Solving a Large-Scale Crew Pairing Problem

Güneş Erdoğan * Mohamed Haouari † Melda Örmeci Matoglu ‡
Okan Örsan Özener §

August 29, 2014

Abstract

Airline companies seek to solve the problem of determining an assignment of crews to a pre-determined flight schedule with minimum total cost, called the *Crew Pairing Problem* (CPP). Most of the existing studies focus on the CPP of North American airlines, which widely differs from that of most European airline companies in terms of the objective function, the flight structure, and the planning horizon. In this study, we develop an optimization-driven heuristic algorithm that can efficiently handle large-scale instances of the CPP that must be solved on a monthly basis. We perform computational experiments using flight schedules of a European airline company to test the performance of the solution method. Our computational results demonstrate that our algorithm is able to provide high quality solutions to monthly instances with up to 27,000 flight legs.
Keywords: crew pairing problem, set partitioning, metaheuristic, monthly problem

1 Introduction

To increase the efficiency of their operations, airlines use optimization techniques for tackling their many interdependent and challenging planning problems. Each of these problems has its own considerations, complexity, and objectives. In practice, the planning processes start with the construction of the flight schedule, a list of flights to be operated in a period of time (usually, 4 to 6 months). This schedule usually depends on the expected demand for the flight segments as well as the fleet size of the airline. Next, aircraft are assigned to the scheduled flight legs based on their types and capacities with the objective of maximizing the net profit, which is called the Fleet Assignment Problem. The next step is to solve the aircraft routing problem where maintenance requirements of the aircraft are considered. After that, the airline must assign cockpit and cabin crew for each of the scheduled flights, where the crew scheduling problems arise. Typically a particular cockpit crew is qualified to fly aircraft belonging to only one family, hence the crew scheduling problem can be decomposed for different fleet types. The objective of the crew scheduling problem is to minimize

*University of Southampton, School of Management, Highfield, Southampton, SO17 1BJ, United Kingdom G.Erdogan@soton.ac.uk

†Department of Mechanical and Industrial Engineering College of Engineering, Qatar University, P.O. Box 2713, Doha, State of Qatar mohamed.haouari@qu.edu.qa

‡Peter T. Paul College of Business and Economics, University of New Hampshire, 10 Garrison Avenue, Durham, NH, USA. Melda.OrmeciMatoglu@unh.edu

§Ozyegin University, School of Engineering, İstanbul, Turkey orsan.ozener@ozyegin.edu.tr

the crew related costs. An intermediate, but crucial, stage before building individual crew schedules requires solving the so-called *Crew Pairing Problem* (CPP). In this context, a *pairing* refers to a sequence of duty periods (i.e. single crew workdays) that starts from and end at the crew base. The feasibility of crew duties and pairings depends on a set of complex rules imposed by government, international organizations (e.g. International Civil Aviation Organization), labor unions, and airline companies themselves. Provided that there exists tens of thousands of flights in a month and thousands of cockpit and cabin crews, even determining a feasible crew pairing is a daunting task.

In this paper, we investigate novel solution approaches for the CPP. The relevance of the CPP stems from the fact that crew related costs constitute a major chunk of cost (second next to fuel costs) for commercial airlines (Gopalakrishnan and Johnson, 2005). As fuel price is usually uncontrollable, crew cost is the biggest operating cost item on which an airline company has control. Therefore, reducing the crew costs may offer significant savings for a commercial airline company, particularly under intense competition. The CPP is a highly complex and large-scale optimization problem as it often involves thousands of flights and crews, and requires advanced and efficient optimization techniques to solve. During the last three decades, the CPP has been intensely investigated in the operations research literature and often used as a testbed for assessing cutting-edge optimization techniques including interior point methods, sophisticated branch-and-price algorithms, and decomposition methods (Makri and Klabjan 2004, Subramanian and Sherali 2008, Sherali et al. 2010). However, a glaring fact is that a great majority of these research efforts address variants of the CPP that arise at North-American airlines.

In the sequel, we focus on the CPP arising at a European Airline Company (hereafter, referred to as EAC). Unlike most previous related contributions published so far that focus on *daily* CPP variants, we investigate in this study a *monthly* variant. Actually, the CPP that is investigated in the present paper significantly differs from most of the CPPs that were studied in the literature in three important aspects: (*i*) schedule periodicity, (*ii*) flight network structure, and (*iii*) cost structure. These distinctive features are detailed in Section 2. Consequently, the resulting CPP monthly instances that we address are typically huge as they usually involve tens of thousands of flights, while the largest solved CPP instances that were reported in the literature barely exceed 1,000 flights. In this paper, we make the following contributions:

- We describe a novel solution methodology that aims at solving a large-scale monthly CPP arising at a major European airline. The proposed solution approach is based on a combination of several optimization paradigms: large neighborhood search, exact enumerative algorithms, and integer programming.
- We report the results of extensive computational experiments on a large set of real data with up to 27,000 flights (in major U.S. airlines the number of daily flights are typically in the order of 1-2,000). Our results provide evidence that the proposed approach is highly competitive and delivers multiple high-quality solutions within an acceptable time limit. Furthermore, the different solutions provide the decision-maker with the flexibility to choose one which better meets their operational concerns.

The remainder of the paper is organized as follows. In Section 2, we provide a detailed description of the problem. In Section 3, we present review of the literature. In Section 4, we describe the algorithmic expedients of the proposed solution methodology. Several variants of increasing levels of sophistication will be presented. In Section 5, we will present the results of our computational experiments. Finally, in Section 5 we conclude by presenting a summary of our contributions and we outline some directions for future research.

2 Problem description

In this section, we introduce the relevant terminology, and provide some specific features of the investigated problem while highlighting major differences with CPP variants that are often studied in the literature. We will also present a formal problem description together with a valid formulation.

2.1 Generalities and terminology

As stated in the Introduction section, airline planning processes start with the *Schedule Generation Problem*. This involves the construction of the flight schedule, a list of flights to be operated in a given time period (usually one to four months). This schedule usually depends on the expected demand for the flight segments as well as the fleet size of the airline. Next, aircrafts should be assigned to the scheduled flight legs based on their types and capacities with the objective of maximizing the net profit, which is called the *Fleet Assignment Problem*. The next step requires solving the *Aircraft Routing Problem* where a specific tail number is assigned to each flight while accommodating aircraft maintenance requirements. After that, the airline must assign cockpit and cabin crew for each of the scheduled flights. Given that a crew is typically qualified to fly only a specific aircraft family, the crew scheduling problem is solved for each aircraft family. This crew scheduling process is usually achieved in two stages: (*i*) Solving the *Crew Pairing Problem*, and (ii) Solving the *Crew Rostering Problem*.

In the crew pairing stage, a set of crew schedules is generated to cover each flight leg scheduled and in the crew rostering stage the generated schedules are combined to create monthly rosters to be flown by the available crews. The former stage begins with generating feasible duties. A *duty* is a sequence of flight legs in a single workday of a crew followed by a rest period. It also includes the time for briefing and debriefing meetings. Each duty has an associated cost. Next, these duties are combined to produce feasible pairings. A *pairing* is a sequence of duties that starts at a crew base and ends at the same crew base with an overnight rest. Typically, a pairing spans one to five days and has a well defined associated cost. For North American airlines, the cost structure is a nonlinear function of three quantities: (*i*) the number of duties in the pairing, (*ii*) the total time away from base (each crew has a well identified base), and (*iii*) the sum of duty costs in the pairing, where the cost of a duty is a function of the duty duration, the total flying time, and the minimum guaranteed pay.

