re-fuelling, (de)boarding, and other flight preparations needs to be factored in. Fig. 2 depicts three example pairings, with and without positioning activities and layovers. Once a set of pairings has been found that covers all flights and minimises the cost, the pairings are used as input for the crew rostering problem. Here, the pairings, free time, and other required activities are assigned to individual crew members, while aiming to distribute the workload (fairly) over all crew members (Deveci and Demirel, 2018b). This results in a roster for each crew member, and completes the airline's schedule.

2.1. Crew scheduling

Crew costs are an airline's second highest expense, being only less expensive than fuel (Wen et al., 2020, 2021). The motivation for the present research was an industrial partner, a leading European low-cost airline. This airline, which remains anonymous for commercial reasons, does like many European airlines, pay cockpit crew irregardless of the number of hours they fly. Because of this pay structure, it is desirable to assign crew to the flight schedule as efficiently as possible. Currently, many airlines solve their various scheduling problems, among which the aircraft routing and crew pairing problems, independently and subsequently (Xu et al., 2023). Because the result of the first problem (aircraft routing) serves as the input for the second (crew scheduling), as seen in Fig. 1, integrally solving them could potentially provide better solutions. The challenge, however, is the greater complexity of integral solving, which demands thoughtful modelling and algorithmic approaches (Sandhu and Klabjan, 2007; Xu et al., 2023).

As stated, since the crew scheduling problem is NP-hard and its size can be challenging for real-world instances, the problem is often solved in two phases. In the crew pairing phase, a set of optimal sequences of workdays (pairings), to be executed by the crew are determined such that all flights are covered; while the individual crew schedules are determined in the crew rostering phase by assigning these pairings to crew members. Although individual crew members only get assigned in the crew rostering phase, the cost-determining phase of the crew scheduling problem begins with the pairing problem (Kohl and Karisch, 2004; Wen et al., 2021). This is because inefficient covering of flights by pairings leads to excessive crew numbers during the rostering.

2.1.1. Objectives in crew pairing

The objective of the pairing problem is highly dependent on how the airline in question operates. In general, studies aim to minimise the cost of executing a set of pairings, as defined by fixed and variable crew salary components, crew transportation costs, and the costs of layovers (Azadeh et al., 2013; Cordeau et al., 2001; Erdogan et al., 2015; Ozdemir and Mohan, 2001). Others also include cost penalties based on time away from base and sitting time (Deng and Lin, 2011; Mercier and Soumis, 2007). Especially with US airlines the concept of *pay-and-credit*, also known as *excess cost*, is frequently used. The pay-and-credit (PaC) for a pairing p containing f_p flight hours, with a minimum of g guaranteed hours is calculated according to Eq. (1). Another frequently used and similar metric for these airlines is the *flight time credit* (FTC), given by Eq. (2) (Chu et al., 1997; Klabjan et al., 2001). The use of these objectives can be explained by the structure of crew costs for North American airlines. As there is a minimum guaranteed number of paid hours, it is advantageous to first use these hours, before generating pairings that have substantial excess cost.

$$PaC(p) = \max(g - f_p, \quad 0) \tag{1}$$

$$FTC(p) = \frac{PaC(p)}{f_p} \cdot 100 \tag{2}$$

In contrast, many European airlines hire pilots on fixed salary contracts, and thus do not know the concept of excess cost and FTC. Instead, objective functions for these problems are often designed to minimise the total costs of all pairings as described above (Desaulniers et al., 1997; Deveci and Demirel, 2018a; Erdogan et al., 2015; Zeren and Özkol, 2016), or in some cases to minimise the total number of pairings in the final solution (Agustin et al., 2017). Note that works which minimise the total number of pairings do not necessarily obtain solutions that minimise the number of crew members.

A selected overview of studies investigating the crew pairing problem, their respective data sources, and used objective functions can be found in Table 1. Surprisingly, quite little is currently known about the relationship between the value of the objective function in the pairing problem and the performance of the resulting set of pairings in the subsequent rostering problem. This is important to consider, as the output of the pairing phase will be used as the input for the rostering phase. Indeed, when rostering crew, the first pairing of a crew member's sequence might have an effect on their availability for subsequent

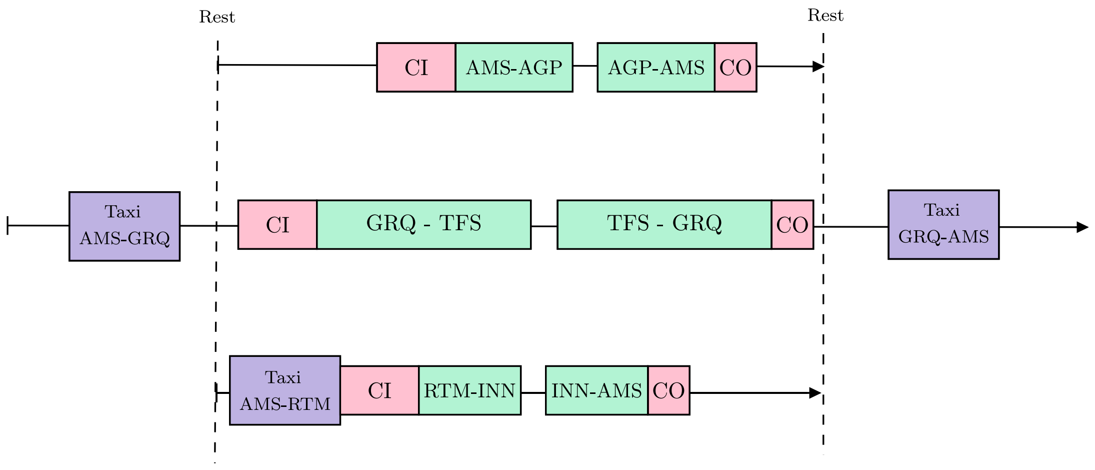


Fig. 2. Examples of three different pairings, depicted as the three horizontal rows. Flight activities are coloured green, while the check-in and check-out activities are shown in red, and the taxi activities are shown in blue. The dashed lines indicate the location of a rest period. The upper and lower example both span 1 day, while the middle pairing spans 3 days. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Formulation of the objective function across selected previous studies, studying only the crew pairing problem and sorted by data source.

Year	Authors	Data	Objective function formulation
1997	Chu et al. (1997)	American Airlines	Minimise pay-and-credit
2001	Klabjan et al. (2001)	United Airlines	Minimise pay-and-credit
2003	Klabjan et al. (2003)	United Airlines	Minimise pay-and-credit and cost of deadheads, maximise repetition in schedules
2013	Saddoune et al. (2013)	Major US airline	Minimise pay-and-credit, pairing duration, duty duration and deadheads
2017	Quesnel et al. (2017)	Major US airline	Minimise pairing costs (duty length, worked time)
2020	Quesnel et al. (2020)	Major North American airline	Minimise pairing cost modulated by crew preferences
2022	Ben Ahmed et al. (2022)	United Airlines	Maximise total profit
2023	Khiabani et al. (2023)	American Airlines	Minimise total cost after flight disruption
1997	Desaulniers et al. (1997)	Air France	Minimise total cost
2015	Erdogan et al. (2015)	European airline	Minimise crew-related costs (deadheads, layovers and ground transportation)
2016	Zeren and Özkol (2016)	Turkish Airlines	Minimise operating costs (cost per duty day, deadhead hour, layovers)
2017	Agustin et al. (2017)	European airline	Minimise number of pairings
2018	Deveci and Demirel (2018a)	Local Turkish airlines	Minimise pairing costs (duty costs, rest expenses, connection time expenses)
2020	Chen et al. (2020)	Asian airline	Multiple objectives about flight characteristics after disruption
2023	Li et al. (2023)	Chinese airline	Minimise number of pairings
2001	Ozdemir and Mohan (2001)	Multiple airlines	Minimise total costs (pay-and-credit, deadheads, sitting time, layovers)
2013	Aydemir-Karadag et al. (2013)	Randomly generated	Minimise total pairing cost (pay-and-credit, time away from base and total duty cost)
2013	Azadeh et al. (2013)	Randomly generated	Minimise total crew cost (flight payments, rest expenses, deadhead costs)

pairings. This is especially the case with very long or late duties, which require longer rest periods in between pairings. Consequently, a set of pairings might minimise operating cost or require fewer pairings to cover all flights, but might not be desirable in the rostering phase. This is especially the case for airlines that hire flight crew on fixed salaries, as no costs are associated with individual pairings, but rather with the required number of crew members. In the case of the airline partner motivating our work, the total cost is defined by the crew's available duty time that is not spent on flying, i.e., the unused hours within and between duties. This cost depends on the routings (being the specific flights in the schedule) and the pairings for them (being the crew tasks to be covered for them).

2.1.2. *Crew planning problem horizon*

Variants of the crew pairing problem exist with different problem horizons. Due to the hardness of the problem and its rapidly increasing size, many studies solve a *daily* problem where the flight schedule that serves as input contains all flights for an airline on one day (Azadeh et al., 2013; Ben Ahmed et al., 2018; Mercier et al., 2005).

Second, another common approach is that of the *weekly* problem (Klabjan et al., 2003). The daily and weekly problems have as an advantage that a single problem can be solved first, after which the solution is copied onto the next day or week, after which any remaining inconsistencies are solved. These kinds of approaches are therefore very suitable for airlines that have repetitive schedules, something that is mostly seen in hub-and-spoke networks.

In many cases, the daily problem is solved first, and it is assumed that the resulting solution can be repeated seven times, such that an entire week is covered. Then, an 'exceptions' problem is solved, where the infeasible pairings – resulting from a not completely repetitive schedule – are repaired. Although this approach is time-efficient compared to solving an entire week, it results in a lot of dead-heading for the crew (Kenan et al., 2018; Klabjan et al., 2001, 2003). Not all authors take the approach of repeat-daily-then-repair: Klabjan et al. (2003), instead of repeating the daily problem, formulate a weekly problem with regularity variables. On the data of United Airlines, this resulted in crew pairings that were more regular, contained fewer dead-heads and had a lower FTC.

Third, far fewer studies aim to solve a *monthly* problem. Erdogan et al. (2015) solve a monthly problem by using a heuristic approach consisting of integer programming, enumeration and large neighbourhood search, while Saddoune et al. (2013) applied a rolling horizon technique.

Lastly, for schedules without a repetitive nature at all, a dated problem can be solved. However, this kind of problem is relatively rare in literature, as few of the investigated airlines have variable schedules (Butchers et al., 2001).

2.2. Solving the crew pairing problem alone

From the first studies onwards, a vast majority of research has formulated the crew pairing problem as either a set partitioning or a set cover problem (Erdogan et al., 2015; Klabjan et al., 2001; Rubin, 1973; Zeren and Özkol, 2016). These formulations enforce all flights to be covered by a pairing, and can easily be extended to capture the scope of the problem by adding additional constraints. To solve the pairing problem as a set cover or set partition problem, a set of candidate pairings is required. This set can be generated by enumeration or with the use of heuristics. Instead of generating the pairings in advance, it is also possible to generate pairings during the solving process by using column generation.

Traditionally, the pairing problem has been solved optimally with the help of column generation and branch-and-bound (Hoffman and Padberg, 1993; Kasirzadeh et al., 2017; Yan et al., 2002; Zeren and Özkol, 2016). Because the crew pairing problem can be formulated as a binary integer programming problem, column generation is often used to solve the relaxed problem, while branch-and-bound-price is used afterwards to obtain the optimal integer solution from the relaxed solution. A typical example is Yan et al. (2002), who solved the pairing problem for cabin crew, while also taking into account different cabin classes, mixed-aircraft types, and home bases. Yan et al. formulated the crew pairing problem using flight networks, similar to Deng and Lin (2011) and Ozdemir and Mohan (2001).

Aiming to get good although not optimal pairings for realistic problem sizes in less time, meta-heuristic and more recently math-heuristic methods are employed (Xu et al., 2023); and in the last few years, the use of data-driven optimisation (Quesnel et al., 2022). Classic examples of meta-heuristics are the use of evolutionary algorithms (Ozdemir and Mohan, 2001), variable neighbourhood search (Agustin et al., 2017) and ant colony optimisation (Deng and Lin, 2011). Deveci and Demirel (2018b) used memetic algorithms, a combination of local search and genetic algorithms, to solve the crew pairing problem. Deveci and Demirel compared their algorithm to meta-heuristic algorithms by Beasley and Chu (1996) and Zeren and Özkol (2012), and found that the memetic algorithm achieved better solutions while runtimes remained similar. Wen et al. (2020) study crew pairing under robustness metrics, while Wen et al. (2021) survey crew scheduling approaches with focus on robustness in face of operational changes. Quesnel et al. (2020) and Quesnel et al. (2022) allow for crew preferences.

Two advantages of solving the entire crew scheduling problem (and pairing and rostering) at once is that it eliminates the need for a separate pairing objective function, which can be hard to design to resemble reality, and that it allows for a more optimal global solution. An early example is Saddoune et al. (2011), who solved the entire crew scheduling problem using a combination of column generation and dynamic constraint aggregation. A notable cost saving of approximately 4% are reported on real-life instances. Unfortunately, the study does not take into account pre-assigned activities, such as safety training, simulator assignments or holidays. A more recent example is Zeighami et al. (2020), who address crew pairing and crew assignment together by means of a Lagrangian integrated approach. Like our work, the authors give attention to specific constraints. They do not consider aircraft routing, however.

Li et al. (2023) bring Monte Carlo tree search to crew pairing, in a simulation–optimisation approach in conjunction with MILP. Their objective relates to the number of pairings and reducing overall computational overhead; results are demonstrated on a simplified pairing problem of a Chinese airline.

2.3. Integrated problems

The sequential solving of different airline planning problems (Fig. 1) might deliver optimal results for each individual problem, but is not likely to produce a solution that is optimal for the entire airline planning problem (Sandhu and Klabjan, 2007). This is due to the interdependence of the problems: the output of one problem acts as input for the next. As a result, the interest grows for integrated scheduling, where different planning problems are solved simultaneously (Xu et al., 2023).

