SABANCI UNIVERSITY
Orhanlı-Tuzla, 34956 Istanbul, Turkey
Phone: +90 (216) 483-9500
Fax: +90 (216) 483-9550
http://www.sabanciuniv.edu

algopt
algorithms & optimization
http://algopt.sabanciuniv.edu/projects
December 5, 2010

Solving A Robust Airline Crew Pairing Problem With Column Generation

İbrahim Muter, Ş. İlker Birbil, Kerem Bülbül, Güvenç Şahin, Hüsnü Yenigün
Sabancı University, Manufacturing Systems and Industrial Engineering Program, Orhanlı-Tuzla, 34956 Istanbul, Turkey

Duygu Taş
Technische Universiteit Eindhoven, Postbus 513, 5600 MB Eindhoven, The Netherlands

Dilek Tüzün
Yeditepe University, System Engineering Department, Kayışdağı-Kadıköy, 34755 Istanbul, Turkey

Abstract: In this study, we solve a robust version of the airline crew pairing problem. Our concept of robustness was partially shaped during our discussions with small local airlines in Turkey which may have to add a set of extra flights into their schedule at short notice during operation. Thus, robustness in this case is related to the ability of accommodating these extra flights at the time of operation by disrupting the original plans as minimally as possible. We focus on the crew pairing aspect of robustness and prescribe that the planned crew pairings incorporate a number of predefined recovery solutions for each potential extra flight. These solutions are implemented only if necessary for recovery purposes and involve either inserting an extra flight into an existing pairing or partially swapping the flights in two existing pairings in order to cover an extra flight. The resulting mathematical programming model follows the conventional set covering formulation of the airline crew pairing problem typically solved by column generation with an additional complication. The model includes constraints that depend on the columns due to the robustness consideration and grows not only column-wise but also row-wise as new columns are generated. To solve this difficult model, we propose a row and column generation approach. This approach requires a set of modifications to the multi-label shortest path problem for pricing out new columns (pairings) and various mechanisms to handle the simultaneous increase in the number of rows and columns in the restricted master problem during column generation. We conduct computational experiments on a set of real instances compiled from local airlines in Turkey.

Keywords: Airline crew scheduling; robust crew pairing; row and column generation; multi-label shortest path.

1. Introduction.

Airline planning and operations problems have formed a particular area of interest in operations research in the last fifty years. Several problems have been posed by both practitioners and researchers which include network planning and design, revenue and yield management, flight scheduling and timetabling, fleet and aircraft assignment, maintenance routing, and crew scheduling. Airlines rely on implementable solutions to these problems to bring down their operational costs and increase their market shares and revenues. Among the airline planning problems, the crew scheduling problem holds a prominent position, and it is one of the most widely studied problems for the following reasons: From the practitioners' point of view, crew costs are the second major cost component after the fuel costs, and various regulations and practical considerations have to be integrated into the planning process. From the researchers' point of view, such regulations and considerations make it computationally hard to even identify a single feasible schedule, let alone an optimal one under complicated cost structures and involved feasibility rules (see [2, 12] for excellent reviews on airline crew scheduling).

Due to its complexity, the crew scheduling problem is tackled in two sequential phases both in practice and in the literature. In the crew pairing problem, each crew is assigned to a sequence of flight legs so that each flight in the schedule is covered and the total cost is minimized. In this problem, a crew is regarded as an entity, but the composition of the crew and the identities of the crew members are not considered. The second phase of the crew scheduling problem is known as crew assignment and requires the solution of the crew pairing problem as an input. It yields a schedule for each crew member by assigning the crew members to the previously constructed pairings in order to obtain crew rosters. The crew pairing and assignment problems are at the interface of tactical and operational level planning. Both the pairings and the rosters are determined and published monthly and are subject to updates during execution due to operational contingencies [23].

A nonstop flight is called a *flight leg*. Two flight legs can be operated by the same crew if the arrival station of the first flight leg is the same as the departure station of the other one, and the time between these two flights is adequate to satisfy the crew feasibility rules. Thus, a sequence of flight legs forms a *duty period* as long as the idle time between two consecutive flights in a duty period is not shorter than the *minimum sit time* and does not exceed the *maximum sit time*. Furthermore, the total elapsed time of a duty period is required to be less than or equal to the *maximum elapsed time*, and the number of flights in a duty period is restricted by the *maximum number of flights*. The flights in a duty period are operated by a single crew, and a consecutive sequence of no more than the *maximum number of duty periods* is referred to as a *pairing*, only if the first duty period starts and the last duty period ends at the same station, which is a designated *crew base* location. Moreover, the time between two successive duty periods in a pairing is restricted by the *minimum rest time* and the *maximum rest time*, and the total elapsed time of a pairing cannot exceed the *maximum time away from base*. Finally, a set of feasible pairings constitutes a feasible pairing solution, if each flight in the schedule is covered by at least one pairing. In case a flight appears in several pairings, one crew operates the flight while the other crews are transferred between two stations for repositioning purposes. This is also known as *deadheading* and is also commonly used to bring crews home at the end of a pairing or to relocate crew members at the beginning of a pairing to cover a flight departing from a non-base station. The reader is referred to [2, 12] for detailed descriptions of the terminology and a discussion of the common cost structures and feasibility rules in crew scheduling. In this study, all of the feasibility rules above are implemented except those that restrict the length of a pairing for two reasons: First, the pairing cost structure is independent from the maximum time away from base and the number of duties in the pairing for the instances used in this work (see Section 3.1). Second, in the computational study in Section 4, we observe that the pairing solutions are composed of a set of short pairings.

The objective of the crew pairing problem (CPP) is to find the least costly set of feasible pairings that cover all flights given in the flight schedule. Formally, by using a set-covering type formulation, CPP is modeled as

$$\begin{array}{lll} \text{minimize} & \sum_{p\in\mathcal{P}} c_p y_p & \\ \text{subject to} & \sum_{p\in\mathcal{P}} a_{ip} y_p \geq 1, & i \in \mathcal{F}, \\ & y_p \in \{0,1\}, & p \in \mathcal{P}, \end{array} \tag{1}$$

where $\mathcal{P}$ is the set of all feasible pairings and $\mathcal{F}$ is the set of all flights. Here, c_p is the cost of pairing p; $a_{ip} = 1$, if flight i is covered by pairing p and 0, otherwise. The decision variable $y_p = 1$, if pairing p is selected by the solution and 0, otherwise. The objective function minimizes the total cost of the selected pairings, while the set of constraints ensures that each flight leg is covered by at least one pairing. The number of feasible pairings (variables) in problem (1) is very large in practice. Therefore, a column generation method is typically employed to solve the linear programming (LP) relaxation of problem (1) [7, 27, 1]. The interested reader is referred to [2, 12] for an overview and comparison of the various solution methods applied to the CPP.

Airline operations are subject to several different types of disruptions, such as; unfavorable weather conditions, aircraft breakdowns, airport congestion, and so on. Disruptions require prompt recovery actions and may lead to flight delays and cancellations, updated passenger, crew and aircraft routes, and fleet changes (see Clausen *et al.* [6] for a recent review of disruption management in the airline industry, and Lettovský and Johnson [15] for an example of recovery models used in crew scheduling). On one hand, the brand image of an airline and the quality of service to the passengers depend much on how quickly and successfully the airline responds to unexpected events. On the other hand, irregular operations are a major source of difference between planned and actual costs and the trade-off between customer satisfaction and cost effectiveness must be handled carefully while executing recovery procedures. Thus, *robust planning* has been considered by some airlines as an effective means of managing disruptions in their plans, while keeping the gap between the actual and planned costs at a minimum. One way of achieving robustness is to explicitly consider and incorporate disruption scenarios at the time of planning. This approach provides two significant advantages: First, if a disruption scenario, which was taken into account at the planning stage, is in fact realized at the time of operation, then the path to recovery is well-defined. Second, the financial impact of recovery is visible to planners that leads to better decisions in the first place. In this work, we concentrate on a robust crew pairing problem when the source of disruptions is a set of extra flights, which are added to the flight schedule after the monthly crew rosters are published. In particular, we demonstrate that promoting pairings with special properties in the optimal crew pairing solution provides us with recourse actions, when one or several new flights are inserted into the flight schedule. In this sense, our work is general. For other types of disruptions, similar robust models may be formulated as long as it is possible to define the desirable properties of pairings, which yield natural paths to recovery. We also note that our work fits into a stream of research on robust airline planning that has been receiving increasingly more attention in recent years. For instance, various robustness issues and models in crew scheduling are considered in [10, 30, 20, 21]. Research on robust fleet assignment models include [18, 22]. In other studies, Weide *et al.* [29] deal with the aircraft routing and crew scheduling decisions in an integrated manner in order to achieve more robust aircraft and crew schedules, and Lan *et al.* [14] demonstrate that airlines can reduce the impact of delays on passengers by routing aircraft more effectively and re-timing flight departure times.