The feasibility of a crew schedule depends on numerous complex work rules imposed by government, international organizations (e.g. International Civil Aviation Organization), labor unions, and airline companies themselves. Typical work-rule restrictions include limits on the number of hours flown in a duty, the number of flights flown in a duty, the rest durations, and the time the crew may be away from their base. Given a set of flight legs to be served by an aircraft family, the CPP seeks to find a minimum-cost set of pairings that covers the set of flights. Due to the combinatorial explosion of the number of the pairings and the complexity of the associated feasibility restrictions, solving the CPP to optimality is a daunting task. However, the benefits of solving CPP largely outweigh its challenges as small improvements can lead to significant savings. Additionally, an efficient solution will offer improved operational crew planning process, decreased crew requirements, and increased employee satisfaction.

2.2 Specific features of the CPP arising at EAC

In this section we provide the details of the CPP arising at EAC that distinguishes it from the CPPs in the literature.

2.2.1 The flight schedule

The flight schedule of the Airbus 320 family includes approximately 12,000 flights per month, to more than 100 destinations. Most of these flights are short- and medium-haul flights that usually take one to five hours to complete. In addition, EAC operates a Boeing 738 aircraft family that serves the same destinations. Each month, both families are used for serving about 25,000 flights. Interestingly, the flight schedule of EAC, similar to most European airlines, does not exhibit any daily periodicity, and exhibits limited weekly periodicity. Indeed, to better match capacity with the varying demand, different aircraft types (either from the Airbus 320 or the Boeing families) might be assigned to the same daily flight. So, for example, while a certain flight is assigned to an A321 on a given day, the next day it may be assigned to a B737. Consequently, to derive effective crew schedules we model the CPP as a *monthly* problem. At this point, it is worth mentioning that CPPs that are traditionally addressed in the literature assume that all flight legs are operated on every day. This assumption is consistent with common practices at North American airlines where the daily problem is solved and adjustments are made for weekly or monthly exceptions. Clearly, the daily problem has significantly lower number of flight legs than the monthly variant.

2.2.2 The flight network structure

An interesting feature of EAC's flight network is that there is only one hub (or, crew home base). Consequently, more than 80% of the scheduled flights are either originated or terminated in this hub. In what follows, we shall refer to this hub as BASE. Furthermore, a significant portion of the flights to several domestic and most international destinations are *round trip* type flights. Here, we define a round trip as a sequence of two consecutive flights that originate from and end at the home base and are *always* served by the same crew and aircraft. For example, if there is a flight from BASE to Rome in the morning on a given day, there is also a flight returning to BASE later that day, as there exists no other flights from Rome to any other destination. Therefore, neither the aircraft nor the crew will not stay in Rome for an additional day due to utilization purposes. As we shall see in Sections 4 and 5, this property of the flight network will prove useful for devising an effective tailored solution approach.

2.2.3 The cost structure

In contrast with North American airlines, at EAC (as in most European carriers) the crew salaries do not depend on the flight assignments, but are fixed. Hence, the objective function of the CPP is to minimize the crew related costs that include *deadheadings* (crews being transported to a flight's departure city as passengers), and *layovers*. These latter costs correspond to the accommodation, ground transportation, and subsistence expenses of the crew members when they rest away from BASE. For a connection of more than 4 hours, half the cost of a layover is incurred. This is due to the fact that, for long connection times, airlines typically get a day-room for their crew.

At this point it is worth emphasizing that even though a separate CPP is solved for each aircraft family, it is necessary to consider the whole set of legs while solving a problem that is defined for a specific family. Indeed, since the cost of a pairing includes the deadheadings, then a pairing of a specific family (say, the Airbus 320 family) might include a flight leg that is assigned to a different family (say, the Boeing 738) and that is used by the crew as a deadheading.

2.2.4 Restrictions on duties and pairings

A formidable challenge is posed by the great complexity of the rules that are used for building duties and pairings. In identifying these rules, we relied on the standard rules mentioned in the literature and policies used by EAC. We provide the following non-exhaustive list for illustrative purposes:

1. All pairings start and end at BASE. A pairing cannot continue past a duty that ends at BASE.
2. A landing is the number of flights a crew flies on in a duty. The maximum number of landings in a duty is restricted to 5.
3. The minimum and maximum number of duties in a pairing are one and six, respectively.
4. The maximum total duty time is : (*i*) Fourteen hours if the duty start time lies with the interval [05:00, 14:00], (*ii*) Thirteen hours if the duty start time lies with the interval [14:01, 17:00], (*iii*) Twelve hours if the duty start time lies with the interval [17:01, 04:59]. However, these total duty times differ depending on the number of landings in a duty, whether the duty is extended range, or performed with single augmented (a cabin crew of three) / double augmented (a cabin crew of four) crew.
5. The minimum connection time for a crew is 30 minutes, if they are not changing aircraft and both flights are either domestic or international. However, this time changes based on whether there is a change of aircraft, domestic/international connections, preceding or following active flights/deadheadings.
6. A rest period starts at the end of a duty and ends at the beginning of the next duty. The duration of a rest period is determined by a function of the duty time and flight time. Depending on whether the duty is a one leg duty, extended range, or a duty with certain duration the rest period may differ.
7. The total duty time during one pairing should not exceed 56 hours.

2.3 Problem definition and formulation

Given the monthly schedule of flights and an aircraft family, CPP requires finding a minimum-cost set of pairings such that exactly one feasible pairing is assigned to each scheduled flight. The objective function is a weighted sum of the deadheading costs and the layover costs. We now provide the notation we will be using for describing the CPP.

Notation

L : set of legs to be covered, indexed by l

P : set of all pairings, indexed by p

P_l : subset of pairings that cover leg $l, l \in L$

γ_1 : cost of the international layovers that are included in a pairing if any, $p \in P$

γ_2 : cost of the domestic layovers that are included in a pairing if any, $p \in P$

γ_3 : cost of the deadheadings that are included in a pairing if any, $p \in P$

c_p : total cost of pairing p (viz. , $c_p = w_1\gamma_1 + w_2\gamma_2 + w_3\gamma_3$, where w_1, w_2 and w_3 are weight parameters), $p \in P$

x_p : binary decision variable that takes value of 1 if pairing p is selected, and 0 otherwise, $p \in P$.

A standard formulation of the CPP as a Set Partitioning Problem (SPP) is the following:

$$\min \sum_{p \in P} c_p x_p \tag{1}$$

subject to

$$\sum_{p \in P_l} x_p = 1, \qquad \forall l \in L, \tag{2}$$

$$x_p \in \{0,1\}, \quad \forall p \in P. \tag{3}$$

The objective function (1) minimizes the total cost of the solution while the partitioning constraint (2) enforces that exactly one pairing is assigned to each leg. It is worth emphasizing that pairings might include deadheadings (and therefore multiple pairings might cover a given leg), but (2) requires that exactly one crew will be assigned to serve that leg.

As a final note, we emphasize that the differences mentioned in the previous subsections result in a CPP that is at least as hard as the CPP of the North American Airlines. This is due to the fact that the flight schedule of the monthly problem has significantly more legs with respect to the daily and weekly problems. Despite the simplifying features of the flight network structure, cost structure, and the restrictions on costs and duties, the sheer size of the problem places it outside the computational reach of the algorithms in the literature.

3 Literature Review

CPP is generally considered as the hardest of the airline planning problems and has been extensively studied in the literature. We refer the interested reader to Gopalakrishnan and Johnson (2005), Arabeyre et al. (1969), Barnhart et al. (2003b,a), Desaulniers et al. (1998) and references therein. Even for small size problems it is not possible to generate all feasible pairings in a reasonable amount of time. In fact, because of the extensive list of complex rules and regulations, determining whether a sequence of flights constitutes a feasible duty can be quite difficult by itself. For example, Vance (1993) reports of an instance with 253 legs to have more than 5 million feasible pairings and states that an instance with 1000 legs is likely to have more than billions of pairings. This exponential growth in problem size is one of the biggest disadvantage of set partitioning (as well as covering) models. CPPs with these models mostly differ in their enumeration techniques (Andersson et al. 1998).