An important line of work on integrated scheduling is combining the crew pairing problem with the aircraft routing problem. This article continues in this direction. By determining the crew pairings and aircraft routes at the same time, it is possible to generate new pairings that were illegal before due to the extra time and costs associated with an aircraft change mid-pairing (Cacchiani and Salazar-González, 2013; Sandhu and Klabjan, 2007).

Papadakos (2009) provides an example of how the crew pairing problem can be fully integrated with the maintenance and aircraft routing problems. Because of the large number of constraints, Papadakos applies Benders decomposition and combines this with a column generation strategy that is accelerated by using heuristics. The author presents results on hub-and-spoke networks, and does not consider flight retiming.

Ben Ahmed et al. (2022) integrate fleet assignment, aircraft routing and crew pairing, with attention to robustness. Their approach is a combined MILP model, solved approximately by using local search around the linear relaxation. The authors present large-scale computational results on a US carrier, for a single day of operation.

Mercier and Soumis (2007) develop a MILP model and a solution approach combining Benders decomposition, column generation and “a dynamic constraint generation procedure”. Solving the integrated aircraft routing and crew pairing problem, and allowing retiming of flights by ± 5 min, the authors are able to improve solution quality over the sequential, fixed-time solutions of two unspecified airlines (with hub-and-spoke networks, it is assumed).

Chen et al. (2020), also like our work, consider integrated aircraft routing and crew pairing, but with a focus on rescheduling. The authors propose a multi-objective evolutionary approach. Khiabani et al. (2023) also consider integrated aircraft routing and crew pairing, for the situation of recovery after disruption. The authors adopt a Benders-based decomposition, and present large-scale computational results on a US carrier, for a single day of operation.

Limited research considers incorporation of schedule changes within crew pairing models. When the schedule is fixed, many useful pairings might not be generated because they violate a constraint by a small margin. By allowing changes to the initial flight schedule, it is possible to facilitate the design of more convenient pairings. An example of a small schedule modification is retiming, as we have noted above for several studies: adjusting the departure and arrival time of a flight by a few minutes (Pothos et al., 1994; Cacchiani and Salazar-González, 2020).

Cacchiani and Salazar-González (2020) propose an entirely integrated model for the crew pairing, aircraft routing and fleet assignment problems including retiming. The MILP formulation includes retimed copies for all flight legs, and constraints to make sure only one of the original or copies is chosen for each flight leg. Given the integrated formulation, the relaxed problem is first solved, after which one of four proposed heuristics is used to reach a final (inexact) solution.

Although the (integrated) crew pairing problem has been studied extensively in the past, the focus has mostly been on US network carriers, hub-and-spoke networks, and minimising pay-and-credit. The case of European low-cost carriers and optimising crew efficiency for point-to-point networks is the concern of the present work.

3. Methodology

This section provides a model for the integrated crew pairing and aircraft routing problem, allowing schedule retiming. First in Section 3.1 we introduce the data structures. The model is formulated as a MILP in Section 3.2. Section 3.3 explains how to derive solutions to the model using branch-and-bound.

3.1. Data structures

The MILP model will minimise the combined costs of the chosen pairings and routes. Its constraints will ensure flights are uniquely covered, flight combinations are feasible, aircraft numbers by type are not exceeded, and departure times are aligned in the pairing and routing problems. Before giving the precise mathematical model, we explain the auxiliary data structures required for the model and its solving. These are four, relating to the crew duties, the flight pairings, the flight routing, and the allowable short connections. We also explain the way in which flight retiming is incorporated.

3.1.1. Feasible duties generation

The aim of the duty generation process is to find all possible sequences of flights that can legally be operated by a crew member in one workday. This information is necessary for the crew scheduling. The resulting sequences of flights are called *duties*.

As explained in Section 2, crew pairings are made up of working days, called duties. While a pairing needs to start and end in the same base, a duty can start at any airport and end at any airport. Both pairings and duties need to abide by a number of complex rules concerned among others with the number of flight duty hours, number of duty hours, and starting time. For example, for some airlines, a duty that contains more than 13 flight duty hours may contain at most 3 landings, and if a duty begins before 04:30 local time, no more than 4 sequential flights may be assigned (Cao, 2017). To be able to generate pairings during the solving process using branch-and-price, instead of generating a priori, it is desirable to represent the possible contents of a pairing in a network.

As pairings consist of duties, which in their turn consist of flights, it is possible to construct this network of either flights or duties. Seeing that the combination of pairing and duty related rules is very complex to represent in a graph consisting of flights, we take the approach to first generate duties, which will afterwards be used to construct the graph. Specifically, we generate the duties by enumeration over a pruned search-tree, rather than using heuristics. This was done to ensure that the process results in a set of all feasible duties, which is desirable because the crew pairing problem is later formulated as a set partition, where every flight needs to be covered exactly once. Even though all duties are generated by enumeration, the process takes at most a few seconds for the tested instances.

In more detail, to generate the duties, a search tree containing all flights is enumerated in a breadth-first manner. The first layer of the search tree consists of all flights in the schedule. At the beginning of the process, a stack that contains all nodes that still need to be explored is created. This entire first layer of nodes is added to the stack. Then, an iterative process takes place that terminates when the stack is empty. This process includes getting the first element, consisting of a path in the search tree, off the stack, and generating a duty including check-in and check-out activities based on the path. If this duty is legal, it will be appended to the list of all legal duties. Additionally, all flights that can be used to further extend the duty will be added as nodes in the tree. These nodes will be children of the last node of the path popped in the first step of the process. The last step of the iterative process is to add the paths from the root to the newly created nodes to the stack of unexplored options. Whenever a duty is not legal, no child nodes will be created, resulting in the branch to be pruned and not expanded upon further. Algorithm 1 gives an overview of the process.

Example 1. Tables 2, 3 and 4 give examples of the duties that result from this approach. The columns indicate the start time of each activity, the IATA code of the departure airport, the activity, the IATA code of the arrival airport, the end time of the activity, and if applicable the corresponding aircraft type. Table 2 gives an example of a duty that starts and ends in the same base, and consists of four activities: a check-in, a flight from Amsterdam to Hurghada and back, and a check-out. Both flights are operated on a Boeing 737–800 aircraft. In contrast, Table 3 shows a duty that starts and ends in two different *outstation* (i.e., non-base) airports. Again, this duty contains a check-in, check-out and two flights. The difference here is that an aircraft change takes place in Amsterdam, where the crew will switch from a Boeing 737–800 to a Boeing 737–700. Lastly, Table 4 shows a duty that contains only one flight, while departing from a base, and arriving in an outstation.

Algorithm 1: Duty generation

```
1  t ← new Tree
2  unexplored ← []
3  duties ← []
   // all flights in the schedule are the first
      layer of children
4  for flight in schedule do
5      flight_node ← new child node of t.root containing flight
6      unexplored.append(flight_node)
7  end
   // breadth-first search
8  while unexplored is not empty do
9      node ← unexplored.pop()    // pop the first element
        of the stack
10     duty_sequence ← path from node to root with added
        check-in and check-out activities
11     if duty is legal then
12         duties.append(duty)
13         adjacent_flights ← list of flights that can be performed
            after last flight
14         for a ∈ adjacent_flights do
15             node.add_child(a)
16             unexplored.append(a)
17         end
18     end
19 end
```

Table 2
Example of a duty starting and ending in the same base.

01/03/2019 13:10	AMS	Check-in	AMS	01/03/2019 14:10	
01/03/2019 14:10	AMS	HV583	HRG	01/03/2019 19:05	B737-800
01/03/2019 20:00	HRG	HV584	AMS	02/03/2019 01:40	B737-800
02/03/2019 01:40	AMS	Check-out	AMS	02/03/2019 02:10	

Table 3
Example of a duty starting and ending at outstations, containing an aircraft change for crew.

02/03/2019 10:40	LPA	Check-in	LPA	02/03/2019 11:40	
02/03/2019 11:40	LPA	HV5664	AMS	02/03/2019 16:20	B737-800
02/03/2019 18:25	AMS	HV5201	MUC	02/03/2019 19:55	B737-700
02/03/2019 19:55	MUC	Check-out	MUC	02/03/2019 20:25	

Table 4
Example of a duty with only one leg.

02/03/2019 05:30	AMS	Check-in	AMS	02/03/2019 06:30	
02/03/2019 06:30	AMS	HV6901	DXB	02/03/2019 13:30	B737-800
02/03/2019 13:30	DXB	Check-out	DXB	02/03/2019 14:00	

3.1.2. Pairing graph construction

Second, given the set of all legal duties, a *pairing graph* consisting of duties and base nodes is constructed, such that a path from a source base node to a sink base node represents a legal pairing. The optimisation problem is which pairings to choose. Fig. 3 gives an example of such a graph. Intuitively, the pairing graph will capture which activities can be done in sequence (via its edges) and also how

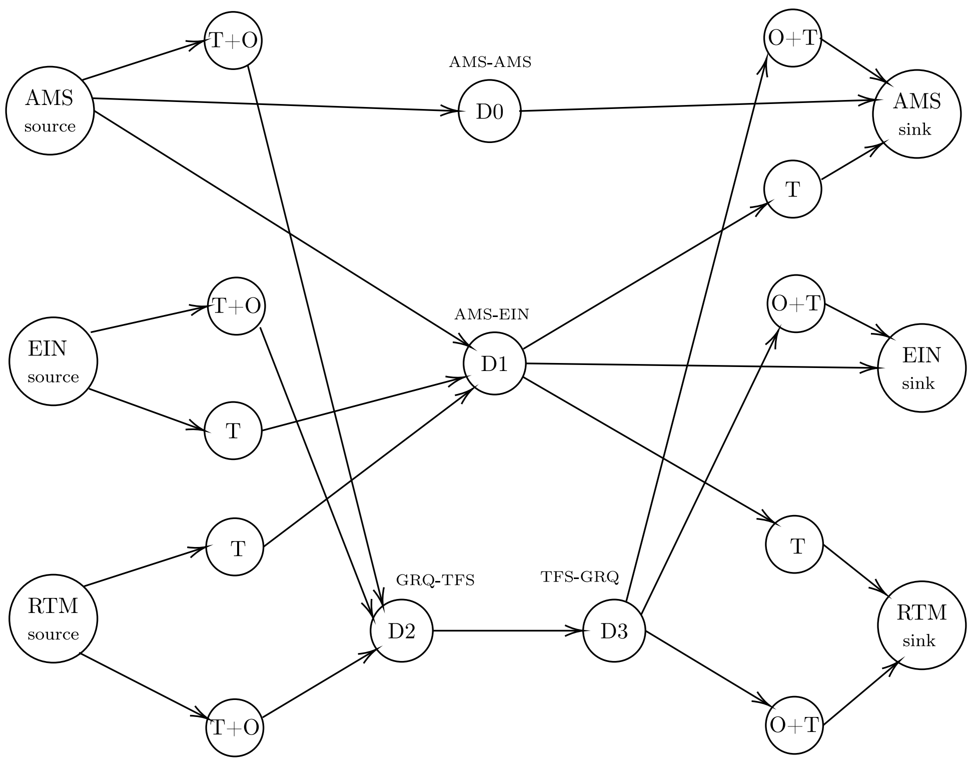


Fig. 3. Example of a pairing graph containing 4 duties (D0–D3), 3 bases (AMS, RTM, EIN), their respective taxi nodes, indicated with 'T', and taxi+overnight nodes, indicated with 'O+T' and 'T+O'. Unused taxi and taxi + overnight stay nodes are not pictured for image clarity.

preferable would be doing so (via its edge weights).

In more detail, the pairing graph is a directed acyclic graph consisting of source and sink nodes (bases), nodes representing taxi movements, and nodes representing the duties from the duty generation step. The presence of an edge between nodes indicates that the activities can be performed sequentially. This resulting graph has $O(n)$ nodes and $O(n^2)$ edges.

In the graph, a path between the source and sink nodes *of the same base* then gives a pairing. To later enforce a maximum pairing duration, the edges have a resource cost attribute, which indicates the time needed to perform the activities in a sequence. The edges also have a weight attribute. The weights are assigned based on the time that the crew is on duty, but is not operating a flight. In this way, a connection between two duties without a lot of excessive rest in between is preferred to a connection that contains much idle time. In addition, the use of taxi's is discouraged by penalising the edge with additional weight. In this way, a pairing that does not contain any positioning activities, is preferred to the same pairing with additional taxi activities. The cumulative weight of the edges in a path from sink to source gives the cost of the pairing to be used in the objective function.

The graph is constructed by first creating individual nodes for all duties, all sources and all sinks. Additionally, two taxi nodes are created for each base. One of these nodes indicates the possibility of a pairing starting with a taxi commute from the base, while the other indicates the possibility of ending a pairing with a taxi commute to the base. Similarly, two nodes per base are created representing the combination of a taxi plus overnight stay.

The rules for the pairing graph are the following:

- All duties starting from a base receive an incoming edge from the corresponding base source node, and all duties ending in a base receive an outgoing edge to the corresponding base sink node
- All duties starting from a base receive an incoming edge from the taxi nodes to all other bases; except for when the duty starts and ends in the same base or the combination does not respect all rules.
- All taxi + overnight nodes corresponding to all other bases receive an edge to a duty only if adding a taxi does not respect all rules; except for when the duty starts and ends in the same base or the combination does not respect all rules.
- All duties ending in a base receive an outgoing edge to all the taxi nodes corresponding to all other bases; except for when the duty starts and ends in the same base or the combination does not respect all rules
- All duties ending in a base receive an outgoing edge to all the taxi + overnight nodes corresponding to all other bases if adding only a taxi does not respect all rules; except for when the duty starts and ends in the same base or the combination does not respect all rules
- All duties that can be performed in sequence with another duties receive an edge from the first duty to the following duty.
- All taxi nodes and taxi + overnight nodes receive an edge to their corresponding source or sink node

Example 2. An example of a pairing that can generated by using the graph from Fig. 3 is given by the path,

AMS source $\longrightarrow$ $T+O$ $\longrightarrow$ $D2$ $\longrightarrow$ $D3$ $\longrightarrow$ $O+T$ $\longrightarrow$ AMS sink

and translates to the pairing of Table 5.

Further details about the pairing graph construction are provided by Appendix A.