This paper discusses how an airline may hedge against a certain type of operational disruption by incorporating robustness into the pairings generated at the planning level. In particular, we address how a set of extra flights may be added into the flight schedule at the time of operation by modifying the pairings at hand, and without delaying or canceling the existing flights in the schedule. We assume that the set of potential extra flights and their associated departure time windows are known at the planning stage. We note that this study was partially motivated during our interactions with the local airlines in Turkey that sometimes have to add extra flights to their schedule. These airlines can typically estimate the potential time windows of the extra flights based on their past experiences but prefer to ignore this information during planning because these flights may not need to be actually operated. Typically, these extra flights are then handled by recovery procedures at the time of operation, which may lead to substantial deviations from the planned crew pairings and costs. Furthermore, this problem is also relevant to any type of airline due to crew licensing regulations. Typically, cockpit crews are licensed to operate a single aircraft type, while cabin crews may serve on two or three aircraft types. Thus, the crew pairing problem is solved separately for each fleet type. However, the aircraft type assigned to a flight may later be modified due to maintenance requirements or lower/higher than anticipated demand. This in turn implies that a flight is deleted from one fleet schedule and added to another one. Note that there is a simple recourse action for the canceled flight in this case; the crew members of the canceled flight may be repositioned on the flight added to the schedule of the updated fleet type. On the other hand, the insertion of a new flight is much more complicated because low-cost pairings often call for short connection times with little slack. The reader is referred to [23] for an in-depth discussion of the conceptual framework of this problem, which we refer to as the Robust Crew Pairing for Managing Extra Flights (RCPEF).

Our main contribution in this study is a column generation algorithm to solve RCPEF which poses some unique challenges as we explain in Section 3. In [23], the authors introduce how an extra flight may be accommodated by modifying the existing pairings and formulate a set of integer programming (IP) models that provide natural recovery options without disrupting the existing flights. Their approach requires that all possible crew pairings are available and hence, it is only viable for relatively small problem instances. The proposed models in that work are solved by standard commercial solvers. In other words, in [23] the focus is on the modeling aspects of RCPEF, while in this paper we develop a column generation algorithm that can handle larger problem instances. Moreover, in the robust models given in [23], the objective is to maximize the number of these desirable pairings while keeping the total crew pairing cost slightly larger than the baseline cost. In contrast, in this paper we tackle the entire problem in one step. That is, we minimize the total cost of the pairing solution while requiring the presence of a predefined number of recovery options for each potential extra flight. We point out further specific differences between these two studies in the next section. We also note that we only solve the problem for the given daily/weekly flight schedule and do not consider the issue of repeating schedules.

2. Robust Crew Pairing Model. In this section, we first introduce the types of pairings that yield natural paths to recovery for potential extra flights and then formulate RCPEF as an integer program with exponentially many variables. In [23], several recovery options are explored for managing potential extra flights. The authors classify these recovery options into two types:

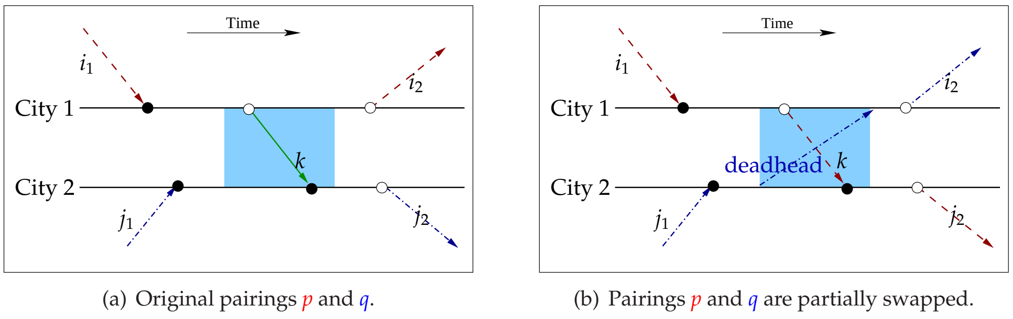
- **Type A.** Two pairings are selected and (partially) swapped to cover an extra flight.
- **Type B.** One pairing with sufficient connection time between two consecutive legs is modified

to cover an extra flight.

Here, we restrict ourselves to type A and type B solutions as illustrated in Figure 1 and Figure 2, where the estimated time window of extra flight k is depicted by the shaded rectangles. The left and right end points of the estimated time window mark the earliest possible departure and latest possible arrival times of extra flight k, respectively. These figures are typical examples of a time-space network representation, where the departure and arrival cities are shown by horizontal lines and the time dimension is considered horizontally from left to right. In the airline crew pairing literature, this network representation is referred to as a flight network, where the arcs correspond to flights and connections and the nodes are the departure and arrival stations [25].

In Figure 1, the original pairing p covering the flight legs i_1 and i_2 is modified so that the corresponding crew operates extra flight k after flight i_1, and subsequently follows the route originally assigned to pairing q from flight j_2 onward. In addition, the crew originally covering flights j_1 and j_2 is repositioned to the departure airport of flight i_2 after flight j_1 in order to complete the route originally assigned to pairing p from flight i_2 onward. Note that this recovery option is only available if the pairings p and q are feasible both before and after swapping. In such a case, the pairing pair (p, q) is said to provide a type A solution for the extra flight k. The interested reader is referred to [23] for a complete description of the required feasibility conditions (see solution A.1 in that paper). Type B solutions depicted in Figure 2 involve a single pairing with ample connection time between two consecutive flight legs i_1 and i_2. In this case, extra flight k is inserted between i_1 and i_2, and the crew is repositioned before or after flight k as necessary. These recovery options may be employed only if pairing p is feasible both with and without extra flight k (see solutions B.1 and B.2 in [23] for the required feasibility conditions), and in this case p is said to provide a type B solution for the extra flight k.

We note that six type A solutions and three type B solutions are taken into account in [23]. In this paper, five of these type A solutions and one type B solution have been eliminated from consideration for two reasons: First, three of the removed type A solutions and the removed type B solution introduce two deadheads just to cover one extra flight. Our recent discussions with industry experts indicate that this is not desirable. Second, it is extremely difficult to keep track of the other two type A solutions in the pricing subproblem in the column generation algorithm.



(a) Original pairings p and q. (b) Pairings p and q are partially swapped.

Figure 1: Two pairings p and q are swapped to cover extra flight k (type A solution).

The proposed robust mathematical model for RCPEF is given below:

$$\text{minimize} \quad \sum_{p\in\mathcal{P}} c_p y_p + \sum_{i\in\mathcal{F}} d_i s_i^+ \tag{2}$$

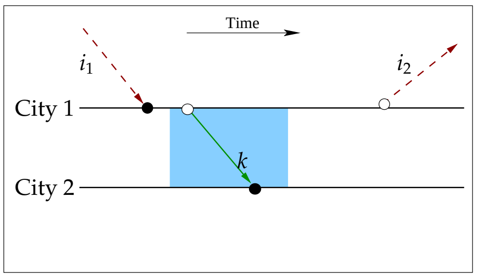
$$\text{subject to} \quad \sum_{p\in\mathcal{P}} a_{ip} y_p - s_i^+ = 1, \qquad i \in \mathcal{F}, \tag{3}$$

$$\sum_{(p,q)\in\mathcal{P}_A(k)} x^k_{(p,q)} \geq \alpha_k, \qquad k \in \mathcal{K}, \tag{4}$$

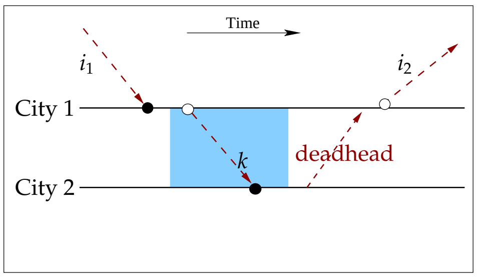
$$\sum_{p\in\mathcal{P}_B(k)} y_p \geq \beta_k, \qquad k \in \mathcal{K}, \tag{5}$$

$$y_p + y_q \leq 1 + x^k_{(p,q)}, \qquad (p,q) \in \mathcal{P}_A(k), k \in \mathcal{K}, \tag{6}$$

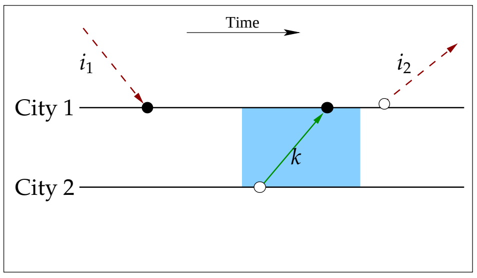
$$y_p \geq x^k_{(p,q)}, \qquad p \in \mathcal{P} : (p,q) \in \mathcal{P}_A(k), k \in \mathcal{K}, \tag{7}$$



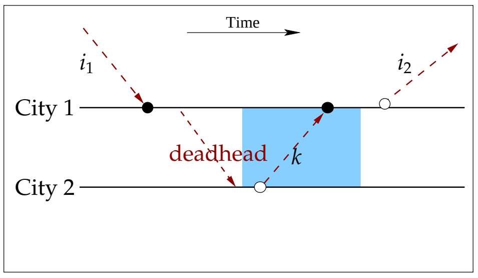
(a) Original pairing p.



(b) Pairing p is modified to cover extra flight k.



(c) Original pairing p.



(d) Pairing p is modified to cover extra flight k.