In early works to overcome the problem size issue heuristic approaches were employed to generate just a small subset of the pairings (Arabeyre et al. 1969, Anbil et al. 1991, Gershkoff 1989, Hoffman and Padberg 1993) and the resulting small SPPs were solved. The main shortcoming of this approach is that it does not provide any quantitative information regarding the quality of the solution. Notably, a pairing with a low cost may be incompatible with the rest of the selected pairings, and may require other pairings with multiple deadheadings and layovers to cover a given leg. Consequently, a poor choice of columns may result in severe suboptimality. To overcome this shortcoming, branch-and-price algorithms have been used, where column generation is used at each node of the branch-and-bound tree to obtain linear programming bounds. Negative reduced cost columns are identified by solving a pricing problem which is usually formulated as a multi-label shortest-path problem where each path corresponds to a pairing (Barnhart et al. 2003a). Branch-and-price techniques have been used frequently in airline problems (Barnhart et al. 1994, Butchers et al. 2001, Desaulniers et al. 1998, Dück et al. 2011, Ryan 1992, Vance et al. 1997). Branch-and-price has a larger computational reach compared to branch-and-bound, however, the pricing algorithm may take a prohibitively long time for very large instances. Saddoune et al. (2011) and Saddoune et al. (2012) address the integrated CPP and rostering problem using a combined column generation and dynamic constraint aggregation method. While the integration helps obtain improvement in cost over sequential approach, computation time increases significantly.

Another common formulation type used in CPPs is network flow models. These models are built using a time-space network, where each node corresponds to a departure or arrival time and origin or destination pair. Desaulniers et al. (1997) formulate a CPP of Air France as an integer, nonlinear multi-commodity network flow problem with additional resource variables. In this model commodities represent crews, nodes represent time-airport pairs and arcs represent crew activities like active flights or deadheadings, connections, rests, briefing, debriefing, ground transportation or pairings. Using several auxiliary variables they also model crew pairing constraints and regulations. Their solution approach relies on a branch-and-price algorithm based on an extension of Dantzig-Wolfe decomposition. They implement their solution approach on the weekly schedule of the airline's fleets and solve CPPs with number of legs varying from 154 to 1157. They build a duty based network model which eliminates the need to check the feasibility of individual duties. The cost function consists of the sum of nonlinear, nondecreasing functions of time away from base, total flight time, number of landings, deadheadings, aircraft changes and night rests. Our work differs from this European application mainly in terms of problem size and cost structure, Desaulniers et al. (1997) uses a more general cost structure which encompasses part of the same cost items. Although we also rely on a duty based solution the size difference in the problems require the use of different solution methods. Barnhart and Shenoi (1998) also formulate a similar time-space network model with additional constraints. An interesting application of network models can be seen in Yan and Tu (2002) where the CPP of Taiwan Airline is solved. Due to the network structure of Taiwan Airline, the number of constraints on the pairings is significantly reduced. In fact, the authors are able to obtain an exact solution in polynomial time using network-simplex method.

Heuristic approaches have also been used in the literature to solve the CPP. Simulated annealing (Emden-Weinert and Proksch 1999), tabu search (Cavique et al. 1999), genetic algorithms (Levine 1996, Ozdemir and Mohan 2001), ant colony algorithms (Deng and Lin 2011), particle swarm optimization (Azadeh et al. 2013) are among the metaheuristics employed. Aydemir-Karadag et al. (2013) employ a hybrid algorithm that combines genetic based heuristics with zero-one integer programming models. Advantages of heuristics include speed and the size of instances that can be handled, whereas the quality of the solutions obtained can only be empirically demonstrated.

Recently there has been increasing interest in approaches addressing robustness, recovery models (e.g. Clausen et al. (2010), Tekiner et al. (2009), Muter et al. (2013)) and integrated models (e.g. Barnhart et al. (1998), Gao et al. (2009), Saddoune et al. (2012), Sherali et al. (2013), M. Dunbar and and Wu (2014)) that take into account several of the airline planning problems described above jointly. These approaches enable airlines to achieve costs closer to optimal values and make managing operations in the face of planned or unplanned schedule changes easier. In addressing these issues the ability to quickly solve the CPP becomes quite important. In fact, with our solution approach it is possible to obtain initial results in less than five minutes.

In this work we address the monthly problem of a EAC, with instances up to 27,000 legs. The large number of legs requires generating novel solution approaches, as the literature mostly addresses the daily or weekly problem with several hundreds to 1-2,000 legs. Furthermore different cost and network structures enables the use of different methods which include large-scale neighborhood search, exact enumerative algorithms, and integer programming.

4 Solution Method

We now describe an optimization-driven heuristic that is able to handle large-scale instances of the CPP. Optimization-driven heuristics (also known as model based metaheuristics) are combinations of metaheuristics and exact optimization methods that take advantage of the diversification provided by the metaheuristics as well as

the strength of the intensification provided by the exact optimization methods. The metaheuristic we present below constructs solutions for the CPP through a Large Neighborhood Search (LNS) heuristic, while the optimization-driven heuristic employs these solutions to solve an SPP that selects the best possible combination.

In the sequel, we should conform with the following terminology and notation. A *solution* S is a set of feasible pairings such that each flight leg is covered by at most one pairing. The cost of S, denoted by $c(S)$, is equal to $\sum_{p \in S} c_p$. We define a working vector of leg indices $\hat{L} := \{1, 2, ..., |L|\}$ that can be randomly reordered, and the i^{th} element of which is referred to as $\hat{L}(i)$. A solution is set to be *feasible* if each leg is covered exactly once.

The LNS is based on the following basic construction procedure. Given a feasible CPP solution S, we randomly remove from it a fraction of its pairings. In so doing, S becomes infeasible since a subset L' of legs now becomes uncovered. To reconstruct a feasible solution, we consider the uncovered legs in S' in random order, and we iteratively identify for each leg $l \in L'$ a *minimum-cost* pairing $p \in P$ that covers that leg. The pairing thus derived is included in the solution and the process is reiterated until no more legs can be covered and a new solution S' is obtained. This process is now restarted with S' being the new working solution, for k_{max} iterations.

Clearly, a crucial step in this local improvement procedure involves finding a minimum-cost pairing that covers a given leg. Provided, that each leg $l \in L$ might be potentially covered by hundreds of feasible duties and therefore millions of pairings, solving this subproblem might prove cumbersome. To solve it optimally, we propose an implicit enumeration procedure in the spirit of a branch-and-bound algorithm. We refer to this algorithm as *Restricted Minimum Cost Covering* (RMCC), the details of which we provide in Section 4.1.

The construction procedure can be embedded into an LNS framework to yield the following solution approach:

Procedure $LNS0$
Input: A set of legs L
Output: A feasible solution S^*
Construct the working vector of leg indices $\hat{L} := \{1, 2, ..., |L|\}$.
Initialize the working solution $S := \emptyset$ and the best known solution $S^* := \emptyset$.
For $k = 1$ **to** k_{max}
  Set $S := S^*$
  Randomly remove α percent of pairings from S
  Randomly reorder $\hat{L}$
  For $i = 1$ **to** $|L|$
    If leg $\hat{L}(i)$ is not covered by S
      Initialize the working pairing $p := \emptyset$ and the incumbent pairing $p^* := \emptyset$
      Apply RMCC$(\hat{L}(i), S, p, p^*)$
      $S := S \cup p^*$
  End For
  If S covers all legs and $c(S) < c(S^*)$
    Update the best known solution, $S^* := S$.
End For
Return S^*.