3.1.3. Routing graph construction

Third, for *each* aircraft type, a *routing graph* is constructed to represent aircraft routes. While the routing graphs are similar to the pairing graph, they however consist of base nodes and flight nodes. The edge weights of these graphs are formulated to represent the time that the aircraft is not flying. The optimisation problem here is which routings

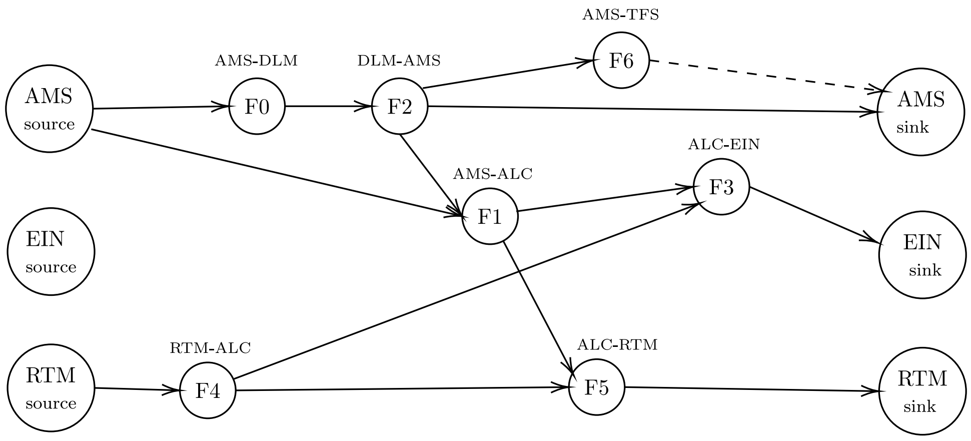


Fig. 4. An example of a flight routing graph, with 3 bases and 7 flights, for one aircraft type.

Table 5
Data for example pairing graph.

07/03/2019 17:30	AMS	Taxi Check-in	AMS	07/03/2019 17:45
07/03/2019 17:45	AMS	Taxi	GRQ	07/03/2019 20:00
07/03/2019 20:00	AMS	Hotel	GRQ	08/03/2019 08:30
08/03/2019 08:30	GRQ	Check-in	GRQ	08/03/2019 09:30
08/03/2019 09:30	GRQ	HV5775	TFS	08/03/2019 14:15
08/03/2019 15:00	TFS	HV5776	GRQ	08/03/2019 19:40
08/03/2019 19:40	GRQ	Check-out	GRQ	08/03/2019 20:10
08/03/2019 20:10	AMS	Hotel	GRQ	09/03/2019 09:10
09/03/2019 09:10	GRQ	Taxi	AMS	09/03/2019 11:25

to choose. Fig. 4 gives an example of such a graph.

In more detail, to generate aircraft routes, an approach similar to that used for pairing generation is used. In contrast to the pairing generation, however, there are fewer rules when generating aircraft routes. The turnaround times should be long enough, and the route should only include flights that can be operated by the same aircraft type, as each route represents one aircraft over (often) multiple days. To accomplish this, a separate directed acyclic graph for each aircraft type is constructed. The graph consists of source and sink nodes for each base, and nodes representing a flight. The edges are added as described below. This results in graphs that together have $O(n)$ nodes and $O(n^2)$ edges. In the process of determining edge weights, we allow a weight parameter w which can be set to increase or decrease the importance of optimising aircraft routes relative to crew pairings.

The rules for the routing graphs are the following:

- All flights starting in a base, receive an incoming edge from the source node of that base
- All flights arriving at a base, receive an outgoing edge to the sink node of the corresponding base
- An edge (f_1, f_2) is constructed between two flight nodes f_1 and f_2, if the flight corresponding to node f_2 can legally be performed after flight f_1 by the same aircraft.
- To allow aircraft routes to end in an airport that is not a base, all nodes that represent a flight that does not end in a base receive an outgoing edge to the first base in the list of bases. An example is the edge from F5 to AMS sink in Fig. 4.

In the graphs, a route is defined as a path from a source to a sink node. Contrary to pairings, aircraft do not have to start and end at the same airport. The cumulative weights of the edges in the path give the cost of the route.

Example 3. One possible route that can result from Fig. 4 is represented by the following path:

$$\text{AMS source} \longrightarrow F0 \longrightarrow F2 \longrightarrow F1 \longrightarrow F3 \longrightarrow \text{EIN sink}$$

and will thus fly the following route:

$$\text{AMS} \longrightarrow \text{DLM} \longrightarrow \text{AMS} \longrightarrow \text{ALC} \longrightarrow \text{EIN}$$

Further details about the routing graphs construction are provided by Appendix B.

3.1.4. *Short connections constraints*

Fourth, to ensure that when crew change aircraft, enough extra time is available, a list of crew–aircraft short connections is created. This list can subsequently be used to enforce that if one of the flight combinations in the list is operated in the same pairing, it should also be operated in the same aircraft route. The list of short connections contains all combinations of two flights f_1 and f_2 for which the following holds:

- f_1 and f_2 are assigned to the same aircraft type
- The arrival airport of f_1 and the departure airport of f_2 are the same
- The time between the arrival of f_1 and the departure of f_2 is larger than the minimum required turn around time, but smaller than the minimum required turn around time + 15 min.

3.1.5. *Incorporating flight retiming*

From previous studies on retiming and airline scheduling in hub-and-spoke networks, it is known that integrating schedule design with other scheduling problems can also have potential to improve crew pairings and aircraft routes (Pothos et al., 1994; Cacchiani and Salazar-González, 2020; Mercier and Soumis, 2007). By allowing the model to move flights in the schedule, a larger number of possible crew and aircraft connections can be formed, thus possibly resulting in more efficient pairings.

In the planning process, the airline acquires departure and arrival slots at the airport it wants to service. These slots indicate that the airline has the right to depart or arrive the airport within the time bracket corresponding to the slot. Per airport there are only a fixed number of slots available per time of day. For example, in the summer of 2019 at Amsterdam Airport Schiphol, the slot between 08:00 and 08:15 local time, has a capacity of 12 arrivals and 25 departures (Airport Coordination Netherlands, 2019). These slots are allocated twice a year; once for the summer season, and once for the winter season. Ideally the departure and arrival times stay within the acquired time-brackets after they have been retimed, as rescheduling flights outside of their slots may not be possible, or at least incur complications. Unfortunately, because the slots and their regulations differ greatly between airports

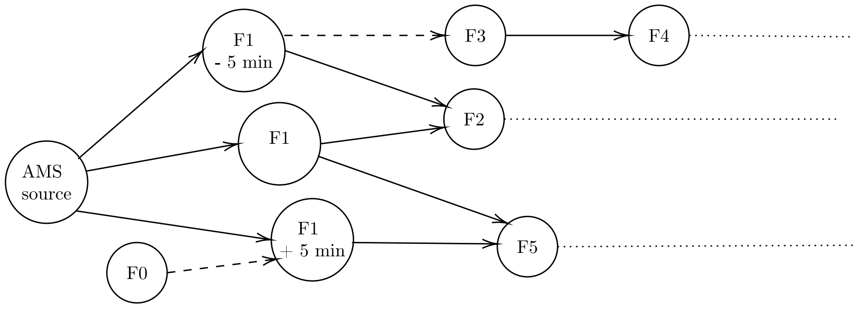


Fig. 5. Flight graph showing the potential new connections, indicated by dashed edges, when allowing flight F1 to retime with ±5 min.

it is at this moment not possible to ensure that flights are always retimed within their slots. However, by allowing flights to move 5 min in the schedule, there is a relatively small probability that the flight's departure or arrival time will be outside the slot's time-bracket (Mercier and Soumis, 2007).

Example 4. An example of the potential of retiming a flight by ±5 min is shown in the flight graph in Fig. 5. Flight 1 is allowed to be retimed with 5 min, therefore creating the additional possibilities to connect to flight 3, or flight 0.

To allow for such five minute changes in timing, flight copies are created. These 'new' flight variables share the same properties as their original flight, but have a departure and arrival time that is shifted by five minutes. The following sections will describe how these new variables can be used in a branch-and-price approach to solve an integrated scheduling model that.

3.2. Mathematical model

Given the data structures described above, we formulate the following integer linear program, based on a set partition formulation in line with the literature. The objective function aims to maximise efficiency by minimising the crew's available duty time that is not spent on flying. Dual variables of constraints are denoted in blue, and explained below.

$$\min \sum_{p\in P} c_p x_p + \sum_{r\in R} c_r x_r \tag{3}$$

$$\text{s.t.} \sum_{t\in D_f} \sum_{p\in P} b^p_{f_t} x_p = 1 \qquad \forall f \in F \qquad \alpha_f \tag{4}$$

$$\sum_{t\in D_f} \sum_{r\in R} b^r_{f_t} x_r = 1 \qquad \forall f \in F \qquad \beta_f \tag{5}$$

$$\sum_{p\in P} b^p_{i_v j_w} x_p - \sum_{r\in R} b^r_{i_v j_w} x_r \le 0 \qquad \forall (i,j) \in C, \quad (v,w) \in C_{i,j} \qquad \gamma_{i_v j_w} \tag{6}$$

$$\sum_{r\in R} b^a_r b^{k^-}_r x_r \le maxAC^k_a \qquad \forall k \in K, a \in A \qquad \delta_{a,k} \tag{7}$$

$$\sum_{p\in P} b^p_{f_t} x_p - \sum_{r\in R} b^r_{f_t} x_r = 0 \qquad \forall f \in F, t \in D_f \qquad \epsilon_{f,t} \tag{8}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{9}$$

$$x_r \in \{0,1\} \qquad \forall r \in R \tag{10}$$

with the parameters and variables as in Table 6.

Objective (3) expresses the minimisation of the combined costs of the chosen pairings and routes. The managerial concern from our partner airline defines this cost as the unused hours within and between duties. Constraint (4) and (5) ensure that all flights are covered by one pairing and one route respectively. Constraint (6) ensures that the flight combinations present in the set of short-connections can only be executed on the same pairing if they are also executed by the same aircraft route. Constraint (7) ensures that there are no more aircraft used than there are available of each type per base. Constraint (8) ensures that the same departure time for each flight was chosen in both the pairing and routing problem. Constraint (9) and (10) are integrality constraints.

3.3. Solving

Given the complexity of the problem addressed, the above model is solved using a branch-and-price approach, a combination of column generation and branch-and-bound. Unsurprisingly, we found that a MILP solver could not scale adequately to the model when presented to the solver as a monolith. Branch-and-price allows us to incrementally solve the model, taking advantage that a pricing sub-problem can be solved relatively easily, as we next explain. Fig. 6 gives an overview of the solving process.

First, beginning with the input flight schedule, the flight copies are created (Section 3.1.5). These flight copies are from there on used to construct the duties, pairing graph, routing graphs and short connections (Sections 3.1.1–3.1.4). For each original flight a 'fake' flight variable is created to ensure that the restricted master problem (RMP, described next) is always feasible. At each iteration, a set of sub-problems are formulated by updating the pairing graph and routing graphs (described below), and solving the resulting resource-constrained shortest path problems on the graphs.

In more detail, for the restricted master problem, the integrality constraints (9) and (10) are relaxed. The pricing problem for this model is a set of shortest path problem in updated pairing and routing graphs: one shortest path problem in the pairing graph, and one shortest path problem for every routing graph. We explain first for the pairing graph; the routing graph is similar.

The pairing graph is updated by taking a copy of the original pairing graph and using the dual variables to update the edge weights:

- The values of dual variables α_f, corresponding to primal constraints (5), are subtracted from the weights of the edges incoming to duty nodes that contain a flight copy of flight f.
- The values of dual variables γ_{i_v,j_w}, corresponding to primal constraints (7), are subtracted from the weights of the edges incoming to duty nodes that contain both flight i with departure time v and flight j with departure time w.
- The values of dual variables $\epsilon_{f,t}$, corresponding to primal constraints (9), are subtracted from the weights of the edges incoming to duty nodes that contain flight f with departure time t.

Subsequently, a negative reduced cost column can be found by solving the shortest path problem on the updated graph (compare Yan et al. (2002)). The source and target nodes for the shortest path problem are represented by source and sink nodes of the same base. The solving methodology for the shortest path problem depends on the number of

Table 6
Parameters and variables of the MILP model. Dual variables are denoted by blue colour.