Figure 2: Extra flight k is inserted into pairing p (type B solution).

$$y_q \geq x^k_{(p,q)}, \qquad q \in \mathcal{P} : (p,q) \in \mathcal{P}_A(k), k \in \mathcal{K}, \tag{8}$$
$$y_p \in \{0,1\}, \qquad p \in \mathcal{P}, \tag{9}$$
$$x^k_{(p,q)} \in \{0,1\}, \qquad (p,q) \in \mathcal{P}_A(k), k \in \mathcal{K}, \tag{10}$$
$$s^+_i \in \mathbb{Z}_+, \qquad i \in \mathcal{F}. \tag{11}$$

In this formulation, $\mathcal{F}$, $\mathcal{K}$, and $\mathcal{P}$ denote the set of all flights, the set of all extra flights, and the set of all feasible pairings, respectively. The index set $\mathcal{P}_A(k)$ denotes the set of pairing pairs that provide a type A solution for extra flight k. Similarly, $\mathcal{P}_B(k)$ is the set of pairings that provide a type B solution for extra flight k. The parameter $a_{ip} = 1$, if flight i is included in pairing p and 0, otherwise. The cost of pairing p is represented by c_p. The parameters α_k and β_k denote the minimum number of type A and type B solutions that are required in the pairing solution, respectively. If pairing p is selected, the decision variable y_p takes the value 1 and 0, otherwise. Furthermore, the auxiliary binary variable $x^k_{(p,q)}$ is set to 1, if the pairing pair $(p,q) \in \mathcal{P}_A(k)$ and both pairings p and q are present in the pairing solution and to 0, otherwise.

The set of constraints (3) are the standard set covering constraints for the regular flights. The integer surplus variables s^+_i, $i \in \mathcal{F}$, in this set of constraints indicate the number of times non-operating crews are repositioned (deadheaded) on flight $i \in \mathcal{F}$. Then, the objective (2) minimizes the total cost of the selected pairings and the cost of deadhead flights, where d_i represents the cost of a single deadhead use of flight i. In Section 3.1, we allow the minimum sit and rest times for deadhead flights to be shorter than those for regular flights. However, note that a flight in a pairing constructed according to the regular rules remains feasible even if it corresponds to a deadhead for the crew at the time of operation. The set of constraints (4) ensures that there are at least α_k type A solutions to cover an extra flight $k \in \mathcal{K}$. The set of constraints (5) plays a similar role for type B solutions. The sets of constraints (6)-(8) prescribe that $x^k_{(p,q)}$ takes the value 1 if and only if $y_p = y_q = 1$. The formulation (2)-(11) has exponentially many variables (pairings). Moreover, in the worst case it may take an exponential amount of time to generate all constraints (6)-(8) because it would amount to generating all possible pairings explicitly and comparing them pairwise to identify all type A solutions for all extra flights. These two aspects render the model (2)-(11) both practically and theoretically challenging. The next section explains in detail how we address these issues.

3. Proposed Solution Approach.

The set covering formulation (1) of the conventional CPP includes exponentially many variables (pairings) as a function of the number of flights. Due to this structure, column generation is a frequently used technique for solving the linear programming relaxation of the CPP (see [9], [16], and [13] for reviews of these concepts). Column generation is then either embedded into a branch-and-price scheme or combined with a primal heuristic in order to find an optimal or a near-optimal integer solution. The column generation algorithm for the CPP is initialized by solving a restricted linear programming master problem (RMP), which contains all constraints, but only a subset of all feasible pairings in (1). Using the optimal dual values of the RMP, we next solve a pricing subproblem, where we search for a new column (pairing) with a negative reduced cost that improves the objective function value of the RMP. The pricing subproblem for the CPP is typically solved over an appropriately constructed flight/duty network (see Figure 3 for a flight network example), where the objective is to find a path from the source to the sink node with the minimum reduced cost that also satisfies restrictions on the flight and duty times, number of flights and duties in a pairing, and so on. In other words, each path in the flight/duty network has several attributes in addition to its associated reduced cost leading to a multi-label shortest path problem (MLSP) as the pricing subproblem for the CPP. The computational complexity of MLSP is exponential in the number of flights, and additional effort is required to solve it in a reasonable time. In Section 3.1, we cover the details of this difficult pricing problem tailored to the case of RCPEF. The column generation algorithm terminates when the MLSP yields no pairing with a negative reduced cost.

Applying a typical column generation algorithm to the linear programming relaxation of (2)-(11) would imply identifying all constraints (6)-(8) a priori before setting up the RMP, and this may correspond to enumerating all feasible pairings and comparing them pairwise in an effort to determine all type A solutions. This is clearly not computationally feasible and prompts us to initialize the RMP with a subset of the constraints (6)-(8) and the associated variables $x^k_{(p,q)}$. Clearly, this subset depends on type A solutions formed by the initial set of pairings. Then, during the process of generating new pairings, additional type A solutions are identified and their corresponding constraints and variables are introduced into the RMP on the fly. Type B solutions involve a single pairing, and all pairings $p \in \cup_{k\in\mathcal{K}}\mathcal{P}_B(k)$ with a negative reduced cost can be generated readily by the pricing subproblem. Thus, the sets $\mathcal{P}$, $\mathcal{P}_A(k)$, and $\mathcal{P}_B(k)$ are replaced by their subsets $\bar{\mathcal{P}}$, $\bar{\mathcal{P}}_A(k)$ and $\bar{\mathcal{P}}_B(k)$, respectively, when RCPEF is solved by column generation. To indicate that the RMP for RCPEF grows both column- and row-wise, we refer to it as the *restricted short master problem* (RSMP) as given below:

$$
\begin{aligned}
\text{minimize}\quad & \sum_{p\in\bar{\mathcal{P}}} c_p y_p + \sum_{i\in\mathcal{F}} (d_i s_i^+ + \sigma s_i^-) + \sum_{k\in\mathcal{K}} \tau(w_k^A + w_k^B) && && (12)\\
\text{subject to}\quad & \sum_{p\in\bar{\mathcal{P}}} a_{ip} y_p - s_i^+ + s_i^- = 1, && i \in \mathcal{F}, && (13)\\
& \sum_{(p,q)\in\bar{\mathcal{P}}_A(k)} x^k_{(p,q)} + w_k^A \geq \alpha_k, && k \in \mathcal{K}, && (14)\\
& \sum_{p\in\bar{\mathcal{P}}_B(k)} y_p + w_k^B \geq \beta_k, && k \in \mathcal{K}, && (15)\\
& y_p + y_q - x^k_{(p,q)} \leq 1, && (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, && (16)\\
& y_p - x^k_{(p,q)} \geq 0, && p \in \bar{\mathcal{P}} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, && (17)\\
& y_q - x^k_{(p,q)} \geq 0, && q \in \bar{\mathcal{P}} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, && (18)\\
& y_p \leq 1, && p \in \bar{\mathcal{P}}, && (19)\\
& x^k_{(p,q)} \leq 1, && (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, && (20)\\
& y_p \geq 0, && p \in \bar{\mathcal{P}}, && (21)\\
& x^k_{(p,q)} \geq 0, && (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, && (22)\\
& s_i^+, s_i^- \geq 0, && i \in \mathcal{F}, && (23)\\
& w_k^A, w_k^B \geq 0, && i \in \mathcal{K}. && (24)
\end{aligned}
$$

During the initialization phase of the column generation procedure detailed in Section 3.2, it turns out to be particularly difficult to determine α_k type A and β_k type B solutions for an extra flight k. Therefore, we relax the constraints (4) and (5) in RCPEF by including the variables w_k^A and w_k^B, $k \in \mathcal{K}$, on the left hand sides of the constraints (14) and (15), respectively. These artificial variables are penalized by a large parameter τ in the objective, which promotes pairings that form type A or B solutions during column generation. By the same token, we do not insist that the set of initial pairings covers all flight legs in the schedule. Therefore, slack variables $s_i^-, i \in \mathcal{F}$, are introduced into the left hand side of the constraints (13) and to advocate feasibility, we set the objective coefficients of these variables to a large value σ.

The challenge of solving the LP relaxation of RCPEF is rooted in our lack of ability to identify all constraints of type (16)-(18). Thus, the pricing subproblem is oblivious to the dual variables of the missing constraints, and we may terminate column generation prematurely before reaching optimality. For instance, a pairing p with a non-negative reduced cost in the current pricing subproblem may actually have a negative reduced cost because it could have formed a type A solution together with a pairing q that has not been generated yet. In other words, the computational intractability of keeping track of pairs of pairings which may form type A solutions during the pricing subproblem is the key difficulty in solving the LP relaxation of RCPEF optimally. We discuss this issue further in Section 3.1. To handle this difficult structure, we propose a two-level iterative heuristic approach. The main idea behind this approach is to fix the number of constraints of type (16)-(18) in RSMP during each iteration and apply column generation to optimality with these fixed constraints. Due to the lack of dual information corresponding to constraints of type (16)-(18) currently absent from the RSMP, the column generation procedure is not guided directly toward producing pairings that may result in a type A solution. However, our column generation procedure is tailored in order to detect pairings that can participate in a type A solution by examining the partial pairings in the flight network one by one. While we apply column generation, the candidate partial pairings that may potentially yield new constraints (type A solutions) are marked and stored separately. After the column generation terminates, we parse the stored columns and the columns present in the RSMP and identify whether they introduce new constraints. If so, then we include these constraints in the RSMP. The iterations then continue as before until no new columns are marked as candidates. A high-level overview of this approach is given in Algorithm 1 and each step is detailed in the following sections.