In the first iteration, LNS0 behaves as a constructive algorithm rather than an improvement algorithm, and the computing effort required for this iteration is significantly more than the following iterations. Starting with an empty solution, LNS0 covers legs in a greedy manner, in the chronological order of departure times. The solution thus constructed defines the initial solution of the CPP, and affects the performance of the overall algorithm. Albeit simple, LNS0 lacks the intensification ability

to provide high quality results. We will remedy this shortcoming by periodically solving SPPs. Before moving on to the optimization-driven heuristic, we provide details about the Restricted Minimum-Cost Covering algorithm.

4.1 The Restricted Minimum-Cost Covering algorithm

As briefly stated above, the RMCC is a branch-and-bound algorithm that aims to find a minimum cost pairing to cover a given leg. The algorithm incrementally constructs pairings starting from the BASE. The weighted sum of deadheadings and layovers incurred is used as a lower bound, and best feasible solution found through the search serves as an upper bound. In addition to pruning by bound, any duty that ends at BASE signals the end of a pairing, and the resulting pairing is pruned. Furthermore, a pairing that moves past the leg to be covered in terms of starting time without covering it, is pruned due to infeasibility. The algorithm is provided below.

Procedure RMCC(Mandatory leg l, solution S, working pairing p, incumbent pairing p^*)
If $p \neq \emptyset$ and $p^* \neq \emptyset$ and $c_p \geq c_{p^*}$
    Return // pruning by bound
If the last duty in p ends after the departure of leg l and p does not cover leg l
    Return // pruning by infeasibility
If p ends at BASE
    If p covers leg l, update the incumbent pairing $p^* := p$ // incumbent update
    Return // pruning due to forced end of pairing
If $p^* = \emptyset$
    For every duty $d \in D$ that starts at BASE up to 6 days before l // first duty
        If the active legs in d do not conflict with any active leg in S
            Add the duty to the working pairing, $p := p \cup d$
            RMCC(l, S, p, p^*)
            Remove the duty from the working pairing, $p := p \setminus d$
    End For
Else
    For every duty $d \in D$ that can connect to the last duty in p // next duty
        If the active legs in d do not conflict with any active leg in S
            Add the duty to the working pairing, $p := p \cup d$
            RMCC(l, S, p, p^*)
            Remove the duty from the working pairing, $p := p \setminus d$
    End For
Return p^*

Despite being an enumerative procedure, RMCC successfully finds solutions in a short time, and the number of legs already covered by S do not seem to have a major effect on the runtime of RMCC. In case S is sparse, RMCC finds simple pairings with no deadheadings and layovers, and quickly proves optimality. If S is dense, then the restrictions imposed by legs that are already covered limit the search space of the RMCC, helping it to converge fast. Through the course of a LNS run, RMCC first finds pairings with low cost. As the LNS progresses and S grows dense, the enumerative power of the RMCC comes into play and finds intelligent pairings with relatively low costs that can accommodate the restrictions. Towards the end of the LNS run, RMCC can only find high cost pairings that cover the remaining legs through multiple deadheadings and layovers. The resulting LNS solutions are consequently composed of the low cost, relatively low cost, and high cost pairings described above, which allows the SPP to find the best combinations among this diversity.

4.2 The intensification strategy

We now focus on improving the intensification of the LNS by solving SPPs. Let us define a *pairing pool* that consists of pairings found through the LNS, and denote it as π, which we will solve periodically. The resulting optimization-driven heuristic is described below.

Procedure $LNS1$
Input: A set of legs L
Output: A feasible solution S^*

Construct the working vector of leg indices $\hat{L} := \{1, 2, ..., |L|\}$.
Initialize the working solution $S := \emptyset$ and the best known solution $S^* := \emptyset$.
Initialize the pairing pool, $\pi := \emptyset$
For $k = 1$ **to** k_{max}
  Initialize $S := S^*$
  Randomly remove α percent of pairings from S
  Randomly reorder $\hat{L}$
  For $i = 1$ **to** $|L|$
    If leg $\hat{L}(i)$ is not covered by S
      Initialize the working pairing $p := \emptyset$, and the incumbent pairing $p^* := \emptyset$
      Apply RMCC$(\hat{L}(i), S, p, p^*)$
      $S := S \cup p^*$
  End For
  If S covers all legs and $c(S) < c(S^*)$
    Update the best known solution, $S^* := S$.
  Add the pairings in the S to the pairing pool, i.e. $\pi := \pi \cup S$.
  If the number of pairings in the pairing pool reach the limit, $|\pi| \geq p_{max}$
    // SPP
    Solve an instance of the SPP with π as the set of pairings
    Denote the optimal solution of the SPP as $\hat{S}$
    If $z(\hat{S}) < z(S^*)$
      Update the best known solution $S^* := \hat{S}$.
    Reinitialize the pairing pool, $\pi := S^*$.
End For
Return S^*.

The structure of LNS1 is similar to that of LNS0. The main difference is the existence of a *pairing pool* π, which is populated through the solutions found during the search and may contain multiple copies of pairings used in consecutive iterations. Once the number of pairings in the pairing pool exceeds p_{max}, an SPP is solved using the pairing pool, i.e. $P = \pi$. Since the pairing pool contains all the pairings in the best known solution S^*, the result of the SPP is no worse than the best known solution. If a better solution $\hat{S}$ is found as the result of the SPP, the best known solution is updated. The pairing pool is reinitialized to contain the pairings of the best known solution after solving the SPP.

4.3 Polishing methods and an acceleration feature

It is possible to further enhance LNS1 by defining a second pairing pool, π^*. Every time the best known solution is updated, the pairings in the new best solutions should be added to π^*. Solving a final SPP with π^* as the pairing set is a way of polishing the solution found by LNS1. This polishing method is aimed at using the information generated by the trajectory of the best known solution through the algorithm. We will refer to this algorithm as LNS2. The second polishing method populates π^* with

the final best known solutions of an instance for different weight combinations, and solves a final SPP using π^* and the original weight combination. This second method is conceived as a means of achieving the best possible solution, and is used for the purpose of evaluating the quality of the solutions returned by the other algorithms. We will refer to this algorithm as LNS3.

During our initial experiments, we observed that the runtime of RMCC is strongly dependent on the cardinality of the duty set $|D|$. Hence, we have looked for ways of reducing the cardinality of the duty set significantly without compromising the solution quality. To that end, we have exploited the flight network structure that results from the fact that EAC has a single hub with most of the flights either originating or ending at BASE. By identifying suitable pairs of flights in the leg list as round trips, and taking them out of the leg list, we have reduced the cardinality of the duty set significantly and therefore made the proposed solution approach faster.

A time extended graph of a sample schedule of flights between BASE and and another city, which we refer as XYZ, is presented in Figure 1. Using the concept of round trip, consecutive pairs of flights in the schedule constitute 7 duties, considering duties where one of the flights in these consecutive pairs is deadheading, results in a total of 21 duties. Without using the concept of round trip, every flight by itself is a duty as an active flight or a deadheading. The first flight departing from BASE can be complemented by any return flight, with at most one of the flights being a deadheading, and results in 21 possible duties. Similarly, the second flight can be complemented by six return flights, and results in 18 duty combinations. The total for this case turns out to be 112, more than five times the number of duties with round trips. Accounting for the 3, 4, and 5 leg duties in a similar manner, the grand total of duties becomes 340. Note that most of these duties are too costly to be practically relevant, and can be substituted by the round trips provided.