Decision variables	
x_p	Binary decision variable that indicates whether pairing p is chosen
x_r	Binary decision variable that indicates whether route r is chosen
Sets	
P	Set of pairings
R	Set of routes
K	Set of bases
A	Set of aircraft types
F	Set of flights in the schedule
D_f	Set of departure times for flight f
C	Set of short connections
$C_{i,j}$	Set of departure time combinations (v, w) where v belongs to flight $i \in C$, and w belongs to $j \in C$
Constants	
c_p	Cost of choosing pairing p
c_r	Cost of choosing route r
$MaxAC_a^k$	Number of available aircraft of type a at base k
$b^p_{f_t}$	Binary constant indicating whether flight f with departure time t is in pairing p
$b^r_{f_t}$	Binary constant indicating whether flight f with departure time t is in route r
b^a_r	Binary constant indicating whether route r is operated by aircraft type a
b^{k-}_r	Binary constant indicating whether route r starts in base a
$b^p_{i,j}$	Binary constant indicating whether flight i and j are in pairing p
$b^r_{i,j}$	Binary constant indicating whether flight i and j are in route r
$b^p_{i_v j_w}$	Binary constant indicating whether flight i with departure time v and j with departure time w are in pairing p
$b^r_{i_v j_w}$	Binary constant indicating whether flight i with departure time v and j with departure time w are in route r
Dual Variables	
α_f	Dual variables for constraints (4)
β_f	Dual variables for constraints (5)
$\gamma_{i_v,j_w} l$	Dual variables for constraints (6)
δ_a, k	Dual variables for constraints (7)
$\epsilon_{f,t}$	Dual variables for constraints (8)

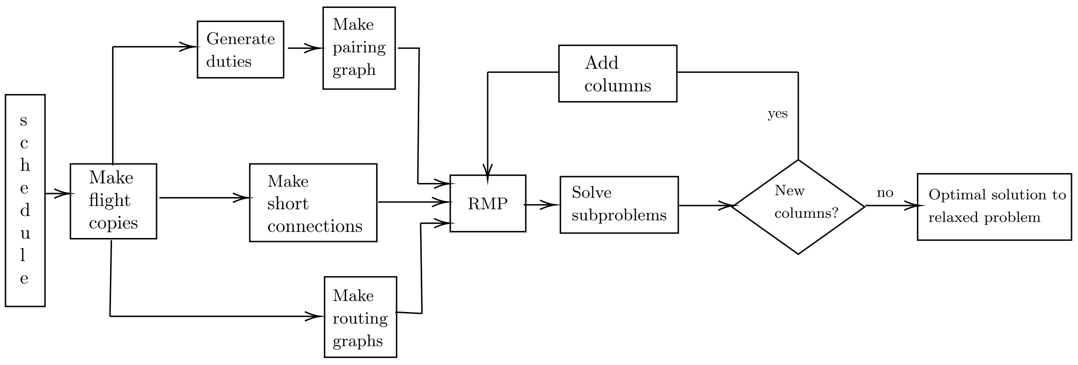


Fig. 6. Overview of the column generation solving process. If the optimal solution to the RMP is not integral, then branch and bound search (with interleaved column generation) derives the optimal integer solution from it.

days that a given flight schedule spans. If this number is smaller or equal to the maximum allowed pairing length, a ‘regular’ shortest path problem will be solved by using the Bellman–Ford algorithm (Bellman, 1958; Ford, 1956). If the schedule spans more days than the maximum allowed pairing length, the problem will become a Shortest Path Problem with Resource Constraints (SPPRC). For this purpose, the resource cost edge attribute has been defined during the pairing graph generation (Section 3.1.2). The sum of these resource costs in a path indicate how much time has passed since the beginning of the pairing. To find a shortest path that respects the maximum pairing duration, the maximum resource that can be consumed in a shortest path is set equal to the maximum pairing length. As the SPPRC is in itself NP-hard, it is first attempted to find negative reduced columns by using a heuristic approach, using a greedy elimination heuristic that eliminates edges that contribute to infeasible resource costs (Torres Sanchez, 2019).

For the routing graphs, the edge weight updates from the dual variables are:

- The values of dual variables β_f, corresponding to primal constraints (6), are subtracted from the weights of the edges incoming to the flight node representing a copy of flight f.
- The values of dual variables γ_{i_v,j_w}, corresponding to primal constraints (7), are added to the weights of the edges between flight nodes representing flight i with departure time v and flight j with departure time w.
- The values of dual variables $\delta_{a,k}$, corresponding to primal constraints (8), are subtracted from the weight of outgoing edges from the base source node of base k in the routing graph of aircraft type a.
- The value of dual variables $\epsilon_{f,t}$, corresponding to primal constraints (9), are added to the weights of the edges incoming to flight nodes that represent flight f with departure time t.

When no more negative reduced cost columns can be found by using the heuristic, an exact label-setting algorithm will determine whether there are no paths with negative reduced cost left (Boland et al., 2006). This ensures that the overall approach still results in an optimal solution.

After the pairing graph and all routing graphs have been updated using the dual variables, negative reduced cost columns are found for each graph. For the pairing graph, a (resource-constrained) shortest path problem is solved as above. This results in a shortest path for each base, of which any with negative reduced cost are chosen to be added to the restricted master problem. For the routing graphs, the shortest path problem is solved for each possible base combination in each graph. This results in $|A| * |K| * |K - 1|$ shortest paths, of which again any with negative reduced cost will be added to the restricted master problem.

As depicted in Fig. 6, the solving process described thus far obtains the optimal solution to the RMP. It is checked whether this relaxed optimal solution consists of all integer values. If this is the case, the optimal solution to the relaxed problem is also the optimal solution to the integer problem. If this is not the case, the problem is embedded in a branch-and- price construction to find the optimal integer solution (Desrosiers and Lübbecke, 2011).

The branch-and-price search uses the optimal solution of the restricted master problem as the root of its search tree. Each node corresponds to a linear programming problem, namely the RMP with additional constraints. In standard branch-and-bound fashion, from the root of the tree, two child nodes are created by branching, here on a master problem variable chosen by the simple lexicographical branching heuristic (i.e., branch on the first non-integer variable). This variable is set to 0 on the first branch, and 1 on the second branch. For both child nodes, the objective value (3) is calculated, and branching takes place from the node with the lowest objective value on the leftover non-integer variable whose value is closest to 0. The relaxed LP is solved and its corresponding pricing problems computed as before. This process is repeated until an integer solution has been found. The process of branching, fathoming and calculating objective values repeats until there are no nodes left that have a lower objective value than the current best integer solution. Some branches will thus not be explored, as their relaxed objective value is already higher than the current best integer value; this helps speed up the process of obtaining an integer solution.

During the process of calculating a bound via the pricing subproblems, it is possible that new negative reduced cost columns are found; these are added in the same way as described for the RMP, until no more columns can improve the solution. A global column pool can be used to filter re-generated columns from branch-and-bound nodes, although for our problem instances we found the default behaviour of the LP solver was adequate (Desrosiers and Lübbecke, 2011). Note we do not explicitly discard columns, nor use further cutting planes. In the end, when all branches have been explored or fathomed, the integer solution with the lowest objective value is returned as the optimal solution to the integer problem. The final integer solution will, next to the optimal crew pairings and routes, give what flight copies should be executed, and thus how the flights should be retimed.

4. Results and discussion

This section studies the integrated model proposed in terms of (1) solution quality, (2) practical value for airlines, and (3) scalability. To this end we undertake a set of computational experiments on real-world data, also comparing the presented model with direct solving and with ablated models.

4.1. Data

The input data for all models include a flight schedule consisting of legs that need to be supplied with crew and an aircraft. In order to later be able to compare the outcome of the currently presented crew pairing models and the partner airline's pairings, it is important to select a range of flights that enables a comparison that is as fair as possible. To this end, the selection of flights took place by looking at the starting date of the airline's pairings. For example, for a 'one-day' dataset, all flights that were performed in the airline's pairings starting from that day were chosen. This avoids that only half of the flights of a pairing made by the partner airline are in the current dataset, therefore enabling a more fair comparison of results. The set of all datasets can be seen in Table 7. The timespan indicates how many pairing starting days were used to get flights from, while the date range indicates between which dates the chosen flights fall. The last column indicates whether a Shortest Path Problem with Resource Constraints is solved.

Table 7
Overview of all data sets and their characteristics.

Dataset number	Dataset name	# Flights	Timespan	Date range	SPPRC
1	1_day	142	1 day	1–2 March 2019	×
2	2_day	282	2 days	1–5 March 2019	✓
3	3_day	438	3 days	1–5 March 2019	✓
4	4_day	567	4 days	1–6 March 2019	✓
5	5_day	678	5 days	1–7 March 2019	✓
6	6_day	789	6 days	1–7 March 2019	✓
7	7_day	926	7 days	1–9 March 2019	✓

4.2. Models

We compare the model and solving approach of Section 3 with four simpler models. Recall that we wish to compare the integrated versus direct models. Table 8 summarises whether each model does, in addition to solving the crew pairing problem, also integrate the aircraft routing problem, and whether it allows flight retiming. The last column indicates whether the model is a monolithic MILP, or whether we employ branch-and-price as described previously. In particular, CP is the vanilla crew pairing model,[2] and CG denotes the use of column generation.

Unless mentioned otherwise, all models use a set of standard parameters for the presented experiments below. Table 9 gives an overview of these parameters.

The models were implemented in Python 3.6, using packages CSPY and Pylgrim to implement the solving strategy of the pricing problems (Weyens and van Vugt, 2019; Torres Sanchez, 2019); CBC 2.10 was used as the MILP/LP solver via the package PuLP. All experiments were run on a Linux machine with an 8-core 2.40 GHz Intel Xeon Gold 6148 processor and 32 GB of RAM. A timeout of 24 h was used.

4.3. Results

We first provide empirical results on the pairing and routing graph generation, then analyse the size of the problem encoding (Section 4.3.1), before reporting the results of the three main experiments. First, the solution quality and how it develops over dataset size and across different models (Section 4.3.2). Second, a comparison with the partner airline's current methodology (Section 4.3.3). Third, the runtime performance of all models (Section 4.3.4). A sensitivity analysis concludes the empirical section (Section 4.4).

4.3.1. Preliminary experiments

Pairing and routing graphs. To give insight into the size of the shortest path problems that are solved for each experiment, Table 10 reports an overview of the pairing and routing graph sizes in terms of the number of edges and the number of nodes for each data set.

[2] For disambiguation, note that 'CP' refers to crew pairing, and not to the optimisation methodology 'constraint programming'. In this article we align with the use of linear mathematical models and MILP solving techniques; we do not explore the use of modelling or solving with constraint programming.

Table 8
Overview of all implemented models and their characteristics.

Model number	Model name	Crew pairings	Aircraft routes	Retiming	Branch-and-price
1	Vanilla CP	✓	×	×	×
2	CP_CG	✓	×	×	✓
3	CP_Integrated	✓	✓	×	×
4	CP_Integrated_CG	✓	✓	×	✓
5	CP_Retiming_CG	✓	✓	✓	✓

Table 9
Parameters and their standard values used for the presented experiments.

Parameter	Explanation	Value
`Check-in time`	Time needed for check-in activity	1 h
`Check-out time`	Time needed for check-out activity	30 min
`Taxi check-in time`	Time needed for check-in before a taxi activity	15 min
`Default turn-around time`	Turn-around time used when no time is found for the airport and aircraft combination	40 min
`Maximum turn-around time`	Maximum time between two sequential flights in one duty	3 h
`Extra time aircraft change`	The extra time needed when crew needs to change aircraft	15 min
`Maximum pairing length`	The maximum time a pairing covers	4 days
`Maximum layover time`	Maximum time between two duties in a pairing	3 days
`w`	Importance of optimising aircraft routes relative to pairings. Same importance if `w` = 1	0.1

Table 10
Specifications of the pairing and routing graphs created for each dataset. In the case of routing graphs, all numbers are reported as a sum over all routing graphs.

Dataset	Pairing graph			Routing graphs			Pairing graph (retimed)			Routing graphs (retimed)		
	# nodes	# edges	mean degree	# nodes	# edges	mean degree	# nodes	# edges	mean degree	# nodes	# edges	mean degree
1_day	130	276	2.12	154	599	3.89	3300	9015	2.73	438	3963	9.05
2_day	386	1375	3.56	294	2713	9.23	[b]	[b]	×	[b]	[b]	×
3_day	832	4638	5.57	[a]	[a]	×	[b]	[b]	×	[b]	[b]	×
4_day	1329	10 535	7.93	[a]	[a]	×	[b]	[b]	×	[b]	[b]	×
5_day	1703	15 721	9.23	[a]	[a]	×	[b]	[b]	×	[b]	[b]	×
6_day	2075	21 429	10.33	[a]	[a]	×	[b]	[b]	×	[b]	[b]	×
7_day	2488	28 332	11.39	[a]	[a]	×	[b]	[b]	×	[b]	[b]	×

[a] Indicates that not enough memory was available to complete the runs.
[b] indicates that the runs took longer than 24 h to complete.

Table 11
Average number of decision variables per problem ($n = 24$).

Data set	CP	CP_CG	Difference	`CPIntegrated`	`CPIntegrated_CG`	Difference	`CPIntegratedSchedule_CG`
1_day	298	221	*−25.8%*	1861	704	*−37.8%*	1927
2_day	1607	506	*−68.5%*	330 586	4308	*−98.7%*	[b]
3_day	11 298	921	*−91.8%*	[a]	[b]	×	[b]
4_day	63 533	1397	*−97.8%*	[a]	[b]	×	[b]
5_day	91 789	2045	*−97.7%*	[a]	[b]	×	[b]
6_day	122 099	2697	*−97.8%*	[a]	[b]	×	[b]
7_day	150 046	3244	*−97.8%*	[a]	[b]	×	[b]

[a] Indicates that the runs did not finish because of not enough memory was available.
[b] indicates that the run did not finish within 24 h.

It can be observed that for each graph the number of nodes and edges increases as the dataset covers more days and flights. Additionally, the retimed pairing graph contains more nodes and edges than the regular pairing graph as a result of the larger set of possible duties due to retiming. It is interesting to see that next to the number of nodes and edges, the mean degree in the retimed pairing graph is also larger than the regular pairing graph for the same data set. This indicates that the retiming results in more possible paths. The same observations can be made for the retimed routing graphs.

Decision variables. To investigate the size of the MILP problems it is interesting to examine the number of decision variables per model and dataset. This is especially relevant to investigate the difference in problem size between the generate-and-test and branch-and-price approaches, as the aim of column generation is to solve smaller problems and add decision variables only as needed. Table 11 reports the average number of decision variables generated during the entire model per model and dataset, over 24 runs per model–dataset combination.

As can be observed from Table 11, the column generation approach uses many fewer decision variables than the vanilla `CP` approach. Further, the integrated models use more decision variables, as there is one per crew pairing and one per aircraft route. With regards to the difference between the generate-and-test and branch-and-price approaches, it can be seen that the branch and price approaches use a significantly smaller number of decision variables. For example, `CPIntegrated_CG` only uses 1.3% of the decision variables compared to `CPIntegrated`. Generally, the percentage of decision variables that do not have to be generated increase with the size of the dataset. In addition, it can be seen that there is a larger percentile difference for the integrated models (see Fig. 7).