Algorithm 1: Algorithm to solve RCPEF.

```
 1: Initialize RSMP by generating and adding an initial set of columns
 2: repeat
 3:   repeat
 4:     Solve RSMP
 5:     Solve the pricing subproblem and mark the candidate type A pairings
 6:     if at least one pairing with a negative reduced cost is identified then
 7:       Add the column with the most negative reduced cost to RSMP
 8:     end if
 9:   until termination criteria are satisfied
10:   if new type A solutions are generated from marked pairings and pairings present in RSMP then
11:     Add these pairings, the corresponding constraints (16)-(18), and the auxiliary variables $x^k_{(p,q)}$ to
        RSMP
12:   end if
13: until no new type A solutions are generated from marked pairings and pairings present in RSMP
    or this loop is repeated at most $T$ times
14: If column generation terminates with a fractional solution, then apply a primal heuristic method or
    solve an integer programming problem constructed with the columns in RSMP
```

The main objective of the initialization procedure in Step 1 of Algorithm 1 is to construct type A solutions. For each type A solution (p, q) for extra flight k, the variables $y_p, y_q, x^k_{(p,q)}$ and three constraints that link these variables are added to the RSMP. Even for a small sized network, the initialization procedure is computationally expensive because it requires enumerating paths (pairings) in the flight/duty network and comparing them pairwise. Hence, we can only perform a partial search for type A solutions during the initialization procedure which is explained in detail in Section 3.2. We also note that the initial set of pairings introduced into the RSMP does not necessarily cover all flight legs in the schedule as mentioned earlier in this section. After the RSMP is initialized, column generation is invoked for the RSMP with a fixed number of constraints in Steps 3-9. Fixing the number of constraints in the RSMP keeps the number of dual variables constant, and column generation can be applied in the conventional sense.

In addition to determining new pairings with negative reduced costs, the pricing subproblem in our column generation method helps us detect new type A solutions (see Step 5). During the multi-label shortest path algorithm, if we stumble upon (partial) pairings that may potentially participate in a type A solution for an extra flight, a restricted number of these are marked and preserved throughout the algorithm and then stored in a column pool. After the column generation terminates with an optimal solution or a sufficiently small optimality gap with respect to the current RSMP (see Section 3.1.2), we parse through the pool and the columns present in the RSMP and check for new type A solutions in Steps 10-12. If new type A solutions are identified, we augment the RSMP with additional constraints and variables as required and re-invoke the column generation procedure for at most $T-1$ times. Otherwise, we proceed to the final step; if column generation provides us with a fractional solution at termination, then we obtain an integer feasible solution for RCPEF in Step 14 by a primal heuristic method or solving an integer programming problem using the columns in the final RSMP. Note that, the proposed approach given in Algorithm 1 is not an optimal method for solving the restricted linear programming master problem for RCPEF, unless we can generate *all* possible type A solutions before and during column generation. However, as we argue in Section 3.1 marking candidate columns (pairings) for potential type A solutions is computationally very demanding, and hence, we have no choice but to restrict the number of marked pairings while solving the multi-label shortest path problem.

In the rest of Section 3, we discuss the building blocks of our column generation algorithm in detail. First, the basic elements of the pricing subproblem are presented in Section 3.1 which also cover concepts common to the conventional CPP, such as searching for a path with a negative reduced cost on an appropriately constructed network. Here, we also elaborate on the modifications required for generating type B solutions and marking pairings for potential type A solutions while solving the pricing subproblem. The basic multi-label shortest path algorithm turns out to be too slow for any practical size problem, and additional features for pruning unfavorable partial pairings in the pricing subproblem and for the early termination of the column generation for the current RSMP with a guaranteed optimality gap are introduced in Section 3.1.1 and 3.1.2, respectively. The initialization routine that generates the first set of pairings and type A solutions is explained in Section 3.2, and the details of our column management for further speeding up the column generation and keeping track of type A solutions are provided in Section 3.3.

3.1 Pricing Subproblem: Multi-label Shortest Path (MLSP). During Steps 3-9 of Algorithm 1, column generation is applied to the RSMP with a fixed number of constraints. In other words, the corresponding *short master problem* (SMP) is obtained from (12)-(24) by replacing the sets $\bar{\mathcal{P}}$ and $\bar{\mathcal{P}}_B(k)$ by $\mathcal{P}$ and $\mathcal{P}_B(k)$, respectively. However, both in SMP and RSMP, we only include a subset $\bar{\mathcal{P}}_A(k)$ of all possible type A solutions $\mathcal{P}_A(k)$ for extra flight k. Recall that the reduced cost of a variable is given by the infeasibility in the associated dual constraint, and therefore the objective of the pricing subproblem in the column generation algorithm is to identify at least one violated constraint in the dual model. In our subsequent discussion, we shall refer to the dual of SMP as DSMP. Let the dual variables corresponding to the sets of constraints (13)-(20) in SMP be u_i, z_k^A, z_k^B, $\delta^1_{(p,q),k}$, $\delta^2_{(p,q),k}$, $\delta^3_{(p,q),k}$, γ_p^y, $\gamma^x_{(p,q),k}$, respectively. Then the mathematical model of DSMP becomes

$$\begin{aligned}
\text{maximize} \quad & \sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}}(\alpha_k z_k^A + \beta_k z_k^B) + \sum_{p\in\mathcal{P}} \gamma_p^y + \\
& \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \gamma^x_{(p,q),k}) && (25)\\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \delta^2_{(p,q),k}) + \\
& \sum_{k\in\mathcal{K}} \sum_{\{(q,p):(q,p)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(q,p),k} + \delta^3_{(q,p),k}) + \\
& \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B + \sum_{i\in\mathcal{F}} a_{ip}u_i + \gamma_p^y \le c_p, && p\in\mathcal{P}, \quad (26)\\
& z_A^k - \delta^1_{(p,q),k} - \delta^2_{(p,q),k} - \delta^3_{(p,q),k} + \gamma^x_{(p,q),k} \le 0, && (p,q)\in\bar{\mathcal{P}}_A(k), k\in\mathcal{K}, \quad (27)\\
& u_i \ge -d_i, && i\in\mathcal{F}, \quad (28)
\end{aligned}$$

$$u_i \leq \sigma, \qquad i \in \mathcal{F}, \tag{29}$$
$$z_k^A \leq \tau, \qquad k \in \mathcal{K}, \tag{30}$$
$$z_k^B \leq \tau, \qquad k \in \mathcal{K}, \tag{31}$$
$$z_k^A, z_k^B \geq 0, \qquad k \in \mathcal{K}, \tag{32}$$
$$\delta^1_{(p,q),k} \leq 0, \qquad (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{33}$$
$$\delta^2_{(p,q),k} \geq 0, \qquad p \in \mathcal{P} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{34}$$
$$\delta^3_{(p,q),k} \geq 0, \qquad q \in \mathcal{P} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{35}$$
$$\gamma_p^y \leq 0, \qquad p \in \mathcal{P}, \tag{36}$$
$$\gamma^x_{(p,q),k} \leq 0, \qquad (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}. \tag{37}$$

The optimal dual variables provided by the current RSMP may violate one or several of the constraints (26) corresponding to pairings that have not been generated yet. The magnitude of violation for pairing $p \in \mathcal{P}$ is referred to as the reduced cost of p and denoted by $\bar{c}_p$, where $\bar{c}_p$ is calculated by

$$\begin{aligned}\bar{c}_p &= c_p - \sum_{i\in\mathcal{F}} a_{ip}u_i - \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B - \\ &\sum_{k\in\mathcal{K}}\left(\sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}}(\delta^1_{(p,q),k}+\delta^2_{(p,q),k}) + \sum_{\{(q,p):(q,p)\in\bar{\mathcal{P}}_A(k)\}}(\delta^1_{(q,p),k}+\delta^3_{(q,p),k})\right) - \gamma_p^y \\ &= c_p - \sum_{i\in\mathcal{F}} a_{ip}u_i - \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B.\end{aligned} \tag{38}$$

Thus, the objective of the pricing subproblem is to determine at least one pairing p with $\bar{c}_p < 0$. The last two terms in the first equality in (38) disappear because a pairing p that has not been generated yet cannot appear in any solution in $\bar{\mathcal{P}}_A(k), k \in \mathcal{K}$, and $y_p = 0$ implies $\gamma_p^y = 0$. Furthermore, the structure of DSMP suggests that $u_i = \sigma$ in any optimal solution of an RSMP in which flight i is not covered by any pairing. Similarly, if the current RSMP does not include any type B solution for an extra flight k, then $z_k^B = \tau$ at optimality. In other words, the slack variables s_i^- and w_k^B with large objective coefficients in the RSMP promote that early in the column generation procedure all flights appear in at least one pairing and all extra flights are covered by at least one type B solution.

In crew pairing problems, it is convenient to represent pairings on a flight or duty network. A small flight network which contains three flights i_1, i_2, and i_3 on a given day is shown in Figure 3, where d_i and a_i stand for the departure and arrival airport of flight i, respectively. Starting and ending at the crew base station in City 1, a single pairing covers all three flights in one duty period. The sit connections between the consecutive flights i_1 and i_2, and i_2 and i_3 are denoted by the dashed lines. For our purposes, the flight network is more appropriate because extra flights or their associated deadheads may be inserted in the middle of a duty. The reader is referred to [25] for the duty network representation.