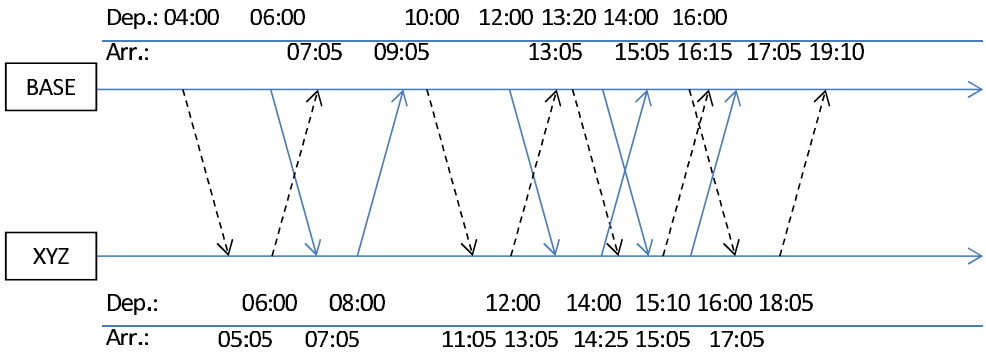

Figure 1: Round trips between BASE and XYZ

5 Computational experiments

In this section, we give the details of our extensive computational experiments based on the flight schedule of EAC to compare the proposed solution algorithms.

5.1 Setting and Data

We focus on the Airbus 320 (A320) family covering a 4-month period. At EAC, A320 and Boeing 737 (B737) families are frequently used for serving the same destinations, so we have included B737 family flights as potential deadheadings. We solve each

month's CPP as a separate instance. Not all crews are at BASE at the beginning and end of each month, since some are still on duty (performing a layover) at other stations. Since we do not have the data of the exact location and number of crews at each station, we include 1 week of warm-up period in the beginning and 1 week of "cool-down" period at the end of each month. Thus, 42 days of flights (a total of 2 weeks for warm-up and cool-down, and 4 weeks of flights that we actually try to solve the crew pairing problem) for two families are taken into account when solving each month's CPP.

We define a pair of matching flights BASE-XYZ and XYZ-BASE with less than $r_{max} = 180$ minutes of connection time (the maximum connection time for a crew at a domestic station) as a round trip. These flights cannot be used as one way flights connecting to other flights, and no other flights may be added before or after these flights. This reduces the cardinality of the duty set significantly, from approximately eight hundred thousand down to approximately two hundred fifty thousand, and makes the LNS approach tractable. However, for "well-connected" stations, which have many incoming and outgoing flights to many stations, the restriction deteriorates the solution quality as the solution space becomes very limited. To determine the "well-connected" stations, we have analyzed the flight network and identified domestic stations with the highest number of flights that either originate or end at that station. Among these we select 4 stations for which we set $r_{max} = 35$ minutes, since 30 minutes is the minimum connection time. In Table 1, the number of legs to be flown by A320 and B737 families is presented. The first and third columns display the number of legs in A320 and B737 families, respectively. The second and fourth columns show the number of legs in A320 and B737 families including the warm-up and cool-down periods. The fifth column includes all the legs in A320 and B737 families including the warm-up and cool-down periods, with an average of approximately 27,000 legs per monthly instance.

Table 1: Number of Legs

	Leg Data					**Number of Duties**	
Month	A320 Active	A320 Total	B737 Active	B737 Total	Total number of legs	Without Round trips	With Round trips
1	5428	8142	12088	18136	26278	763031	248070
2	5430	8151	12219	18368	26519	767694	249223
3	5536	8288	12623	18869	27157	813651	264501
4	5572	8358	12672	19008	27366	837556	271766

Since the crew salaries are fixed, while solving the CPP, we try to minimize the number of deadheadings as well as international and domestic layovers. The reported number of deadheadings and layovers are based on the number of people in the crew, e.g. a layover of a crew of two people counts as two layovers and a deadheading by a crew of three counts as three deadheadings. We test our algorithms' performance under different weight structures. In determining the weights to assign to deadheadings and layovers we take several factors into account. Airlines aim to maximize their occupancy rates in order to increase efficiency, and a seat allocated to a crew member in one of the legs can be costly. Furthermore, deadhead flights count as flight time for the crews, and consequently the deadheadings decrease the utilization efficiency of crews. Thus it is clear that deadheadings can be very costly for airlines and must be minimized. Another cost item is domestic and international layovers. Airlines typically have discounted rates with hotels for their crew, lower rates for domestic layovers and relatively higher rates for international layovers. Thus, we set the weight of domestic layovers to 1 and assign relative weights to the others, based on a brief market search on airline ticket prices and hotel room rates. For deadheadings we use

200, 100, 50, and 10 as weights, and for international layovers we use 100, 50, 10, and 5 as weights. These values are chosen to reflect the possible fluctuations of the ticket and hotel rates. In the sequel we shall refer to the experimental setting of weights of the objectives as 3-tuples of the form (weight of deadheadings, weight of international layovers, weight of domestic layovers), and compare the weighted objective function value.

We first run our two proposed methods, LNS0 and LNS1 with each months of data using different weight combinations. For both algorithms, we obtain and record several outputs from each run. We repeat the runs with different weight combinations to get relatively differentiated results, i.e. solutions with relatively fewer deadheadings as well as solutions with relatively fewer international layovers, which gives flexibility to an airline company to choose the solution that better suits their objectives. In our experimentation, we have set $k_{max} = 500$ for both LNS0 and LNS1 and $p_{max} = 25000$ for LNS1. The pairing removal parameter α was set to be 0.15 for LNS0, to retain the intensification power provided by the best known solution. Since we have relied upon the intensification ability of the SPP, a value of $\alpha = 0.7$ was used for LNS1 in order to maximize the diversification.

We then analyze the performances of LNS2 and LNS3. The potential for improvement is higher for LNS3 since we utilize the diversity of the solutions obtained by using different weight combinations. For example, when we combine the outputs of LNS0 using weight combinations of (100, 10, 1), (200, 5, 1) and (10, 50, 1) and solve a final SPP using the weight combination of (100, 10, 1), we are likely to obtain a better result than using only the output of LNS0 with weight combination of (100, 10, 1) only.

5.2 Comparison of the Proposed Algorithms

We have implemented algorithms LNS0, LNS1, LNS2, and LNS3 in C++, using CPLEX 12.3 as the mixed integer programming solver. In what follows, we will be presenting the results of these algorithms, using the result of LNS0 as a basis for measuring the quality of solutions. Although a more rigorous assessment of the quality of solutions is theoretically possible, it requires the exact solution of a relaxation of the problem, and developing an exact algorithm is beyond the scope of this paper.

In Table 2, we present the results of the LNS0 algorithm for Month 1 under different parameter sets. The column "Weights" presents the weights used for deadheadings, international and domestic layovers, respectively. The columns "DH", "IL", and "DL" presents the number of deadheadings, international layovers, and domestic layovers, respectively. Note that the values in the "IL" and "DL" columns can be fractional, due to the half day layovers for connections of more than 4 hours for the crew. The column "Obj." presents the objective function value of the corresponding solution. The first 16 rows display different weight combinations used and the final two rows presents the average and maximum improvement of the objective function value from the LNS0 solution.

In Table 3, we present results of the LNS1, LNS2, and LNS3 algorithms for the first month, where the column "Impr." presents the percentage improvement of the objective function value from the LNS0 solution. We observe from the result in Table 3, LNS1 improves the LNS0 solution by 43.40% on average. Furthermore, LNS2 and LNS3 increase this improvement percentage to 43.51% and 47.17%, respectively. Detailed results for the remaining months can be provided upon request from the authors. The main improvement comes with LNS1, which improves the LNS0 solution by 43.07% on average. LNS2 and LNS3 increase this improvement percentage to 43.14% and 47.01% respectively. Table 4 gives a summary of the average and maximum improvement from LNS0 solution under each method for each month.