4.3.2. Experiment 1: Solution quality

Purpose and metrics. A question of central interest is the impact of the different models on the total objective value. Recall that objective function (3) is a minimisation. Standard metrics for this purpose will be the objective value (3) of the integer and linear problems, and the resulting integrality gap. Additionally, to be able to compare the different model outcomes, it is important to look at the objective values themselves. To compare the branch-and-price against the generate-and-test approach for efficiency, it is interesting to compare the number of

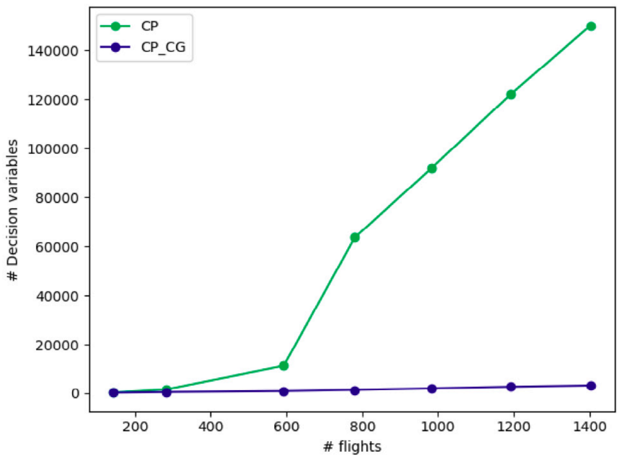


Fig. 7. Number of decision variables for models CP and CP_CG over problem size.

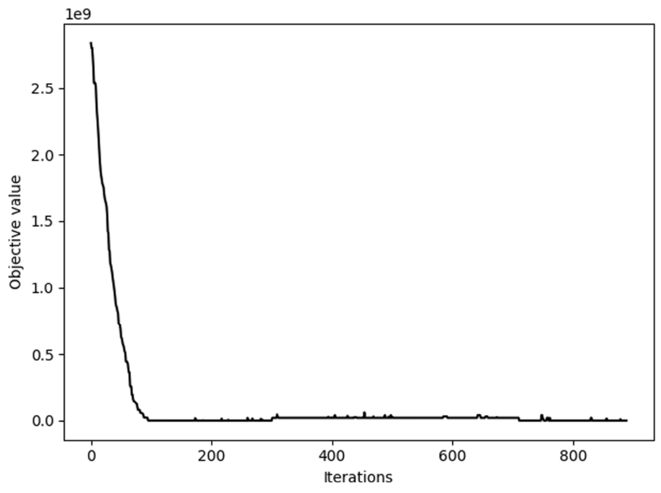


Fig. 8. Objective value of model CPIntegratedSchedule_CG on the 1_day dataset per iteration.

possible pairings and routes, with the number of generated and used pairings and routes.

Hypothesis. The integrated models CPIntegrated_CG and CPIntegrated will have a worse objective value than the retiming-integrated model CPIntegratedSchedule_CG.

Results. The solution quality of the presented models on the different datasets is presented in Table 12. It can be seen that the objective value of models CP and CP_CG, and models CPIntegrated and CPIntegrated_CG are equal to each other. This is an indicator of the correctness of the column generation technique in the branch-and-price approaches, as the branch-and-price algorithm should not alter the exact outcome of the problem, but only decrease the number of decision variables. In addition, the CPIntegrated and CPIntegrated_CG models have a higher objective value than the non-integrated models, due to the inclusion of routing costs. As expected, the retiming model CPIntegratedSchedule_CG greatly reduces the objective value compared to the other integrated models. Interesting to note from Table 12 is that the solution for the linear problem is equal to the solution value for the integer problem. With the integrated models, this is not the case, but the integrality gap is still high at a lowest value of 0.99.

Fig. 8 displays the value of the objective function over the column generation iterations throughout the branch-and-price process. It can be seen that the objective value greatly decreases in the first 100 iterations, after which the solution to the linear relaxation is found around iteration 175. From the 200th iteration onwards, small spikes in objective value can be observed. These spikes are caused by the branch-and-bound process, where a variable's value is fixed while keeping the existing decision variables. These peaks often get lower because additional columns are added, or the solution is fathomed.

4.3.3. Experiment 2: Practical value

Purpose and metrics. Since our central interest is to achieve an improvement in the crew productivity, it is important to investigate whether the models return more productive solutions than the partner airline designed for the given flight schedules. To achieve this, the results of all models on all data sets will be compared to the pairings used by the partner airline for the exact set of flights. One important metric is the average block hours/duty hours per duty over all pairings as this gives a measure for the crew productivity. Additionally, the average number of duties needed to cover all flights is important because it gives a lower bound on the number of crew needed each day to operate a flight schedule. It will also be interesting to see whether more layovers will be implemented, as this might increase the crew productivity, but is currently not a favourable option for the partner airline because of the costs associated with them.

Hypotheses.

- It is expected that all models will return an improvement in crew productivity as measured by the average block hours per duty hours ratio, compared to the partner airline's results.
- It is expected that the integrated models will return a larger improvement compared to the non-integrated models.

Results. Fig. 9 shows a visualisation of the pairings resulting from running model CP on the 2_day dataset. Each horizontal value represents one pairing, where the green colour indicates a flight, blue indicates a check-in or check-out activity, pink is for taxi activities and grey represents a hotel stay. It can be seen that mostly pairings consisting of only one duty are formed; only two pairings with layovers are necessary.

Fig. 10 shows how the results from the different models compare to the partner airline's current pairings, in terms of the number of (long) duties, positioning activities, layovers, and the block hours over duty period (BH/DP). All these statistics have been multiplied by a factor to cover their real values. This has been done to protect the sensitivity of the results, while still being able to show the relationship between them. As can be seen from Figs. 10(a), 10(b) and 10(c), generally all models over the 1, 2 and 7_day datasets result in a higher number of duties and fewer positioning. Additionally, the results for datasets 1_day and 2_day also contain fewer layovers. This is a positive outcome, as the number of duties gives a lower bound on the number of crew members that are needed to cover a schedule, and positioning and layover activities cost time and money while the crew are at that moment not operating flights.

The hashed statistics corresponding to these observations can be found in Table 13. It can be seen that all models have a higher block hour per duty hour ratio. It stands out that significantly fewer (4%–10%) pairings are used by the currently presented models. This percentage decreases with problem size. In addition, the resulting crew efficiency measured as the ratio between flight hours and duty hours per duty is 0.6% to 1.5% higher compared to the partner airline's pairings. These results decrease with problem size, but generally increase with integration level, as expected.

4.3.4. Experiment 3: Runtime

Purpose and metrics. Solving time is relevant when deciding whether to use the approach proposed in Section 3. Therefore, the runtime of all models will be examined for all datasets by recording the total runtime and the time needed to find a solution to the linear relaxation.

Table 12
Objective values for the integer and linear problem associated to the different models and data sets, with the average number of column generation iterations (n = 24).

Data set	CP	CP_CG				CPIntegrated	CPIntegrated_CG				CPIntegratedSchedule_CG			
	IP	LP	IP	Gap	Iterations	IP	LP	IP	Gap	Iterations	LP	IP	Gap	Iterations
1_day	2 141 700	2 141 700	2 141 700	1.0	42	2 469 180	2 469 180	2 469 180	1.0	93	447 779.99	447 780.00	0.99	401
2_day	4 400 700	4 400 700	4 400 700	1.0	101	5 501 370	5 501 369.99	5 501 370	0.99	926	[b]	[b]	×	×
3_day	6 763 200	6 763 200	6 763 200	1.0	203	[a]	[b]	[b]	×	×	[b]	[b]	×	×
4_day	8 883 000	8 883 000	8 883 000	1.0	369	[a]	[b]	[b]	×	×	[b]	[b]	×	×
5_day	10 729 500	10 729 500	10 729 500	1.0	647.54	[a]	[b]	[b]	×	×	[b]	[b]	×	×
6_day	12 535 200	12 535 200	12 535 200	1.0	898.875	[a]	[b]	[b]	×	×	[b]	[b]	×	×
7_day	14 759 700	14 759 700	14 759 700	1.0	1114.125	[a]	[b]	[b]	×	×	[b]	[b]	×	×

[a] Indicates runs were not completed because not enough memory was available.

[b] Indicates that the run did not finish within 24 h.

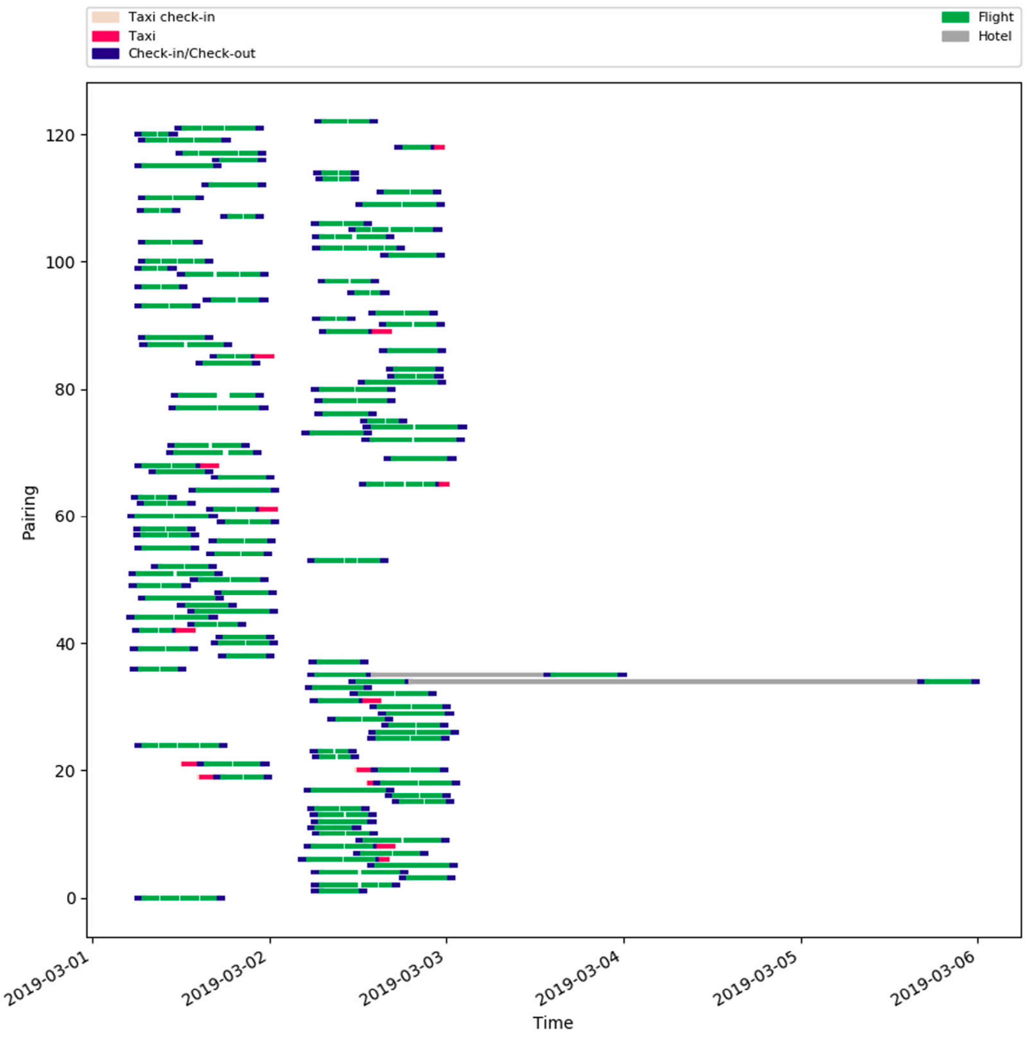


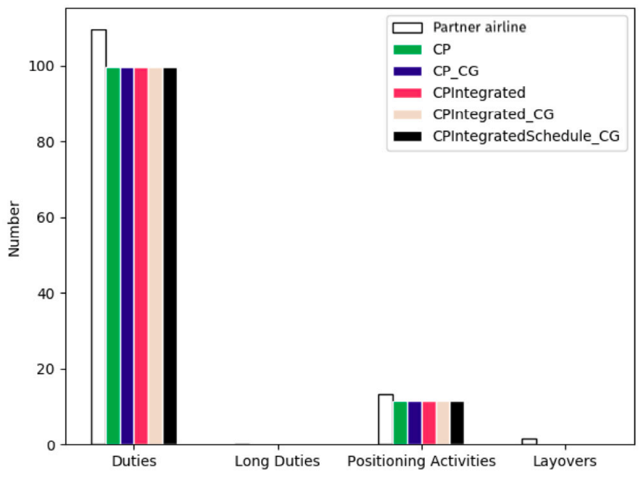
Fig. 9. Visualisation of the pairings resulting from model CP ran on the 2_day data set, where each horizontal line at the y-axis represents a pairing. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Hypotheses.

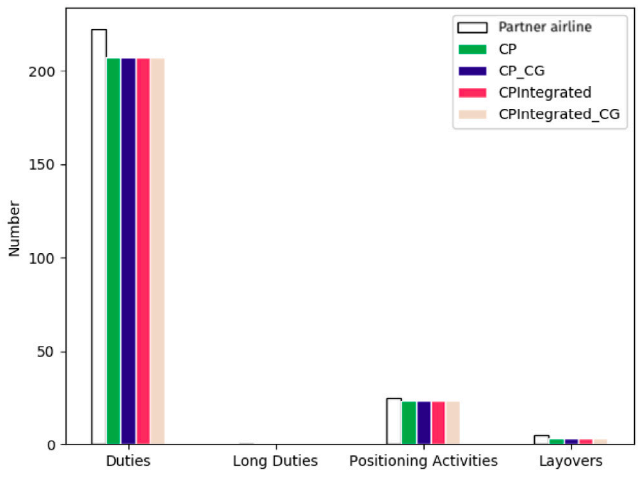
- Although it is expected that the more integrated models will provide better solutions, it is also expected that the more a model is integrated, the longer its runtime will be for the same data set.
- The branch-and-price methodology will be slower than the generate-and-test approach for small data sets due to the large number of steps, but faster for larger data sets.

Results. The third experiment studies runtime results for all experiments. The runtimes can be observed from Table 14, and are visualised in Fig. 11(a). To better show the runtimes that are close together, Fig. 11(b) shows the normalised runtime.