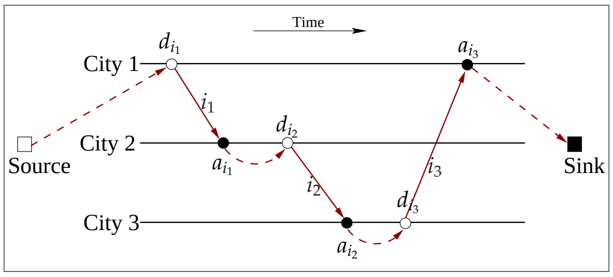


Figure 3: A flight network with crew base City 1.

Formally, for a given set of flights $\mathcal{F}$, we denote our flight network with $G = (V, E)$, where V denotes the set of nodes and E denotes the set of arcs. As illustrated in Figure 3, we add one departure node d_i and one arrival node a_i to V for each flight $i \in \mathcal{F}$. Each pairing is denoted by a path in the network that originates at a dummy *source* node and terminates at a dummy *sink* node. Thus, $V = \bigcup_{i\in\mathcal{F}}\{d_i, a_i\} \cup \{source, sink\}$. The set E consists of four different sets of arcs $E = E_F \cup E_C \cup E_B \cup E_0$. The set of *flight arcs* is defined as $E_F = \{(d_i, a_i) \mid i \in \mathcal{F}\}$. The set of *connection arcs* given by E_C are defined as follows. For two flights

$i, j \in \mathcal{F}$, $(a_i, d_j) \in E_C$ if and only if the arrival airport of i and the departure airport of j are the same, and the time lag between the departure time of j and the arrival time of i is either no less than a minimum sit time and does not exceed a maximum sit time, or is no less than a minimum rest time and does not exceed a maximum rest time. These arcs represent both feasible sit and rest connections between two successive flights for a crew. Every departure from the crew base may be the start of a pairing and every arrival at the crew base may complete a pairing. Inserting the set of arcs $E_B = \{(source, d_i) \mid i \in \mathcal{F}$ and i departs from the crew base $\} \cup \{(a_i, sink) \mid i \in \mathcal{F}$ and i arrives at the crew base $\}$ into G enables us to model this feature. Depending on the planning horizon of the crew pairing problem, it may be impossible to cover certain flights by pairings that start and complete during the current planning period. For such a lingering flight i, d_i has no incoming arc or a_i has no outgoing arc in $E_F \cup E_C \cup E_B$. The only means of including these flights in the pairing solution in the current planning period is by repositioning crews by deadhead flights on other airlines or by ground transportation. In order to represent this possibility, we define the set of arcs $E_0 = \{(source, d_i) \mid i \in \mathcal{F}$ and d_i has no incoming arc in $E_F \cup E_C \cup E_B\} \cup \{(a_i, sink) \mid i \in \mathcal{F}$ and a_i has no outgoing arc in $E_F \cup E_C \cup E_B\}$.

Observe that the flight network G is a directed acyclic graph, and hence, without loss of generality we can assume that the nodes of G are topologically sorted. Any path from the *source* to the *sink* node is a potential pairing subject to several feasibility rules such as the lower and upper bounds on the connection times between two consecutive flights, the limits on the maximum number of flights and the maximum elapsed time in a duty period, and so on. Therefore, although the restrictions on the minimum and maximum sit times may be verified on the fly while the flight network is constructed, other rules pertaining to duty period and pairing feasibility and the cost structure require us to store several attributes for a (partial) path from the *source* node to any node $v \in V$ in the flight network. A vector of attributes is referred to as a *label*, and in this work we consider a set of six attributes for ensuring feasibility and computing cost. The attributes l_1 through l_5 are all commonly used in crew pairing problems. These are the sum of the costs of the completed duty periods (l_1), the sum of the dual values of the flights covered (l_2), the total elapsed time of the current duty period (l_3), the total number of flights covered by the current duty period (l_4), and the cost of the current duty period (l_5). We designate the final attribute l_6 to keep track of the partial paths in the flight network which may potentially form type A or B solutions. The possible values for this attribute are $\bigcup_{k \in \mathcal{K}} \{A^e_k, A^d_k, B_k\} \cup \{none\}$, where

- ⋄ *none* means that the partial pairing does not provide a type A or B solution for any extra flight,
- ⋄ A^e_k means that the partial pairing may provide a type A solution for the extra flight k together with another pairing and it covers the extra flight itself (see Figure 1),
- ⋄ A^d_k means that the partial pairing may provide a type A solution for the extra flight k together with another pairing and it covers the deadhead flight corresponding to k (see Figure 1), and
- ⋄ B_k means that the partial pairing provides a type B solution for the extra flight k (see Figure 2).

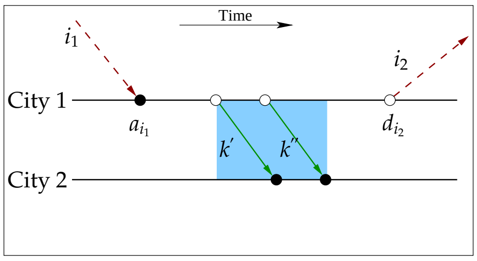
Thus, a (partial) path p from the *source* node to a node $v \in V$ in the flight network is represented by a sequence of attributes $l^p = \langle l^p_1, l^p_2, ..., l^p_6 \rangle$, which we refer to as a label, and our objective in the pricing subproblem is to determine a feasible path from the *source* node to the *sink* node with the minimum reduced cost. Hence, this problem is referred to as a multi-label shortest path problem or a shortest path problem with resource constraints (SPPRC) (see [11] for a comprehensive survey on the subject). In the literature, pairing costs with nonlinear structures are frequently considered [2, 12]. That is, the cost of a pairing may be different than the sum of the costs on the arcs of the corresponding path in the flight network. However, the crew pairing instances considered in this study are obtained from one major and one relatively smaller airline company in Turkey, and they exhibit additive cost structures. For both of the companies, the main objective is to utilize their available crews as efficiently as possible by reducing idle time in their schedules. Thus, we charge a cost to both sit and rest connections, where rest connections also incur an overnight accommodation cost. The key observation here is that the cost of a partial path can be updated easily while traversing a sit or rest connection arc in the flight network. This cost is computed as $l^p_1 + l^p_5$ for a path p on node $v \in V$ from the *source* node to v. Then, for any (complete) path l^p on the *sink* node, which represents a feasible pairing p, the reduced cost is determined as $\bar{c}_p = l^p_1 + l^p_5 - l^p_2 = l^p_1 - l^p_2$, where $l^p_5 = 0$ for any such path because a feasible pairing is constituted by completed duty periods only. A pairing with the most negative reduced cost is then added to the RSMP.

Our multi-label shortest path algorithm is adapted from the algorithms by Desrosiers *et al.* [8]. The set of partial paths that terminate at $v \in V$ is denoted by $\mathcal{L}^v$, where the cardinality of this set is bounded from above by the number of all possible paths from the *source* node to v. The basic operation in the multi-label shortest path algorithm is to extend the partial paths on v by one more node by traversing all arcs (v, v') originating at v. Each (partial) path $l^p \in \mathcal{L}^v$ is modified according to the properties of the arc (v, v'), and the modified (partial) path is added to $\mathcal{L}^{v'}$ if it corresponds to a feasible (partial) path from the *source* node to v'. Clearly, some of the paths in $\mathcal{L}^v$ may be omitted from further consideration during this operation because they are inferior. Formally, we define a partial ordering relation $\preceq_d$ on $\mathcal{L}^v$:

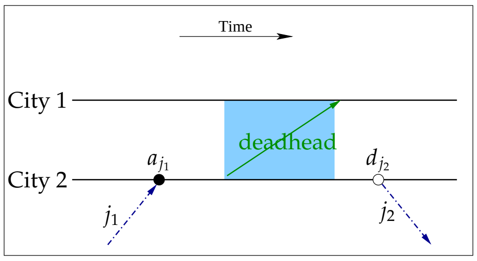
Definition 3.1 *Let $l^p, l^q \in \mathcal{L}^v$ be two (partial) paths p and q from the source node to $v \in V$. $l^p \preceq_d l^q$, if $l_1^p + l_5^p - l_2^p \le l_1^q + l_5^q - l_2^q$, $l_3^p \le l_3^q$, $l_4^p \le l_4^q$ and $l_6^p = l_6^q$. If $l^p \preceq_d l^q$, then p is said to dominate q.*

In the worst case, all paths in the flight network may be non-dominated, and determining the optimal solution of the MLSP may be equivalent to enumerating all paths from the *source* to the *sink* node in the flight network. This complete enumeration yields a worst-case complexity that is exponential in the number of flights. Otherwise, all dominated paths from the *source* node to node $v \in V$ may be excluded from further consideration in the multi-label shortest path algorithm while preserving optimality. Hence, without loss of generality we assume that $\mathcal{L}^v$, $v \in V$, are composed of non-dominated (partial) paths in the rest of the paper. Moreover, note that unless their final attributes (l_6) are identical, two partial pairings cannot dominate each other. In other words, the final attribute partitions the set of partial paths $\mathcal{L}^v$ on node $v \in V$ into mutually exclusive subsets, and the domination rule in Definition 3.1 is applied separately to each subset.