It can be observed that the results of LNS0 for weight schemes with identical ratios

Table 2: The results obtained using LNS0 for Month 1

Weights	DH	IL	DL	Obj.
(200, 100, 1)	347	871	1013	157513
(200, 50, 1)	308	967.5	990	110965
(200, 10, 1)	321	1093	957	76087
(200, 5, 1)	320	1159	994	70789
(100, 100, 1)	429	602.5	1034	104184
(100, 50, 1)	347	871	1013	79263
(100, 10, 1)	319	1105.5	969	43924
(100, 5, 1)	320	1159	994	38789
(50, 100, 1)	485	465	1024	71774
(50, 50, 1)	429	602.5	1034	52609
(50, 10, 1)	313	1037.5	975	27000
(50, 5, 1)	319	1119.5	978	22525.5
(10, 100, 1)	660	464.5	1069	54119
(10, 50, 1)	576	456.5	1038	29623
(10, 10, 1)	429	602.5	1034	11349
(10, 5, 1)	352	850	981	8751

Table 3: The results obtained using LNS1, LNS2, and LNS3 algorithms for Month 1

	LNS1					LNS2					LNS3				
Weights	**DH**	**IL**	**DL**	**Obj.**	**Impr.**	**DH**	**IL**	**DL**	**Obj.**	**Impr.**	**DH**	**IL**	**DL**	**Obj.**	**Impr.**
(200, 100, 1)	144	492.5	725	78775	49.99	143	491.5	790	78540	50.14	125	497	724	75424	52.12
(200, 50, 1)	133	516	742	53142	52.11	133	516	725	53125	52.12	119	511	724	50074	54.87
(200, 10, 1)	129	524	688	31728	58.30	127	524	758	31398	58.73	119	511	724	29634	61.05
(200, 5, 1)	130	532	817	29477	58.36	130	532	806	29448	58.40	119	511	724	27061	61.77
(100, 100, 1)	225	442	735	67435	35.27	226	440	737	67337	35.37	158	443.5	710	60860	41.58
(100, 50, 1)	144	490	655	39555	50.10	142	493	652	39502	50.16	125	497	724	38074	51.96
(100, 10, 1)	128	521	629	18639	57.57	126	520	679	18479	57.93	119	511	724	17734	59.63
(100, 5, 1)	131	532	677	16437	57.62	131	532	677	16419	57.67	119	511	724	15161	60.91
(50, 100, 1)	253	424	732	55782	22.28	251	424	766	55716	22.37	162	439.5	710	52760	26.49
(50, 50, 1)	224	444	814	34214	34.97	224	444	804	34204	34.98	158	443.5	710	30785	41.48
(50, 10, 1)	130	523	800	12530	53.59	130	523	799	12529	53.60	119	511	724	11784	56.36
(50, 5, 1)	128	522	768	9778	56.59	128	522	760	9752	56.71	119	511	724	9211	59.11
(10, 100, 1)	422	438.5	690	48760	9.90	422	438.5	690	48760	9.90	248	416.5	679	44809	17.20
(10, 50, 1)	345	424	660	25310	14.56	341	424	675	25285	14.64	228	420.5	647	23952	19.14
(10, 10, 1)	234	437	634	7344	35.29	234	437	634	7344	35.29	168	443.5	566	6681	41.13
(10, 5, 1)	160	489.5	511	4558.5	47.91	158	493.5	511	4540	48.12	137	497	546	4383	49.91
				Ave	**43.40**				**Ave**	**43.51**				**Ave**	**47.17**
				Max	**58.36**				**Max**	**58.73**				**Max**	**61.77**

of deadheading cost and layover cost (e.g. (100,100,1) and (50,50,1) and (10,10,1)) are identical. Furthermore, the results of LNS1 for these instances also show a significant amount of similarity. We attribute this to the fact that the cost of domestic layovers is dominated by the cost of deadheadings and international layovers. Hence, the ratio of deadheading cost and layover cost becomes the determining factor for the results of LNS0 and LNS1.

Table 4: Summary of Improvement Levels

	LNS1		**LNS2**		**LNS3**	
	Avg	Max	Avg	Max	Avg	Max
Month 1	43.40%	58.36%	43.51%	58.73%	47.17%	61.77%
Month 2	42.96%	57.03%	43.02%	57.03%	47.14%	60.03%
Month 3	42.76%	54.13%	42.84%	54.22%	46.84%	58.37%
Month 4	43.15%	55.06%	43.18%	55.06%	46.87%	58.39%

We present Figure 2 to depict the improvement in objective functions for LNS0 and LNS1. LNS0 initially makes quick but small improvements followed by small improvements after long intervals of iterations. On the other hand LNS1 achieves a sharp reduction in the objective function after the solution of the first SPP and then starting from this solution continues to improve gradually. Note that within our experimental setup, the end of the first SPP step roughly corresponds to 5% of the overall CPU time. Hence, LNS1 can be used with $k_{max} = 25$ to obtain a high quality solution in a short time.

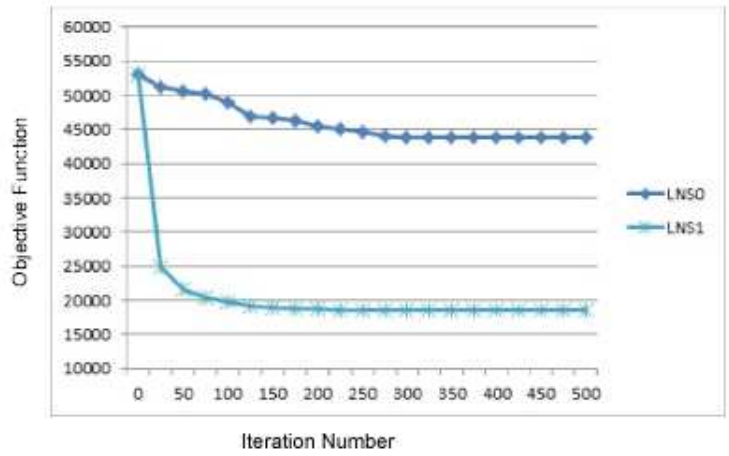

Figure 2: Objective function trajectory under LNS0 and LNS1, Month 1

5.3 Running Times of the Algorithms

Table 5 and 6 summarize the CPU times required to get the reported solution of our algorithms. Less than 1 minute of this time is used to generate duties. The CPU times of LNS1 are almost twice those of LNS0, nevertheless, it yields solutions within a reasonable time. The running times of LNS2 are very similar to those of LNS1, and differ only by the solution of an extra SPP, which takes about 5 minutes. Since LNS3 uses outputs from multiple streams, the total running time should include the

running times of all the streams, which corresponds to approximately 25 times the time requirement of LNS1.

It can be observed that the running times for the weight schemes (10, 100, 1) and (10, 50, 1) are significantly larger than the others, for both LNS0 and LNS1. This phenomenon can be explained by the importance of deadheadings, which is understated by these weight schemes. Cheaper deadheadings result in longer duties with more deadheadings, which in turn limit the search space for RMMC, requiring a longer search for feasible solutions. In each iteration, the RMMC is run for one fifth of the legs on the average, and the longer search times for RMMC are reflected to the overall runtime with a factor of 5000.