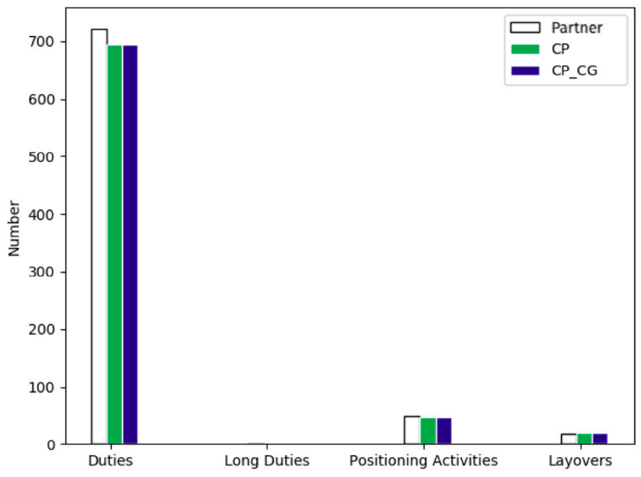
As expected given the more challenging problem modelled, the more integrated a model is, the longer its runtime will be for the



(a) Number of duties, long duties (>14 hours), duties with positioning, and pairings with layovers on a 1_day dataset



(b) Number of duties, long duties (>14 hours), duties with positioning, and pairings with layovers on a 2_day dataset



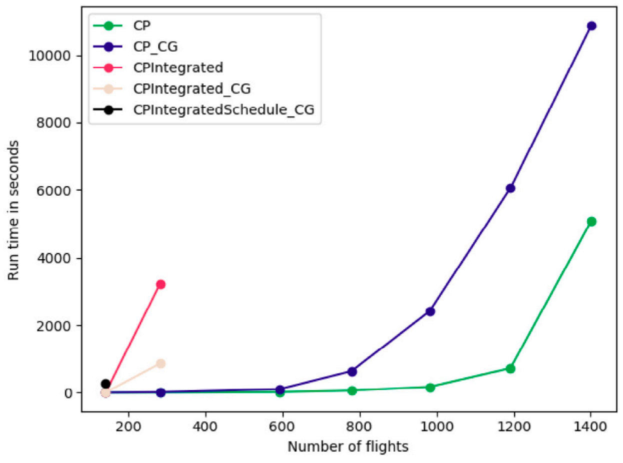
(c) Number of duties, long duties (>14 hours), duties with positioning, and pairings with layovers on a 7_day dataset

Fig. 10. Barchart comparing the results between the partner airline's pairings and the different models' results.

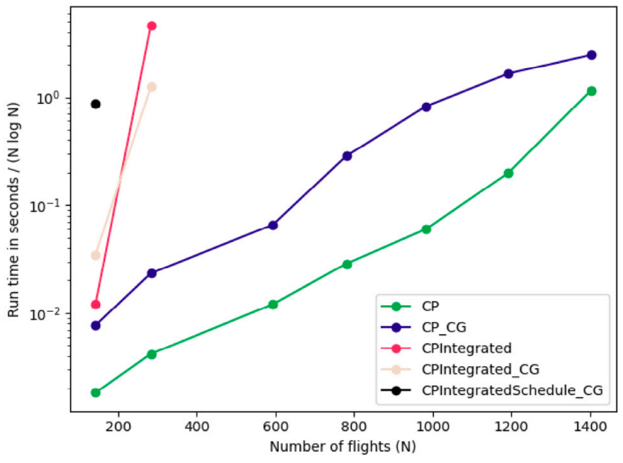
Table 13
Hashed practical results of the models over the 1, 2 and 7_day datasets. Column ‘Current’ reports the actual values of the partner airline's current solution.

Dataset	Metric	Current	CP	CP_CG	CPIntegrated	CPIntegrated_CG	CPIntegratedSchedule_CG
1_day	#duties	109.56	99.6	99.6	99.6	99.6	99.6
	#positioning	13.28	11.62	11.62	11.62	11.62	11.62
	BH/DP	1.133	1.150	1.150	1.148	1.150	1.148
2_day	#duties	222.44	207.5	207.5	207.5	207.5	[b]
	#positioning	24.9	23.24	23.24	23.24	23.24	[b]
	BH/DP	1.160	1.170	1.172	1.172	1.177	[b]
7_day	#duties	722.1	693.88	693.88	[a]	[b]	[b]
	#positioning	48.14	46.48	46.48	[a]	[b]	[b]
	BH/DP	1.172	1.175	1.179	[a]	[b]	[b]

[a] Indicates that the run was not completed due to too little memory.
[b] Indicates that the run did not complete within 24 h.



(a) Runtime (seconds)



(b) Normalized runtime

Fig. 11. Average (normalised) runtime over problem size for all models (n = 24).

same dataset. This can clearly be seen in Fig. 11(a), where there is a large difference between the runtimes of the CP and CPIntegrated models, as well as between model CP_CG and CPIntegrated_CG. For the non-integrated models, the runtime of the generate-and-test approach is always smaller than the runtime of the branch-and-price approach. This is most likely due to the relatively small number of decision variables. When looking at the CPIntegrated and CPIntegrated_CG models, it stands out that the generate-and-test method is faster for the 1_day dataset, while the branch-and-price approach is faster on the larger 2_day dataset. The branch-and-price approach therefore seems to be promising for very large problem sizes; this is conform expectation.

Additionally, it is interesting to look at the mean and maximal runtimes for all models, as presented in Table 14 and the boxplots in Fig. 12. Generally, there is a relatively small difference between the worst-case runtime and the average runtime. This difference is larger for models using the branch-and-price approach. This is as expected, because of the different order in which the columns can be added due to paths having the same weight, and the use of a heuristic for the SPPRC. The spread of the runtimes is also dependent on the problem size, where the larger the problem, the larger the difference between the mean and maximal runtimes becomes. An example hereof is given by the boxplots in Fig. 12.

As Section 5 will note, we reserve for the future means to increase the scaling ability of the most integrated models. Possibilities include improving the branching rule of Section 3.3, reimplementing in a more performant language such as C++, using learning from a distribution of problem instances to guide the MILP solving, and moving from exact solving as in this article to the use of meta-heuristics.

4.4. Sensitivity analysis

Finally, this section investigates the influence of the maximum pairing length parameter on the solution quality and runtime results. This parameter is studied due to its importance in practice. To this end, 12 additional runs per dataset-model combination have been performed, with the maximum pairing length set to 6 instead of 4 days. All parameters, except the maximum pairing length, were set to the values presented in Section 4.2. Setting the maximum pairing length to a larger value influences which solving method is required for the shortest path problem. Table 15 shows the updated dataset characteristics with regards to the SPPRC. It can be seen that the pricing problems of the 2, 3, and 4_day datasets can be now be solved by solving a regular shortest path problem, instead of the SPPRC.

As can be seen in Fig. 13, the number of decision variables increases greatly from the 4_day dataset onwards, as that is the first dataset that has a large number of possible pairings that are longer than 4 days. Interestingly, a similar piecewise linear function as for a maximum pairing length of 4 days can be observed. This time only, a strong increase can be noticed for the CP model from the 4_day dataset onwards, as many more combinations are possible with a maximum pairing length of 6 days. These increased number of decision variables also take up a lot of memory; the CP model with maximum pairing length of 6 days did not finish on the 6_day and 7_day datasets, as the program ran out of memory. From Fig. 13 it can also be seen that the

Table 14
Average and maximum runtimes per model and data set in seconds.

Dataset	CP		CP_CG		CP Integrated		CP Integrated _CG		CP Integrated Schedule_CG	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
1_day	0.56	0.86	2.35	3.64	3.66	5.61	10.54	16.0	265.15	267.19
2_day	2.9	4.23	16.19	24.73	3219.23	3330.85	864.13	879.68	[b]	[b]
3_day	19.81	24.56	108.05	112.55	[a]	[a]	[b]	[b]	[b]	[b]
4_day	64.62	65.42	642.67	663.68	[a]	[a]	[b]	[b]	[b]	[b]
5_day	175.52	177.37	2411.87	3139.14	[a]	[a]	[b]	[b]	[b]	[b]
6_day	729.23	735.93	6062.17	8528.28	[a]	[a]	[b]	[b]	[b]	[b]
7_day	5099.19	5603.89	10 883.6	14 122.46	[a]	[a]	[b]	[b]	[b]	[b]

[a] Indicates that the runs did not complete due to too little available memory.
[b] Indicates that the run took longer than 24 h.

Table 15
Datasets and their updated SPPRC attribute when maximum pairing length is set to 6 days.

Dataset number	Dataset name	SPPRC (length = 4 days)	SPPRC (length = 6 days)
1	1_day	×	×
2	2_day	✓	×
3	3_day	✓	×
4	4_day	✓	×
5	5_day	✓	✓
6	6_day	✓	✓
7	7_day	✓	✓

CP_CG model needs to add a small amount of decision variables before it finds an optimal solution, compared to the 4-day maximum pairing length. However, this increase is far less extreme than can be seen for the CP model.

When looking at the solution quality, a larger maximum pairing length shows to have no effect on the objective value. This can be seen in Fig. 14, as the dotted and star marked lines for the 6 day maximum pairing length runs are covered by the lines of the 4 day maximum pairing length runs. This can be explained by the high idle time that comes with hotel layovers. Therefore, longer pairings are not favourable over shorter pairings, unless they cover a flight that could not be covered before.

With regards to the runtime of the models, Fig. 14 shows that the runtime for the CP and CP_CG models is lower when the maximum length is set to 6 days up until the 4_day dataset. For larger datasets, the runtime for the maximum length of 6 days increases above the runtime for the maximum length of 4 days. This can again be attributed to the difference between solving a shortest path problem by using Bellman–Ford or solving a shortest path problem with resource constraints. This difference is smaller for the integrated models (see Fig. 15).

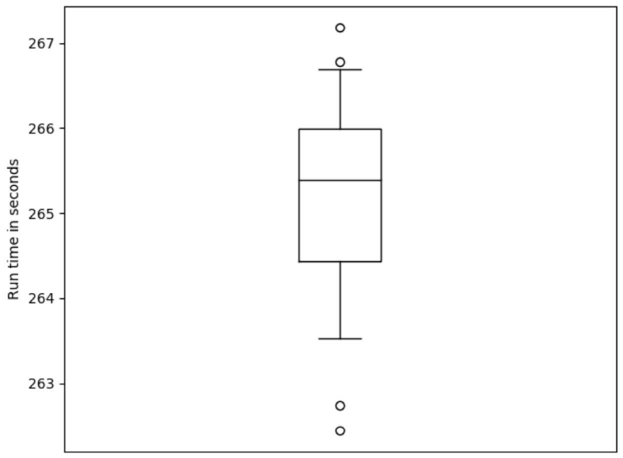
5. Conclusion

This article investigated how the crew pairing problem can be integrated with the aircraft routing problem and schedule retiming as to improve the crew productivity of a low-cost carrier operating a point-to-point network. A case study was provided on the network and data of a European airline which, like many other European airlines, pays flight crew irregardless of the number of hours flown. Because of this pay structure, it is desirable to assign crew to the flight schedule as efficiently as possible. While the (integrated) crew pairing problem has been studied extensively in the past, the literature has mostly focused on US network carriers and minimising pay-and-credit, while European low-cost carriers and optimising crew efficiency are often overlooked.

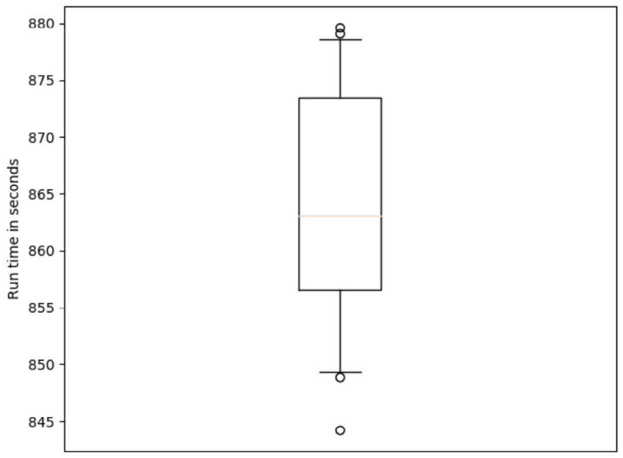
To fill these gaps, and explore the potential of integrated and aircraft routing problem with schedule retiming for the partner airline, we developed an integrated mixed-integer linear programming model. We overcome the scale and computational complexity of solving the model by developing a branch-and-price decomposition approach. Computational results show that the pairings created include fewer positioning and layover activities, and have a higher block hour to duty hour ratio, than the pairings used by the partner airline in practice. Additionally, while the more integrated models have longer runtimes, they are also more able to decrease the objective value; moreover, the runtime even with a prototype implementation is acceptable for the frequency and scale of the problem in the case study. In short, the practical results of up to 10% fewer duties and up to 1.5% higher crew efficiency, show that integrated crew scheduling is also promising for airlines operating a point-to-point network.

Despite these promising outcomes, there are directions to extend the presented model. Firstly, each aircraft type has different possible configurations, of which, for example, the maximum take-off weight is of great importance. Although two aircraft are of the same type, it might be that one has a lower maximum take-off weight and is therefore unable to fly to all of the destinations in the schedule. Secondly, the current research did not take required maintenance activities into account. As aircraft need maintenance in a hangar as often as once a week, including this constraint in the integrated problem will further increase the applicability of the results in real life. Thirdly, the currently presented models can be extended with additional possibilities in the duty generation phase. For example, currently, the models do not produce duties where a positioning activity is located in the middle of a duty, surrounded by flight activities. This is however a possibility that can be explored in the future.

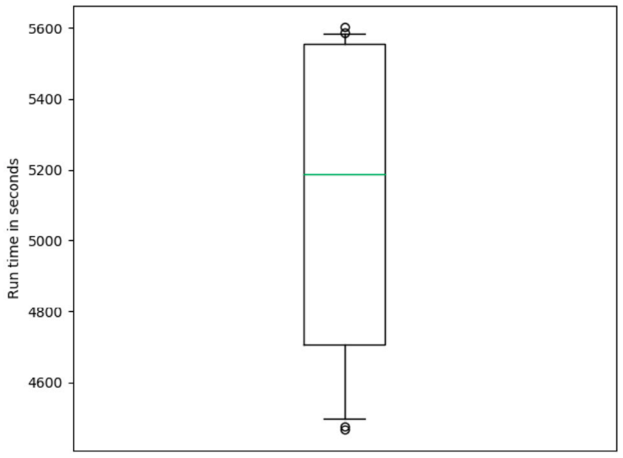
From the algorithmic perspective, there are multiple directions for further improvement. First, we note that the current branch-and-price phase takes a large number of iterations. This can be a motive to look further into promising branching strategies that need fewer iterations to find an integral solution (Vanderbeck, 2000) and to more sophisticated formulations (Lam et al., 2022). Second, it could be beneficial for the retiming model to not retime all flights, but a select number chosen by a heuristic: those which are expected to yield improvements in the pairings that can be made by allowing retiming. Third, to further decrease the general runtime of all models, it is advisable to implement



(a) CPIntegratedSchedule_CG on 1_day data set



(b) CPIntegrated_CG on 2_day data set



(c) CP on 7_day data set

Fig. 12. Non-integrated, integrated, and schedule-integrated model showing the run-time results for their largest data sets ($n = 24$). The whiskers indicate the area between the 5th and 95th percentile.