Finally, we explain how we mark partial pairings, which may potentially form type A or B solutions, by setting their final attributes to one of the appropriate values $\bigcup_{k \in \mathcal{K}} \{A_k^e, A_k^d, B_k\}$. Otherwise, the default value for l_6 is *none* for all partial paths. In general, a partial path can cover different extra flights. However, in such a case, we create copies of this partial path where each copy handles only one of the extra flights. This approach simplifies both the discussion and the implementation. Besides the existing feasibility rules (such as; the total elapsed time of the current duty period, the total number of flights covered by the current duty period), other rules that we additionally require for type A solutions are illustrated in Figure 4. For all partial paths p that terminate with a flight arc $(d_{i_1}, a_{i_1}) \in E_F$ at the departure station of an extra flight k, we first identify all feasible sit or rest connections (a_{i_1}, d_{i_2}). If this feasible connection spans the estimated time window of extra flight k and the time lag between the arrival time of i_1 and the departure time of extra flight k is a feasible sit connection or a feasible rest connection, regardless of where k is located in its estimated time window, then $l_6^p = A_k^e$. This is depicted in Figure 4(a) where k' and k'' denote the earliest and latest possible realizations of extra flight k, respectively. Similarly, for all partial paths q that terminate with a flight arc $(d_{j_1}, a_{j_1}) \in E_F$ at the arrival station of an extra flight k, we first identify all feasible sit or rest connections (a_{j_1}, d_{j_2}). If the estimated time window of extra flight k is contained within this connection time and the period of time between the arrival time of j_1 and the earliest departure time of extra flight k is either a feasible sit or rest connection as defined for deadhead flights, then $l_6^q = A_k^d$. This is illustrated in Figure 4(b). In addition, observe that a restriction on the departure time of the deadhead flight is essential; otherwise, a huge number of partial paths may be marked as A_k^d, which would increase the solution time of MLSP considerably. This would also impact the time spent at Step 10 in Algorithm 1 and the time to solve RSMP in the following iterations of Algorithm 1 due to a large number of additional constraints of type (16)-(18).



(a) l_6^p is set to A_k^e for partial pairing p.



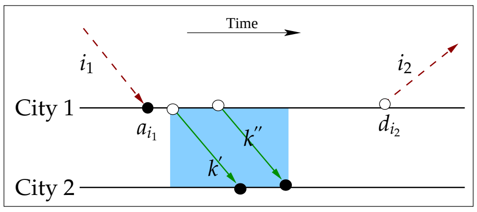
(b) l_6^q is set to A_k^d for partial pairing q.

Figure 4: Identifying partial pairings which may potentially participate in type A solutions.

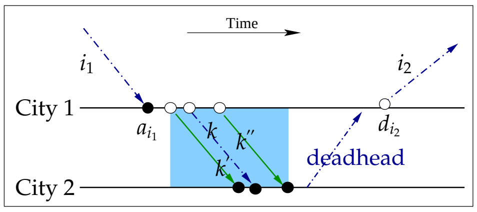
The procedure described above does also shed light on why it is very difficult to account for the type

A solutions while calculating the reduced cost of a pairing in the MLSP. As described above, all we can determine for a partial path is that it may be completed to a full pairing p that may cover either an extra flight or its associated deadhead in a type A solution. However, ultimately the existence of a complementary pairing q determines whether pairing p participates in a type A solution or not, and keeping track of such cross-effects is computationally intractable in the MLSP algorithm.

A type B solution requires a single pairing with sufficient connection time to cover both an extra flight and its associated deadhead as demonstrated in Figure 2. Therefore, we can verify all relevant feasibility rules on the fly and determine the reduced cost of a pairing that provides a type B solution in the MLSP algorithm. In the following, we analyze the case where an extra flight k is operated before deadheading back to d_k. The alternate case in which the deadhead flight precedes the extra flight may be analyzed analogously and is omitted here. First, for all partial paths p that terminate with a flight arc $(d_{i_1}, a_{i_1}) \in E_F$ at the departure station of an extra flight k, we identify all feasible sit or rest connections (a_{i_1}, d_{i_2}). In other words, if we append arc (a_{i_1}, d_{i_2}) to p, we ensure that we obtain a feasible partial path to be added to $\mathcal{L}^{d_{i_2}}$ (see Figure 5(a)). In addition, if the estimated time window of extra flight k is included in the feasible connection (a_{i_1}, d_{i_2}), then we confirm two conditions for all possible departure times of k in its estimated time window. First, we verify that the period of time from the arrival time of flight i_1 to the departure time of extra flight k is a feasible sit or rest connection. Second, we ensure that there is sufficient time between the arrival time of k and the departure time of i_2 for covering the duration of the deadhead flight and two feasible connections before and after the deadhead. If these conditions hold, we create a copy q of the partial path p and set $l_6^q = B_k$. After appending (a_{i_1}, d_{i_2}) to both p and q, we insert them into $\mathcal{L}^{d_{i_2}}$ if they are also feasible otherwise.



(a) Original partial pairing p, l_6^p is set to *none*.



(b) Partial pairing q obtained from p, l_6^q is set to B_k.

Figure 5: Identifying type B solutions in the MLSP algorithm.

Intuitively, pairings that form type A or B solutions have longer connection times, and consequently, they are more costly. This poses an important problem for partial pairings that may potentially participate in type A solutions; i.e., for partial pairings of the form $p = \langle l_1^p, l_2^p, \ldots, l_5^p, A_k^e \rangle$ and $p = \langle l_1^p, l_2^p, \ldots, l_5^p, A_k^d \rangle$. Such a partial pairing p incurs a relatively higher cost as reflected by the sum $l_1^p + l_5^p$, but the sum of the dual values l_2^p it accumulates does not account for its potential to form a type A solution for extra flight k. In other words, p is likely to be dominated and removed from further consideration based on these criteria only. This is precisely why we prescribe in Definition 3.1 that p may only be dominated by another partial pairing q that carries the same value for the final attribute. However, this approach creates a computational challenge. In preliminary experiments, we observe that if we keep all non-dominated partial paths according to Definition 3.1, then a very large number of partial pairings with high costs may be carried all the way to the *sink* node, which slows down our algorithms substantially and provides no or very slim benefits. Thus, in order to strike a balance between computational efficiency and obtaining a sufficient number of pairings from MLSP that may help us construct type A solutions, we keep at most $N_{\max}^A$ non-dominated partial pairings at a node $v \in V$ during MLSP whose final attributes are either set to A_k^e or A_k^d for some extra flight k. The sensitivity of our solution quality and the computation time to this parameter is explored in our computational study given in Section 4. Moreover, we assert that no special provisions are required for type B solutions. Following our discussion earlier in this section, we observe that while the original pairing p in Figure 5(a) is likely to be dominated, the long connection time between the arrival time of i_1 and the departure time of i_2 is now utilized efficiently by covering an extra flight and its associated deadhead in the modified pairing q in Figure 5(b), and in return l_2^q incorporates the dual value of constraint (15) corresponding to extra flight k. Thus, if we can extend q to a complete feasible pairing all information is available to calculate the associated reduced cost correctly. In the remainder of Section 3.1, we enhance the basic MLSP algorithm for computational efficiency.

3.1.1 Pruning Methods. MLSP is a computationally expensive problem due to the fact that there may be exponentially many paths from the source node to a node $v \in V$. Thanks to the feasibility rules and the domination between partial paths, not all of these paths need to be computed. However, in practice still a huge number of feasible partial paths that are mutually incomparable by relation $\preceq_d$ may accumulate on v.

In order to reduce the number of partial paths accumulated on a node v, we apply two pruning methods. The pruning rules used in this work are similar to those given in [17] in terms of the underlying intuition of the rules and their types. As in [17], we also have two kinds of rules: *approximate* and *exact*. However, modifications are required because our underlying mathematical model, the pairing cost structure, and the type of network representation employed for the pricing problem are different.

The exact rule is conservative in the sense that if a partial path p on node v is pruned by the exact rule, then there is no possible way of completing p into a pairing that corresponds to a column with a negative reduced cost. The approximate rule is based on a heuristic with a simplifying assumption for estimating the reduced cost of a pairing from a prefix of that pairing. This rule yields faster MLSP iterations but is not conservative. In other words, if an MLSP iteration employing the approximate rule cannot find a column with a negative reduced cost, then this might be due to an over-pruning realized by this rule. Therefore after such an unsuccessful MLSP iteration, another iteration of MLSP is invoked, where only the exact pruning rule is active. We now explain these pruning rules in detail.

Approximate Rule. To reduce the number of partial paths accumulated on a node, we first apply the approximate rule. The intuition underlying this rule is that a partial path will be extended to a complete pairing by appending edges with average cost and average dual values. Formally, we define the average cost of an edge and the maximum pairing length over all the pairings generated so far as

$$\bar{c} = \frac{\sum_{p \in \bar{\mathcal{P}}} c_p}{\sum_{p \in \bar{\mathcal{P}}} |p|} \text{ and } \lambda = \max_{p \in \bar{\mathcal{P}}}\{|p|\}, \tag{39}$$

respectively. Here, $|p|$ denotes the length of the (partial) path p measured in the number of edges in the flight network G. We also calculate the average dual values of the regular and extra flights covered by all the generated pairings by

$$\bar{u} = \frac{\sum_{p \in \bar{\mathcal{P}}} \sum_{i \in \mathcal{F}} a_{ip} u_i}{\sum_{p \in \bar{\mathcal{P}}} |p|} \text{ and } \bar{z} = \frac{\sum_{k \in \mathcal{K}} z_k^B |\bar{\mathcal{P}}_B(k)|}{\sum_{p \in \bar{\mathcal{P}}} |p|}, \tag{40}$$

respectively.