Table 5: CPU Times of LNS0 Algorithm (seconds)

Weights	**Month 1**	**Month 2**	**Month 3**	**Month 4**
(200, 100, 1)	1697.5	1906.1	1990.8	2054.1
(200, 50, 1)	1568.8	1561.7	1558.4	1789.7
(200, 10, 1)	1555.0	1533.0	1490.5	1661.5
(200, 5, 1)	1701.4	1589.8	1655.6	1728.4
(100, 100, 1)	2500.9	2497.4	2557.5	2747.8
(100, 50, 1)	1699.3	1910.8	1989.4	2052.2
(100, 10, 1)	1552.6	1528.7	1517.9	1676.5
(100, 5, 1)	1685.7	1592.2	1654.7	1740.2
(50, 100, 1)	3976.9	4156.6	4306.7	4674.3
(50, 50, 1)	2465.7	2504.9	2557.4	2732.0
(50, 10, 1)	1611.4	1544.7	1567.9	1856.4
(50, 5, 1)	1683.6	1582.3	1651.4	1731.7
(10, 100, 1)	14601.6	14255.3	15732.4	16295.9
(10, 50, 1)	12574.5	12852.7	13580.1	14876
(10, 10, 1)	2472.9	2456.6	2578.3	2616.3
(10, 5, 1)	1786.7	1974.4	1949.3	2168.3
Ave	3445.9	3465.5	3646.1	3900.1
Max	14601.6	14255.3	15732.4	16295.9

Table 6: CPU Times of LNS1 Algorithm (seconds)

Weights	**Month 1**	**Month 2**	**Month 3**	**Month 4**
(200, 100, 1)	4053.2	4112.8	4548.7	4630.4
(200, 50, 1)	3325.5	3252.1	3484.0	3803.9
(200, 10, 1)	3160.1	3199.9	3562.3	3716.7
(200, 5, 1)	3322.4	3198.2	3420.1	3715.3
(100, 100, 1)	5276.0	5334.7	5905.0	5906.8
(100, 50, 1)	4042.5	4176.4	4363.1	4670.9
(100, 10, 1)	3227.3	3256.7	3499.4	3775.1
(100, 5, 1)	3308.0	3218.3	3414.2	3719.2
(50, 100, 1)	9553.0	9461.9	10105.1	10778.6
(50, 50, 1)	5055.1	5381.8	5855.8	6040.3
(50, 10, 1)	3254.1	3322.0	3478.2	3787.7
(50, 5, 1)	3236.4	3309.5	3497.6	3705.1
(10, 100, 1)	48508.8	49643.8	51927.9	54469.8
(10, 50, 1)	41340.7	41545.4	44873	46569.1
(10, 10, 1)	5212.9	5377.3	5795.6	6111.3
(10, 5, 1)	4082.8	3881.6	4504.6	4687.8
Ave	9372.4	9479.5	10139.7	10630.5
Max	48508.8	49643.8	51927.9	54469.8

5.4 Robustness analysis

We now analyze the robustness of the algorithms we have presented. For the sake of simplicity, we focus on LNS1, which dominates LNS0, and forms the basis of LNS2 and LNS3. We have also chosen Month 3 as a representative month, based on the number of legs scheduled. We have performed ten runs of LNS1 for the flight schedule of Month 3 and every weight combination, each run with a different random number seed, the results of which we present in Table 7. In addition to the minimum, average, and maximum results, we also report the standard deviation of the results, and the coefficient of variation (average / standard deviation). The results clearly show that the algorithm returns with an insignificant deviation, regardless of the weight combination.

Table 7: Robustness analysis for LNS1

Weights	Min	Average	Max	Stdev	C.V.
(200, 100, 1)	85247.0	85941.0	86917.0	537.0	0.62%
(200, 50, 1)	59390.0	60309.6	61083.0	561.1	0.93%
(200, 10, 1)	37084.0	38109.9	39423.0	747.8	1.96%
(200, 5, 1)	34511.5	35462.6	36605.5	756.9	2.13%
(100, 100, 1)	70286.0	71033.4	71851.0	490.9	0.69%
(100, 50, 1)	42946.0	43214.3	43645.0	243.9	0.56%
(100, 10, 1)	21554.0	22203.8	22814.0	359.0	1.62%
(100, 5, 1)	18861.0	19216.8	19638.0	280.1	1.46%
(50, 100, 1)	58532.0	58889.4	59192.0	214.5	0.36%
(50, 50, 1)	35608.0	35898.6	36496.0	288.3	0.80%
(50, 10, 1)	14226.0	14342.0	14458.0	74.5	0.52%
(50, 5, 1)	10967.0	11420.6	11695.5	243.3	2.13%
(10, 100, 1)	50208.0	50267.0	50487.0	79.9	0.16%
(10, 50, 1)	26089.0	26343.8	26503.0	152.3	0.58%
(10, 10, 1)	7602.0	7681.9	7771.0	56.3	0.73%
(10, 5, 1)	4828.0	4843.9	4860.5	13.0	0.27%

6 Conclusion

The CPP is a challenging optimization problem, which has been studied extensively by the operations research community. In this paper, we have studied the CPP using the flight schedule of a European airline company, which has unique characteristics. We have set our objective to minimize crew based costs due to deadheadings, international layovers, and domestic layovers. We have attempted to solve the monthly problem of considerable size (around 27,000 flight legs), and proposed an optimization-driven heuristic algorithm that combines heuristic and exact approaches: large-scale neighborhood search, exact enumerative search, and integer programming. We presented the results of a computational study that provides evidence that the proposed approach delivers multiple high-quality solutions within a relatively short CPU time.

An interesting issue that is worthy of future research is to implement a similar approach for solving the so-called airline crew rostering problem. This important airline operational problem is usually solved after the CPP. It aims at assigning crew pairings to anonymous bidlines and, in a final step, assigning these bidlines to individual crew members. In some sense, this latter problem is conceptually similar to the CPP since it involves building feasible sequences of pairings (instead of duties) to derive effective rosters/bidlines (instead of pairings). We believe that the main ideas that were developed in this paper could be advantageously adapted for solving this crew rostering problem. Furthermore, the ability of solving large-scale CPP instances within

a reasonable time is very important as it opens the doors to efficiently solve more complicated integrated problems of the airline industry, such as the *Integrated Fleet Assignment and Crew Pairing Problem*. The ability to jointly solve these integrated problems is very important as they provide means to optimize airline operations as a whole and avoid suboptimality (or, infeasibility) that arises from solving airline planning problems in subsequent steps. We believe that our LNS approach could be used as a subroutine in solving integrated airline problems.

Acknowledgements. This project is funded in part by TUBITAK grant 110M308. We thank the referees for their valuable comments, which have improved the paper substantially.

References

R. Anbil, E. Gelman, B. Patty, and R. Tanga. Recent advances in crew-pairing optimization at american airlines. *Interfaces*, 21:62–74, 1991.

E. Andersson, E. Housos, N. Kohl, and D. Wedelin. Crew pairing optimization. In G. Yu, editor, *Operations Research in the Airline Industry*, chapter 8, pages 228–258. Kluwer Academic Publishers, Norwell, 1998.

J.P. Arabeyre, J. Fearnley, F.C. Steiger, and W. Teather. The airline crew scheduling problem: A survey. *Transportation Science*, 3(2):140–163, 1969. doi: 10.1287/trsc.3.2.140. URL `http://transci.journal.informs.org/content/3/2/140.abstract`.

A. Aydemir-Karadag, B. Dengiz, and A. Bolat. Crew pairing optimization based on hybrid approaches. *Computers & Industrial Engineering*, 65:87 – 96, 2013.

A. Azadeh, M. H. Farahani, H. Eivazy, S. Nazari-Shirkouhi, and G. Asadipour. A hybrid meta-heuristic algorithm for optimization of crew scheduling. *Applied Soft Computing*, 13:158 – 164, 2013.