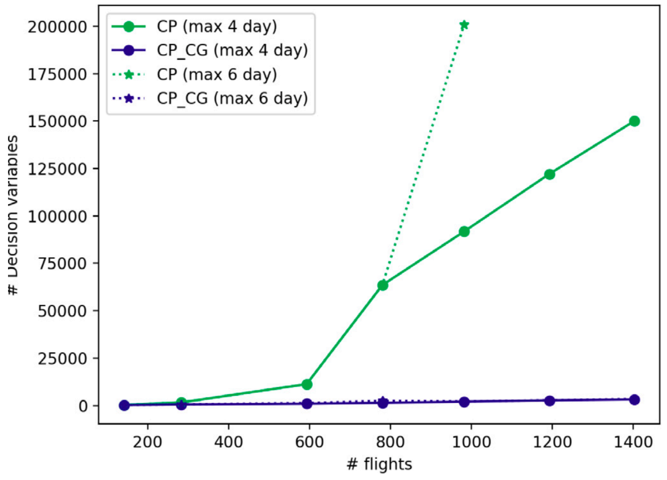


Fig. 13. Average number of decision variables per dataset for maximum pairing lengths of 4 and 6 days.

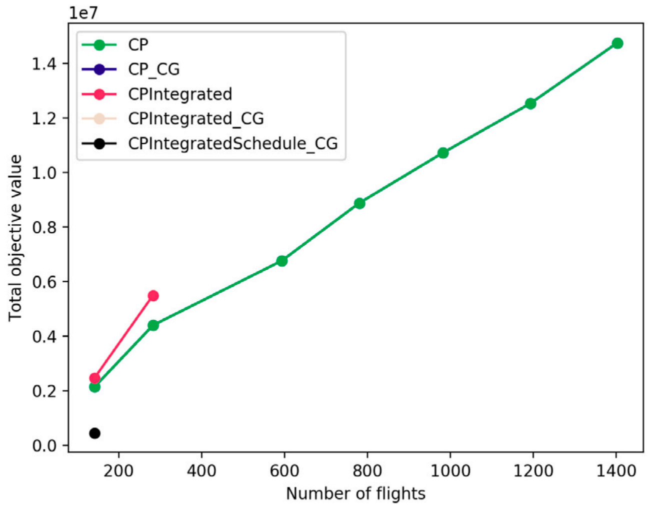
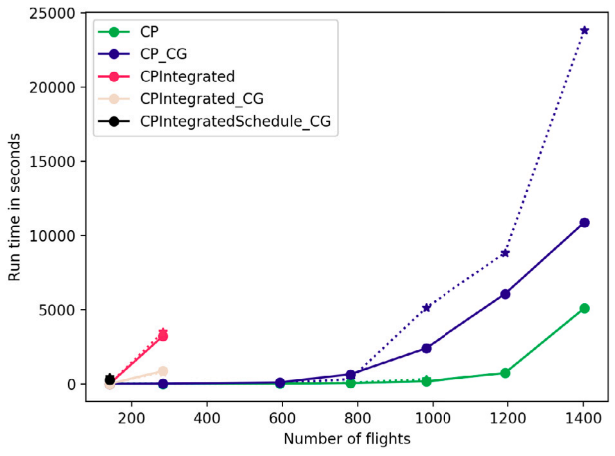
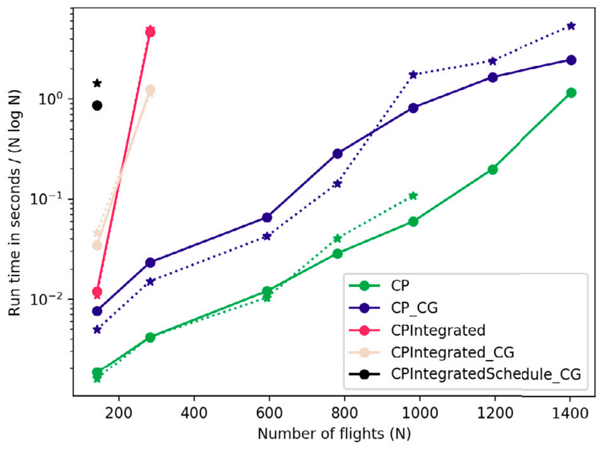


Fig. 14. Objective value per model and dataset with maximum pairing lengths of 4 and 6 days.

these and similar models in a language such as C++ or Rust that is more suitable for computational efficiency and for multiprocessing. This reduces the computational overhead of the implementation, and allows the sub-problems of the integrated models to be solved in parallel, making the entire solving process a lot faster. Further, the LP solver of CBC could be replaced by a state-of-art solver such as that of CPLEX or Gurobi. Lastly, given the potential of machine learning (ML) aided combinatorial optimisation, particularly in the face of stochastic settings (Bengio et al., 2021; Dumouchelle et al., 2022), exploring how ML can accelerate (exact) integrated airline schedule optimisation remains an interesting avenue; one relevant step is that of Quesnel et al. (2022).



(a) Runtime (seconds)



(b) Normalized runtime

Fig. 15. Average (normalised) runtime over problem size for all models and maximum pairing lengths of 4 and 6 days (n = 12). Dotted lines and star markers indicate the use of a 6-day maximum pairing length.

CRediT authorship contribution statement

Johanna P. Korte: Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Neil Yorke-Smith:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Funding acquisition, Conceptualization.

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

This research was partially supported by the TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under grant number 952215. Thanks to W.-J. van Hoeve, S. van der Laan and K. Sidorov. The authors express thanks to the anonymous reviewers whose comments helped to improved this article.

Appendix A. Pairing graph construction

This appendix explains in detail how the pairing graph is constructed. The pairing graph consists of nodes representing bases (source and sinks), taxi activities, taxi+overnight activities, and duties. The edges that are possible between all these nodes are given in Fig. 16. Each edge has an associated weight, representing the amount of idle time: the available duty time not used to operate flights. Each edge also contains a resource cost attribute that records the time needed to execute the pairing. The models that use branch-and-price to find an optimal set of pairings need this attribute to ensure that the created pairings do not violate the maximum pairing length. To construct the pairing graph, all nodes are created first, after which edges and their corresponding weight and resource costs are created. This is done by following the steps presented in Section 3.1.2.

Because there is only one taxi node per base source node, the edges from the source node to the taxi node have both weight and resource cost 0. The cost of using a taxi is put into the edge attributes of the edges from the taxi node to the first duty. For the weight, these edges get their usual weight as edges going into a duty node, but are increased by the taxi time from the source to the duty's departure airport. This also holds for the resource cost attributes. Similarly, the edges going to a taxi node just include the taxi time as its weight and resource cost. A full overview can be seen in Fig. 16.

To improve the runtime of all models, and recognising the size of the graph (Table 10), the pairing graph is kept as small as possible. This is done by iterating over the pairing graph and removing all nodes that have an out-degree or in-degree of 0. By iteratively removing these nodes it is ensured that edges and nodes that cannot be part of a path between a source and a sink node do not remain in the graph and therefore do not slow down the process.

Appendix B. Routing graph construction

This appendix explains in detail how the routing graphs are constructed for all models presented in the article. Firstly, it is important to remember that each given aircraft type has its own routing graph, to simplify finding a solution. To construct a routing graph for one of the aircraft types, a source and sink node is created for all bases. Additionally, for all flights that are assigned to that aircraft type, a node is created.

The process of assigning edges between these nodes is straightforward, and only happens when there is enough turn around time between the two flights. Additionally, all flights that depart from a base get an incoming edge from the source node of that base, while all flights that arrive at a base get an outgoing edge to the base's sink node.

The edges in the routing graphs only have one attribute: weight. Similar to the pairing graph, the weight is represented as the idle time of the aircraft (in seconds), as a consequence this also minimises the number of aircraft that are used to cover all flights. This is accomplished by giving outgoing edges from the source node a weight that corresponds to the difference between the departure time of the node's flight and the first flight in the schedule. Similarly, all incoming edges to the sink nodes get a weight that represents the time between the arrival time of the flight and the arrival time of the last flight in the schedule.

As explained in Section 3.1.3, the priority between optimising pairings and aircraft routes can be shifted by adjusting the w (i.e., relative weight) parameter. All edge weights in the routing graphs are multiplied by this parameter. All possible edges and their respective weights can be observed from Fig. 17.

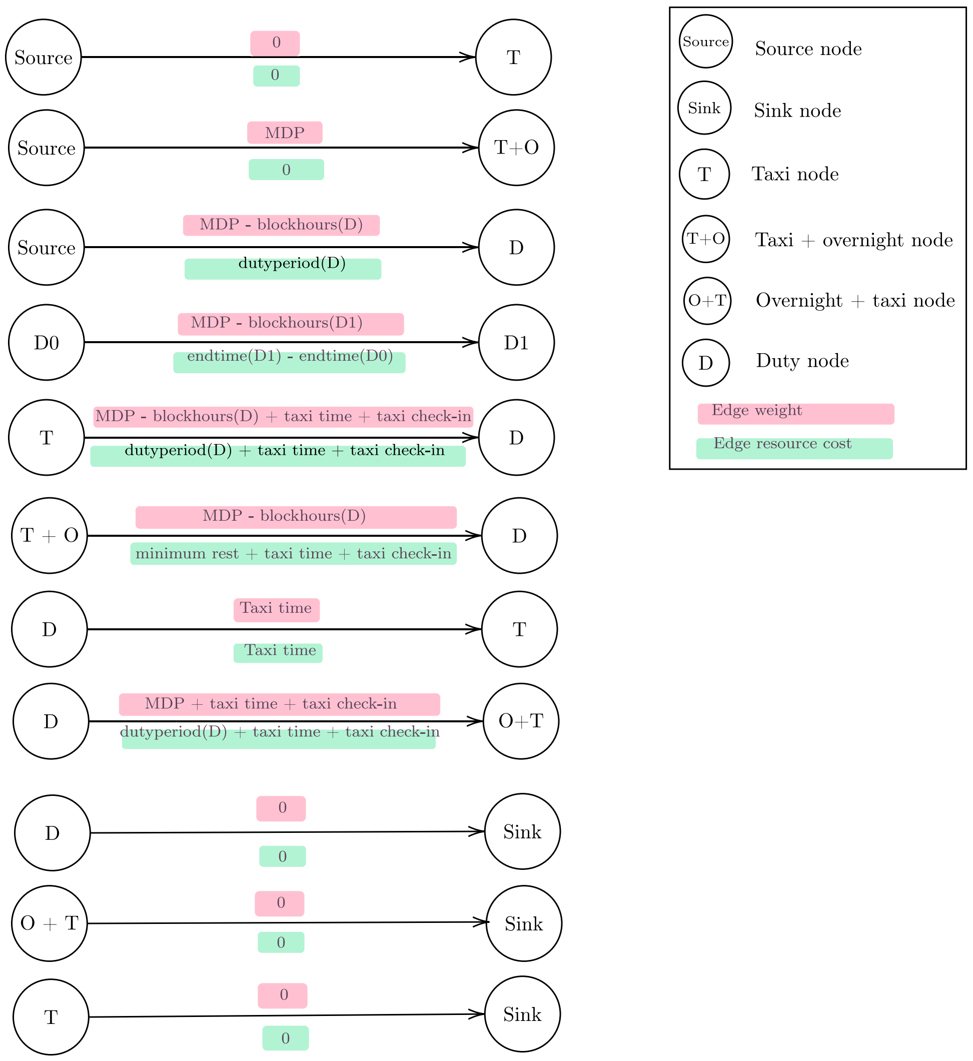


Fig. 16. All possible edges in the pairing graph, with their corresponding weights and resource costs.

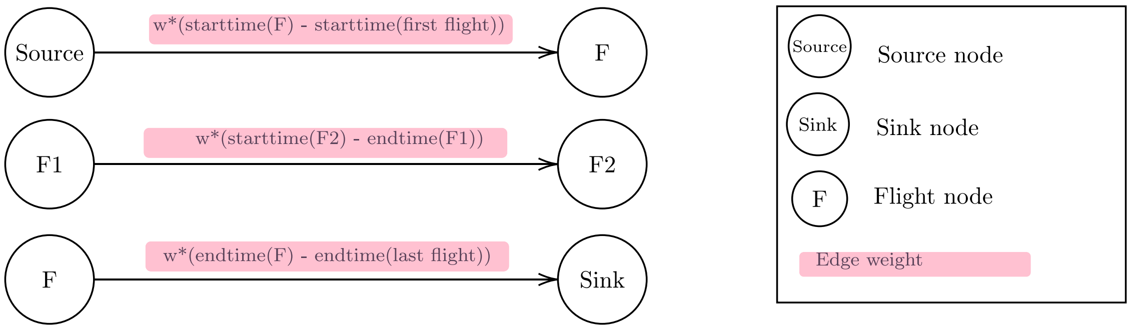

Source
w*(starttime(F) - starttime(first flight))
F
F1
w*(starttime(F2) - endtime(F1))
F2
F
w*(endtime(F) - endtime(last flight))
Sink
Source node
Sink node
Flight node
Edge weight

Fig. 17. All possible edges in the routing graphs and their respective weights.

References

Agustin, Alba, Juan, Angel, Pardo, Eduardo G., 2017. A variable neighborhood search approach for the crew pairing problem. Electron. Notes Discrete Math. 58, 87–94. http://dx.doi.org/10.1016/j.endm.2017.03.012.