We assume that $\lambda - |p|$ more edges will be appended to a partial pairing p, resulting in a full pairing. The cost of the edges and the dual values of the regular and extra flights already covered by p are known. The cost of the edges and the dual values for the regular and extra flights yet to be covered by p are estimated by using the average values given in relations (39) and (40). This value is given by

$$\tilde{e}_p = \underbrace{\left(l_1^p + l_5^p - l_2^p\right)}_{\text{known}} + \underbrace{(\lambda - |p|)(\bar{c} - \bar{u} - \bar{z})}_{\text{estimated}}. \tag{41}$$

Suppose q is a pairing obtained by extending p; i.e., p is a prefix of q when p and q are considered as a sequence of edges in the flight network. Obviously, $\tilde{e}_p$ does not provide a definite information about the reduced cost of q, however we use it as an estimate. If $\tilde{e}_p < 0$, then we state that q is likely to have a negative reduced cost, and hence, we do not prune any partial path in the set $\{p \in \mathcal{L}^v | \tilde{e}_p < 0\}$. Nonetheless, it is still possible for q to have a negative reduced cost when $\tilde{e}_p \geq 0$. Therefore, we only keep a subset of predefined size and discard the rest from the partial paths in the set $\{p \in \mathcal{L}^v | \tilde{e}_p \geq 0\}$.

Exact Rule. In the exact rule, we obtain a lower bound on the reduced cost of any partial path from any node $v \in V$ to the *sink* node by solving an appropriate shortest path problem in the flight network $G(V, E)$, where the length of the shortest path from a node $v \in V$ to the *sink* node is denoted by δ_v. The cost of a connection arc is computed based on the cost per unit time for sit connections (which is smaller than that for rest connections) in order to ensure a lower bound on the reduced cost. In addition, we subtract z_k^B from the cost of a connection arc, if extra flight k may be covered as a type B solution during the corresponding connection time. An arc for flight i is assigned a cost of $-u_i$. Note that δ_v may change from one iteration to another, since it is based on the dual values.

When applying the exact rule, a *partial reduced cost* of each partial path p on a node v is calculated by

$$e_p = l_1^p + l_5^p - l_2^p, \quad p \in \mathcal{L}^v. \tag{42}$$

The following result shows the use of e_p and how the exact rule identifies partial paths on a node v that cannot possibly lead to full pairings with a negative reduced cost.

Proposition 3.1 *Let $p \in \mathcal{L}^v$ be a partial path on v and q be a pairing obtained by extending p to a complete pairing. If $e_p + \delta_v \geq 0$, then the reduced cost of q is non–negative.*

Proof. Suppose that r is the path from node v to the *sink* node such that when r is appended to p we get q. Let $\bar{c}_r$ be the reduced cost of r. By using the definition of δ_v given above we have

$$\bar{c}_r \geq \delta_v. \tag{43}$$

Based on this we can find a lower bound on the reduced cost of q as

$$\begin{aligned} \bar{c}_q &= l_1^q + l_5^q - l_2^q \\ &= l_1^p + l_5^p - l_2^p + \bar{c}_r \\ &\geq l_1^p + l_5^p - l_2^p + \delta_v \\ &= e_p + \delta_v. \end{aligned} \tag{44}$$

Finally by using the premise of the proposition and relation (44), we obtain $\bar{c}_q \geq 0$. □

Intuitively, δ_v gives the best way of extending a partial path p terminating at v to a complete pairing q, although such an advantageous extension may not be feasible. Moreover, if q does not have a negative reduced cost, then all feasible pairings obtained by extending p are guaranteed to have a non-negative reduced cost.

Note that from a computational point of view, the exact rule is not more expensive than the approximate rule. The computations required for $\bar{u}$ and $\delta_v, v \in V$, need to be performed once for each iteration of MLSP but they both can be performed in linear time. The other computations required for each iteration of MLSP can be performed in constant time with an appropriate bookkeeping. In practice however, the approximate rule leads to much faster MLSP iterations, because it discards a larger number of partial paths at each node as compared to the exact rule. We use both pruning methods hand in hand in our MLSP implementation. The implementation uses the approximate rule to speed up the pruning and employs the exact rule when the approximate rule fails. We conduct a set of experiments to justify this strategy and present the results at the beginning of Section 4.

3.1.2 Computing a Lower Bound for RSMP. Column generation algorithms typically obtain a good solution quickly, but they suffer from a so-called "tailing-off effect," and they may take a very long time to prove optimality (see [3] for a discussion on this issue). Thus, column generation algorithms are frequently terminated prematurely before achieving optimality, especially if a guaranteed optimality gap can be computed. In the context of this work, we list three additional reasons in favor of early termination of the column generation. First, if a set of pairings is already marked for potential type A solutions in the course of solving the current RSMP, then we are almost sure that column generation will be re-invoked after new constraints and variables are introduced into the RSMP (see Steps 10-12 in Algorithm 1). Second, our ultimate goal is to obtain an integer feasible solution from the LP solution after column generation is finished for the final RSMP (see Step 14 in Algorithm 1). To this end, a near-optimal solution may just suffice. Third, MLSP iterations are expensive. In the sequel, we explain how to calculate an optimality gap for the current solution of the RSMP each time MLSP is solved with the exact pruning rule in Section 3.1.1. This helps us terminate the column generation procedure when a near-optimal solution is identified. Similar lower bounding strategies are used for the cutting-stock problem in [24] and [26]. Also, see [5], [28].

The optimal objective function value z_{RSMP} of the current restricted short master problem RSMP is an upper bound on the optimal objective value z_{SMP} of the short master problem SMP, where the relationship between RSMP and SMP is described at the beginning of Section 3.1. Moreover, duality theory states that the objective value of any feasible solution to the dual DSMP of SMP provides a lower bound on z_{SMP} and that the reduced cost of a variable is the infeasibility in the corresponding dual constraint. These two facts together imply that we can construct a feasible solution for DSMP based on the optimal dual values from the current RSMP and the optimal objective function value of the MLSP which yields the maximum infeasibility over all constraints in DSMP. An optimality gap is then computed by employing the objective function value of this dual feasible solution. The formulation (45)-(48) below is equivalent to DSMP and is obtained from (25)-(37) by applying the transformation $u_i = u_i' - d_i, i \in \mathcal{F}$:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i\in\mathcal{F}} u_i' + \sum_{k\in\mathcal{K}} (\alpha_k z_k^A + \beta_k z_k^B) + \sum_{p\in\mathcal{P}} \gamma_p^y + \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \gamma^x_{(p,q),k}) - \sum_{i\in\mathcal{F}} d_i && (45)\\
\text{subject to} \quad & \sum_{i\in\mathcal{F}} a_{ip} u_i' + \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B + \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \delta^2_{(p,q),k}) + \\
& \sum_{k\in\mathcal{K}} \sum_{\{(q,p):(q,p)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(q,p),k} + \delta^3_{(q,p),k}) + \gamma_p^y \leq c_p + \sum_{i\in\mathcal{F}} a_{ip} d_i, \qquad p \in \mathcal{P}, && (46)\\
& (27), (30) - (37), \\
& u_i' \leq \sigma + d_i, \qquad i \in \mathcal{F}, && (47)\\
& u_i' \geq 0, \qquad i \in \mathcal{F}. && (48)
\end{aligned}$$

Now, assume that the optimal dual values corresponding to constraints (13)-(20) are retrieved after solving the current RSMP to optimality, and the MLSP is executed with the exact pruning rule afterwards. For every pairing p with $\bar{c}_p < 0$ according to (38), we define a score value $\bar{s}_p$ (see [4]) based on (46):

$$\bar{s}_p = \frac{c_p + \sum_{i\in\mathcal{F}} a_{ip} d_i}{\sum_{i\in\mathcal{F}} a_{ip} u_i' + \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B} = \frac{c_p + \sum_{i\in\mathcal{F}} a_{ip} d_i}{\sum_{i\in\mathcal{F}} a_{ip} u_i + \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B + \sum_{i\in\mathcal{F}} a_{ip} d_i}, \tag{49}$$

where $0 < \bar{s}_p < 1$. Computing $0 < \bar{s}_{\min} = \min_{\{p\in\mathcal{P}:\bar{c}_p<0\}} \bar{s}_p < 1$, we can construct a feasible solution for (45)-(48):

$$\begin{array}{ll}
\hat{u}_i = \bar{s}_{\min} u_i' = \bar{s}_{\min}(u_i + d_i), & i \in \mathcal{F}, \\
\hat{z}_k^A = z_k^A, & k \in \mathcal{K}, \\
\hat{z}_k^B = \bar{s}_{\min} z_k^B, & k \in \mathcal{K}, \\
\hat{\delta}^1_{(p,q),k} = \delta^1_{(p,q),k}, & (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \\
\hat{\delta}^2_{(p,q),k} = \delta^2_{(p,q),k}, & p \in \mathcal{P} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \\
\hat{\delta}^3_{(p,q),k} = \delta^3_{(p,q),k}, & q \in \mathcal{P} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \\
\hat{\gamma}_p^y = \gamma_p^y, & p \in \mathcal{P}, \\
\hat{\gamma}^x_{(p,q),k} = \gamma^x_{(p,q),k}, & (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K},
\end{array} \tag{50}$$