C. Barnhart and R.G. Shenoi. An approximate model and solution approach for the long-haul crew pairing problem. *Transportation Science*, 32(3):221–231, 1998. doi: 10.1287/trsc.32.3.221. URL `http://transci.journal.informs.org/content/32/3/221.abstract`.

C. Barnhart, E. Johnson, R. Anbil, and L. Hatay. A column generation technique for the long-haul crew assignment problem. In T. Ciriano and R. Leachman, editors, *Optimization in Industry: Volume "II"*. John Wiley and Sons, U.K., 7–22 edition, 1994.

C. Barnhart, F. Lu, and R. Shenoi. Integrated airline schedule planning. In G.Yu, editor, *Kluwer Academic Publishers*, pages 384–403. Kluwer Academic Publishers, Boston,MA, 1998.

C. Barnhart, P. Belobaba, and A.R. Odoni. Applications of operations research in the air transport industry. *Transportation Science*, 37(4):368–391, 2003a. doi: 10.1287/trsc.37.4.368.23276. URL `http://transci.journal.informs.org/content/37/4/368.abstract`.

C. Barnhart, A.M. Cohn, E.J. Johnson, D. Klabjan, G.L. Nemhauser, and P.H. Vance. Airline crew scheduling. In R. W. Hall, editor, *Handbook of Transportation Science*, chapter 14, pages 517–560. Kluwer Academic Publishers, Norwell, 2 edition, 2003b.

E.R. Butchers, P.R. Day, A.P. Goldie, S. Miller, J.A. Meyer, D.M. Ryan, A.C. Scott, and C.A. Wallace. Optimized crew scheduling at air new zealand. *Interfaces*, 31(1):30–56, 2001. doi: 10.1287/inte.31.1.30.9688. URL `http://interfaces.journal.informs.org/content/31/1/30.abstract`.

L. Cavique, C. Rego, and I. Themido. Subgraph ejection chains and tabu search for the crew scheduling problem. *Journal of the Operational Research Society*, 50 (6):608 – 616, 1999.

J. Clausen, A. Larsen, J. Larsen, and N. J. Rezanova. Disruption management in the airline industryconcepts, models and methods. *Computers & Operations Research*, 37(5):809 – 821, 2010. ISSN 0305-0548. doi: 10.1016/j.cor.2009.03.027. URL http://www.sciencedirect.com/science/article/pii/S0305054809000914. Disruption Management.

G.-F. Deng and W.-T. Lin. Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications*, 38(5): 5787 – 5793, 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.10.053. URL http://www.sciencedirect.com/science/article/pii/S0957417410011978.

G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon, and F. Soumis. Crew pairing at air france. *European Journal of Operational Research*, 97(2):245 – 259, 1997. ISSN 0377-2217. doi: 10.1016/S0377-2217(96)00195-6. URL http://www.sciencedirect.com/science/article/pii/S0377221796001956.

G. Desaulniers, J. Desrosiers, M. Gamache, and F. Soumis. Crew scheduling in air transportation. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*. Kluwer Academic Publishers, Norwell, 1998.

V. Dück, F. Wesselmann, and L. Suhl. Implementing a branch and price and cut method for the airline crew pairing optimization problem. *Public Transport*, 3:43–64, 2011. ISSN 1866-749X. URL http://dx.doi.org/10.1007/s12469-011-0038-9. 10.1007/s12469-011-0038-9.

T. Emden-Weinert and M. Proksch. Best practice simulated annealing for the airline crew scheduling problem. *Journal of Heuristics*, 5:419–436, 1999. ISSN 1381-1231. URL http://dx.doi.org/10.1023/A:1009632422509. 10.1023/A:1009632422509.

C. Gao, E. Johnson, and B. Smith. Integrated airline fleet and crew robust planning. *Transportation Science*, 43(1):2–16, 2009. doi: 10.1287/trsc.1080.0257. URL http://transci.journal.informs.org/content/43/1/2.abstract.

I. Gershkoff. Optimizing Flight Crew Schedules. *Interfaces*, 19:29–43, 1989. doi: 10.1287/inte.19.4.29.

B. Gopalakrishnan and E. Johnson. Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, 140:305–337, 2005.

K.L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993. doi: 10.1287/mnsc.39.6.657. URL http://mansci.journal.informs.org/content/39/6/657.abstract.

D. Levine. Application of a hybrid genetic algorithm to airline crew scheduling. *Computers & Operations Research*, 23(6):547 – 558, 1996. ISSN 0305-0548. doi: 10.1016/0305-0548(95)00060-7. URL http://www.sciencedirect.com/science/article/pii/0305054895000607.

G. Froyland M. Dunbar and and C-L Wu. An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Computers & Operations Research*, 45:68 – 86, 2014.

A. Makri and D. Klabjan. A new pricing scheme for airline crew scheduling. *INFORMS Journal on Computing*, 16(1):56–67, 2004. URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1020.0026.

I. Muter, S.I. Birbil, K. Bülbül, G. Sahin, H. Yenigün, D. Taş, and D. Tüzünç. Solving a robust airline crew pairing problem with column generation. *Computers & Operations Research*, 40:815 – 830, 2013.

H.T. Ozdemir and C.K. Mohan. Flight graph based genetic algorithm for crew scheduling in airlines. *Information Sciences*, 133(34):165 – 173, 2001. ISSN 0020-0255. doi: 10.1016/S0020-0255(01)00083-4. URL http://www.sciencedirect.com/science/article/pii/S0020025501000834. Evolutionary Algorithms.

D.M. Ryan. The solution of massive generalized set partitioning problems in aircrew rostering. *The Journal of the Operational Research Society*, 43(5):pp. 459–467, 1992. ISSN 01605682. URL http://www.jstor.org/stable/2583565.

M. Saddoune, G. Desaulniers, I. Elhallaou, and F. Soumis. Integrated airline crew scheduling: A bi-dynamic constraint aggregation method using neighborhoods. *European Journal of Operational Research*, 212:445 – 454, 2011.

M. Saddoune, G. Desaulniers, I. Elhallaou, and F. Soumis. Integrated airline crew pairing and crew assignment by dynamic constraint aggregation. *Transportation Science*, 46(1):39 – 55, 2012.

H. D. Sherali, K.-H. Bae, and M. Haouari. An integrated approach for airline flight selection and timing, fleet assignment, and aircraft routing. *Transportation Science*, 47(4):455 – 476, 2013.

H.D. Sherali, K.-H. Bae, and M. Haouari. Integrated airline schedule design and fleet assignment: Polyhedral analysis and benders' decomposition approach. *INFORMS Journal on Computing*, 22(4):500–513, 2010. URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1090.0368.

S. Subramanian and H.D. Sherali. An effective deflected subgradient optimization scheme for implementing column generation for large-scale airline crew scheduling problems. *INFORMS Journal on Computing*, 20(4):565–578, 2008. URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1080.0267.

H. Tekiner, S.I. Birbil, and K. Bülbül. Robust crew pairing for managing extra flights. *Computers & Operations Research*, 36(6):2031 – 2048, 2009. ISSN 0305-0548. doi: 10.1016/j.cor.2008.07.005. URL http://www.sciencedirect.com/science/article/pii/S030505480800124X.

P.H. Vance. *Crew Scheduling, Cutting Stock, and Column Generation: Solving Huge Integer Programs*. PhD thesis, Ph.D.Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA., 1993.

P.H. Vance, C. Barnhart, E.L. Johnson, and G.L. Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45:183–200, 1997.

S. Yan and Y.-P. Tu. A network model for airline cabin crew scheduling. *European Journal of Operational Research*, 140(3):531 – 540, 2002. ISSN 0377-2217. doi: 10.1016/S0377-2217(01)00215-6. URL http://www.sciencedirect.com/science/article/pii/S0377221701002156.