Airport Coordination Netherlands, 2019.Declared Capacity AMS Summer 2019. URL https://slotcoordination.nl/slot-allocation/declared-capacity/.

Aydemir-Karadag, Ayyuce, Dengiz, Berna, Bolat, Ahmet, 2013. Crew pairing optimization based on hybrid approaches. Comput. Ind. Eng. 65 (1), 87–96. http://dx.doi.org/10.1016/j.cie.2011.12.005.

Azadeh, Ali, Farahani, Mehdi Hosseinabadi, Eivazy, H., Nazari-Shirkouhi, Salman, Asadipour, G., 2013. A hybrid meta-heuristic algorithm for optimization of crew scheduling. Appl. Soft Comput. J. 13 (1), 158–164. http://dx.doi.org/10.1016/j.asoc.2012.08.012.

Beasley, J.E., Chu, P.C., 1996. A genetic algorithm for the set covering problem. European J. Oper. Res. 94, 392–404. http://dx.doi.org/10.1016/0377-2217(95)00159-X.

Bellman, Richard, 1958. On a routing problem. Quart. Appl. Math. 87–90.

Ben Ahmed, Mohamed, Hryhoryeva, Maryia, Hvattum, Lars Magnus, Haouari, Mohamed, 2022. A matheuristic for the robust integrated airline fleet assignment, aircraft routing, and crew pairing problem. Comput. Oper. Res. 137, 105551. http://dx.doi.org/10.1016/j.cor.2021.105551.

Ben Ahmed, Mohamed, Zeghal Mansour, Farah, Haouari, Mohamed, 2018. Robust integrated maintenance aircraft routing and crew pairing. J. Air Transp. Manag. 73, 15–31. http://dx.doi.org/10.1016/j.jairtraman.2018.07.007.

Bengio, Yoshua, Lodi, Andrea, Prouvost, Antoine, 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. European J. Oper. Res. 290 (2), 405–421. http://dx.doi.org/10.1016/J.EJOR.2020.07.063.

Boland, Natashia, Dethridge, John, Dumitrescu, Irina, 2006. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Oper. Res. Lett. 34 (1), 58–68. http://dx.doi.org/10.1016/j.orl.2004.11.011.

Butchers, E. Rod, Day, Paul R., Goldie, Andrew P., Miller, Stephen, Meyer, Jeff A., Ryan, David M., Scott, Amanda C., Wallace, Chris A., 2001. Optimized crew scheduling at Air New Zealand. Interfaces 31 (1), 30–56. http://dx.doi.org/10.1287/INTE.31.1.30.9688.

Cacchiani, Valentina, Salazar-González, Juan José, 2013. A heuristic approach for an integrated fleet-assignment, aircraft-routing and crew-pairing problem. Electron. Notes Discrete Math. 41, 391–398. http://dx.doi.org/10.1016/j.endm.2013.05.117.

Cacchiani, Valentina, Salazar-González, Juan-José, 2020. Heuristic approaches for flight retiming in an integrated airline scheduling problem of a regional carrier. Omega 91, 102028. http://dx.doi.org/10.1016/j.omega.2019.01.006.

2017. Collectieve Arbeidsovereenkomst Transavia Vliegers 2013–2016. URL http://cao.minszw.nl/pdf/174/2017/174_2017_13_238976.pdf.

Chen, Chiu-Hung, Chou, Fu-I, Chou, Jyh-Horng, 2020. Multiobjective evolutionary scheduling and rescheduling of integrated aircraft routing and crew pairing problems. IEEE Access 8, 35018–35030. http://dx.doi.org/10.1109/ACCESS.2020.2974245.

Chu, Hai D., Gelman, Eric, Johnson, Ellis L., 1997. Solving large scale crew pairing problems. European J. Oper. Res. 97, 260–268.

Cook, Gerald, Goodwin, Jeremy, 2008. Airline networks: A comparison of hub-and-spoke and point-to-point systems. J. Aviat./Aerosp. Educ. Res. 17 (2), http://dx.doi.org/10.15394/jaaer.2008.1443.

Cordeau, Jean-François, Stojković, Goran, Soumis, François, Desrosiers, Jacques, 2001. Benders decomposition for simultaneous aircraft routing and crew scheduling. Transp. Sci. 35 (4), 375–388. http://dx.doi.org/10.1287/trsc.35.4.375.10432.

Deng, Guang-Feng, Lin, Woo-Tsong, 2011. Ant colony optimization-based algorithm for airline crew scheduling problem. Expert Syst. Appl. 38, 5787–5793. http://dx.doi.org/10.1016/j.eswa.2010.10.053.

Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M.M., Soumis, F., 1997. Crew pairing at Air France. European J. Oper. Res. 97 (1), 245–259. http://dx.doi.org/10.1016/S0377-2217(96)00195-6.

Desrosiers, Jacques, Lübbecke, Marco E., 2011. Branch-price-and-cut algorithms. In: Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, http://dx.doi.org/10.1002/9780470400531.eorms0118.

Deveci, Muhammet, Demirel, Nihan Çetin, 2018a. Evolutionary algorithms for solving the airline crew pairing problem. Comput. Ind. Eng. 115, 389–406. http://dx.doi.org/10.1016/j.cie.2017.11.022.

Deveci, Muhammet, Demirel, Nihan Çetin, 2018b. A survey of the literature on airline crew scheduling. Eng. Appl. Artif. Intell. 74, 54–69. http://dx.doi.org/10.1016/j.engappai.2018.05.008.

Dumouchelle, Justin, Patel, Rahul, Khalil, Elias B., Bodur, Merve, 2022. Neur2SP: Neural two-stage stochastic programming. http://dx.doi.org/10.48550/ARXIV.2205.12006, CoRR, abs/2205.12006.

Erdogan, Günea, Haouari, Mohamed, Matoglu, Melda Örmeci, Özener, Okan Örsan, 2015. Solving a large-scale crew pairing problem. J. Oper. Res. Soc. 66, 1742–1754. http://dx.doi.org/10.1057/jors.2015.2.

Ford, Lester R., 1956. Network flow theory. RAND Corp. URL https://www.rand.org/pubs/papers/P923.html.

Golden, Darragh, Erne, Roland, 2022. Ryanair pilots: Unlikely pioneers of transnational collective action. Eur. J. Ind. Relat. 28 (4), 451–469. http://dx.doi.org/10.1177/09596801221094740.

Hoffman, Karla L., Padberg, Manfred, 1993. Solving airline crew scheduling problems by branch-and-cut. Manag. Sci. 39 (6), 657–682. http://dx.doi.org/10.1287/mnsc.39.6.657.

Kasirzadeh, Atoosa, Saddoune, Mohammed, Soumis, François, 2017. Airline crew scheduling: models, algorithms, and data sets. EURO J. Transp. Logist. 6, 111–137. http://dx.doi.org/10.1007/s13676-015-0080-x.

Kenan, Nabil, Jebali, Aida, Diabat, Ali, 2018. The integrated aircraft routing problem with optional flights and delay considerations. Transp. Res. Part E: Logist. Transp. Rev. 118, 355–375. http://dx.doi.org/10.1016/j.tre.2018.08.002.

Khiabani, A., Rashidi Komijan, A., Ghezavati, V., Mohammadi Bidhandi, H., 2023. A mathematical model for integrated aircraft and crew recovery after a disruption: a Benders' decomposition approach. J. Model. Manag. 18 (6), 1740–1761. http://dx.doi.org/10.1108/JM2-02-2022-0046.

Klabjan, Diego, Johnson, Ellis L., Nemhauser, George L., Gelman, Eric, Ramaswamy, Srini, 2001. Solving large airline crew scheduling problems: Random pairing generation and strong branching. Comput. Optim. Appl. 20 (1), 73–91. http://dx.doi.org/10.1023/A:1011223523191.

Klabjan, Diego, Johnson, Ellis L., Nemhauser, George L., Gelman, Eric, Ramaswamy, Srini, 2003. Airline crew scheduling with regularity. Transp. Sci. 35 (4), 359–374. http://dx.doi.org/10.1287/trsc.35.4.359.10437.

Kohl, Niklas, Karisch, Stefan E., 2004. Airline crew rostering: Problem types, modeling, and optimization. Ann. Oper. Res. 127, 223–257. http://dx.doi.org/10.1023/B:ANOR.0000019091.54417.ca.

Lam, Edward, Desaulniers, Guy, Stuckey, Peter J., 2022. Branch-and-cut-and-price for the electric vehicle routing problem with time windows, piecewise-linear recharging and capacitated recharging stations. Comput. Oper. Res. 145, 105870. http://dx.doi.org/10.1016/J.COR.2022.105870.

Li, Yuewen, Wang, Xiaolin, Kang, Qi, Fan, Zheng, Yao, Shuaiyu, 2023. An MCTS-based solution approach to solve large-scale airline crew pairing problems. IEEE Trans. Intell. Transp. Syst. 24, 5477–5488. http://dx.doi.org/10.1080/01605682.2023.2253839.

Mercier, Anne, Cordeau, Jean François, Soumis, François, 2005. A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. Comput. Oper. Res. 32, 1451–1476. http://dx.doi.org/10.1016/j.cor.2003.11.013.

Mercier, Anne, Soumis, François, 2007. An integrated aircraft routing, crew scheduling and flight retiming model. Comput. Oper. Res. 34, 2251–2265. http://dx.doi.org/10.1016/j.cor.2005.09.001.

Ozdemir, H. Timucin, Mohan, Chilukuri K., 2001. Flight graph based genetic algorithm for crew scheduling in airlines. Inform. Sci. 133, 165–173. http://dx.doi.org/10.1016/S0020-0255(01)00083-4.

Papadakos, Nikolaos, 2009. Integrated airline scheduling. Comput. Oper. Res. 36, 176–195. http://dx.doi.org/10.1016/j.cor.2007.08.002.

Pothos, Dionisios, Richards, Barry, Stokic, Davor, 1994. Aircraft allocation in a parallel constraint logic programming environment. In: Proceedings of the 34th AGIFORS Symposium. Zandvoort, The Netherlands.

Quesnel, Frédéric, Desaulniers, Guy, Soumis, François, 2017. A new heuristic branching scheme for the crew pairing problem with base constraints. Comput. Oper. Res. 80, 159–172. http://dx.doi.org/10.1016/j.cor.2016.11.020.

Quesnel, Frédéric, Desaulniers, Guy, Soumis, François, 2020. Improving air crew rostering by considering crew preferences in the crew pairing problem. Transp. Sci. 54 (1), 97–114.

Quesnel, Frédéric, Wu, Alice, Desaulniers, Guy, Soumis, François, 2022. Deep-learning-based partial pricing in a branch-and-price algorithm for personalized crew rostering. Comput. Oper. Res. 138, 105554. http://dx.doi.org/10.1016/j.cor.2021.105554.

Rubin, Jerrold, 1973. A technique for the solution of massive set covering problems, with application to airline crew scheduling. Transp. Sci. 7 (1), 34–48. http://dx.doi.org/10.1287/trsc.7.1.34.

Saddoune, Mohammed, Desaulniers, Guy, Elhallaoui, Issmail, Soumis, Franois, 2011. Integrated airline crew scheduling: A bi-dynamic constraint aggregation method using neighborhoods. European J. Oper. Res. 212, 445–454. http://dx.doi.org/10.1016/j.ejor.2011.02.009.

Saddoune, Mohammed, Desaulniers, Guy, Soumis, François, 2013. Aircrew pairings with possible repetitions of the same flight number. Comput. Oper. Res. 40, 805–814. http://dx.doi.org/10.1016/j.cor.2010.11.003.

Sandhu, Rivi, Klabjan, Diego, 2007. Integrated airline fleeting and crew-pairing decisions. Oper. Res. 55 (3), 439–456. http://dx.doi.org/10.1287/opre.1070.0395.

Torres Sanchez, David, 2019. `cspy` – A Python package with a collection of algorithms for the (resource) constrained shortest path problem. URL https://github.com/torressa/cspy.

Vanderbeck, François, 2000. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Oper. Res. 48 (1), 111–128. http://dx.doi.org/10.1287/OPRE.48.1.111.12453.

Wen, Xin, Ma, Hoi-Lam, Chung, Sai-Ho, Khan, Waqar Ahmed, 2020. Robust airline crew scheduling with flight flying time variability. Transp. Res. Part E: Logist. Transp. Rev. 144, 102132. http://dx.doi.org/10.1016/j.tre.2020.102132.

Wen, Xin, Sun, Xuting, Sun, Yige, Yue, Xiaohang, 2021. Airline crew scheduling: Models and algorithms. Transp. Res. Part E: Logist. Transp. Rev. 149, 102304. http://dx.doi.org/10.1016/j.tre.2021.102304.

Weyens, Toon, van Vugt, Daan, 2019. `Pylgrim` – Elementary Shortest Path Problem with or without Resource Constraint. URL https://github.com/ToonWeyens/pylgrim.

Xu, Yifan, Wandelt, Sebastian, Sun, Xiaoqian, 2023. Airline scheduling optimization: Literature review and a discussion of modeling methodologies. Intell. Transp. Infrastruct. 3, liad026. http://dx.doi.org/10.1093/iti/liad026.

Yan, Shangyao, Tung, Tun-Tai, Tu, Yu-Ping, 2002. Optimal construction of airline individual crew pairings. Comput. Oper. Res. 29, 341–363. http://dx.doi.org/10.1016/S0305-0548(00)00070-8.

Zeighami, Vahid, Saddoune, Mohammed, Soumis, François, 2020. Alternating Lagrangian decomposition for integrated airline crew scheduling problem. European J. Oper. Res. 287 (1), 211–224. http://dx.doi.org/10.1016/j.ejor.2020.05.005.

Zeren, Bahadir, Özkol, Äbrahim, 2012. An improved genetic algorithm for crew pairing optimization. J. Intell. Learn. Syst. Appl. 4 (1), 70–80. http://dx.doi.org/10.4236/jilsa.2012.41007.

Zeren, Bahadir, Özkol, Ibrahim, 2016. A novel column generation strategy for large scale airline crew pairing problems. Expert Syst. Appl. 55, 133–144. http://dx.doi.org/10.1016/j.eswa.2016.01.045.