where $0 \leq \hat{u}_i \leq u_i' = (u_i + d_i), i \in \mathcal{F}$, and $0 \leq \hat{z}_k^B \leq z_k^B, k \in \mathcal{K}$. In order to see why the values in (50) are feasible for (45)-(48), observe that we decrease the left hand sides of the violated constraints in (46) by scaling down the non-negative variables $u_i', i \in \mathcal{F}$, and $z_k^B, k \in \mathcal{K}$, by at least the required amount to restore their feasibility. This scaling also clarifies why we need the transformation $u_i = u_i' - d_i, i \in \mathcal{F}$, which leads to non-negative variables $u_i' = u_i + d_i, i \in \mathcal{F}$. Moreover, the feasibility of the constraints which are already satisfied is not affected by our scaling. The only difference in the objective function value associated with the dual feasible solution (50) with respect to z_{RSMP} is due to the scaling of $u_i', i \in \mathcal{F}$, and $z_k^B, k \in \mathcal{K}$, by $\bar{s}_{\min}$. We have

$$\hat{z}_{\text{DSMP}} = z_{\text{RSMP}} - (1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u_i' + \sum_{k\in\mathcal{K}} \beta_k z_k^B \right) = z_{\text{RSMP}} - (1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}} \beta_k z_k^B + \sum_{i\in\mathcal{F}} d_i \right), \tag{51}$$

which provides a lower bound on z_{SMP}. Thus,

$$\frac{z_{\text{RSMP}} - z_{\text{SMP}}}{z_{\text{SMP}}} \leq \frac{z_{\text{RSMP}} - \hat{z}_{\text{DSMP}}}{\hat{z}_{\text{DSMP}}} = \frac{(1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}} \beta_k z_k^B + \sum_{i\in\mathcal{F}} d_i \right)}{z_{\text{RSMP}} - (1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}} \beta_k z_k^B + \sum_{i\in\mathcal{F}} d_i \right)}. \tag{52}$$

The relation (52) yields an upper bound on the optimality gap of the current RSMP, and the column generation algorithm may be stopped before achieving optimality, if this upper bound is sufficiently small. This is a partial remedy to tailing off.

3.2 Initialization Procedure. In general, the primary function of the initialization step in a column generation algorithm is to provide an initial feasible solution for the restricted master problem. Moreover, a good initial solution leads to proper dual information passed to the pricing subproblem early in the column generation. In this study, we address the feasibility of the RSMP by incorporating artificial variables in the coverage constraints for both regular and extra flights. In our case, the challenging issue is to identify a good number of type A solutions a priori because the MLSP algorithm described in Section 3.1 identifies pairings that may potentially lead to new type A solutions by chance only, and each execution of the MLSP is computationally expensive. Therefore, at the beginning of Algorithm 1, we run a slightly modified MLSP algorithm that is tailored toward determining as many type A solutions as possible. The pairings generated in the initialization routine constitute the initial set $\bar{\mathcal{P}}$. Furthermore, for each feasible type A solution (p, q) for extra flight k we insert (p, q) into $\bar{\mathcal{P}}_A(k)$ and add the variables $y_p, y_q, x^k_{(p,q)}$ as well as the three constraints of type (16)-(18) that link these variables to the RSMP. The procedure outlined in the following is repeated for each $k \in \mathcal{K}$: The first step in the initialization is to identify those nodes in the flight network $G(V, E)$ that may immediately precede an extra flight k or its associated deadhead in a type A solution. Then, starting at each of these nodes we perform a breadth- or depth-first-search in G, except that the directions of all arcs in E are reversed. The intention here is to mark all nodes in V which may appear before an extra flight or its associated deadhead in a pairing that may form a type A solution. We refer to this set of nodes as V_k and define $E_k = \{(i, j) \in E : i \in V_k, j \in V_k\}$. Then, we determine pairings for potential type A solutions in two phases. First, we call the MLSP algorithm over $G(V_k, E_k)$, where the objective is to find the set of all paths from the *source* node to the nodes that may immediately precede an extra flight k or its associated deadhead in a type A solution. In the second phase, we update the final attributes of all partial paths terminating at these nodes based on the feasibility conditions discussed at the end of Section 3.1. Next, we only take into account those partial paths marked as A^e_k or A^d_k in their final attributes and complete them to full pairings by the MLSP algorithm. Finally, when the MLSP algorithm terminates, we match up appropriate pairings and check the feasibility rules for type A solutions. RSMP is modified as required for all feasible type A solutions.

During the initialization routine, dual information from RSMP is not available and cost is of no concern. Therefore, the domination rule in Definition 3.1 is not applied. Unfortunately, this may lead to a prohibitively large number of paths and excessive computation times in the MLSP. To alleviate this, we impose an upper bound $N^{init}_{\max}$ on the number of (partial) paths that can accumulate on each node during the MLSP algorithm. Clearly, if it is possible to carry out the initialization in the absence of this parameter, then we can determine all constraints (6)-(8) upfront, and RCPEF can then be solved by conventional column generation. However, this is only possible for small problem instances, and we investigate the sensitivity of the objective function value and the ultimate number of type A solutions identified by our algorithms to the value of $N^{init}_{\max}$ in Section 4.

3.3 Column Management. Column management is an integral part of any successful column generation algorithm as discussed in detail by Barnhart *et al.* [3]. In this research, column management has two primary goals. First, solving MLSP is computationally very expensive, and if possible we would like to avoid it in some iterations of the column generation algorithm. Second, as we explain in detail in Section 3.1, MLSP can only mark pairings which may potentially lead to type A solutions, and we need to store such candidate pairings until the termination of the column generation algorithm (Steps 3-9 of Algorithm 1) before we can determine whether any type A solution may be constructed employing these stored pairings and the pairings in the RSMP (refer to Steps 10-12 of Algorithm 1). An overview of our column management strategy is provided in Figure 6.

Initially, the pairings produced through the initialization routine in Section 3.2 are added to the RSMP, and the type A and buffer pools are empty. Each run of the MLSP algorithm generally provides us with several pairings with negative reduced costs. A typical trade-off here is between adding several negatively priced columns to the RSMP simultaneously, which may lead to a decreased number of calls to MLSP overall and a rapid growth in the computation time spent by the simplex method to solve the RSMP. We strike a middle ground here and choose to add only one pairing with the most negative

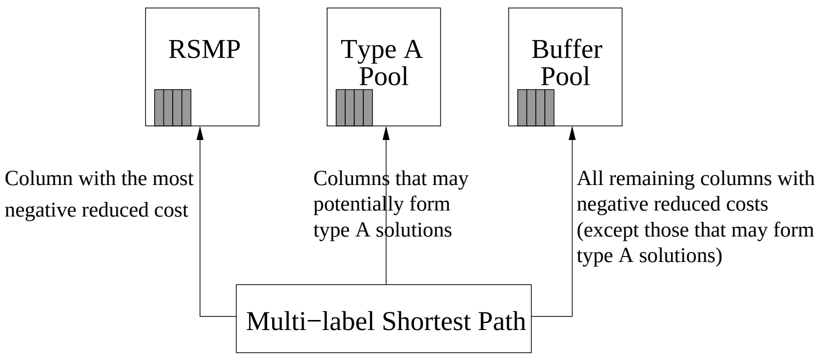

RSMP
Type A Pool
Buffer Pool
Column with the most negative reduced cost
Columns that may potentially form type A solutions
All remaining columns with negative reduced costs (except those that may form type A solutions)
Multi−label Shortest Path

Figure 6: Constructing the column pools.

reduced cost to the RSMP after each execution of the MLSP. However, the remaining columns with negative reduced costs are not discarded but instead stored in the buffer pool, similar to the function of the column pool in [19]. Moreover, any pairing with its final attribute set as A_k^e or A_k^d for some extra flight k is sent to the type A pool regardless of its reduced cost. Note that negatively priced pairings marked for potential type A solutions end up in the type A pool. Now, if the dual values obtained from the RSMP do not change significantly after the next call to the simplex method to solve the RSMP, then one of these stored pairings may still price out favorably. In other words, before we invoke MLSP, we first re-compute the reduced costs of the pairings stored in the buffer and type A pools with respect to the current dual values, and if we identify a pairing with a negative reduced cost, then we save ourselves the computational burden of solving the MLSP in the current iteration. Otherwise, pairings with reduced costs greater than or equal to a fraction of $\bar{c}_{\max}$ are discarded from the buffer pool, where $\bar{c}_{\max}$ denotes the largest reduced cost value over the buffer pool. No pairings are ever deleted from the type A pool in order to create as many opportunities as possible for forming type A solutions once column generation is completed. Also note that the type A pool is redundant if no restriction is imposed on $N_{\max}^{init}$ (this case is denoted by $N_{\max}^{init} = \infty$) in the initialization procedure because all type A solutions are constructed a priori in this case. The steps of our column generation procedure are summarized in Algorithm 2. When the column generation algorithm (Steps 3-9 of Algorithm 1) is completed, we attempt to construct additional type A solutions by pairing up appropriately marked columns in the RSMP and the type A pool. For all feasible type A solutions, RSMP is augmented with additional variables and constraints as necessary and Algorithm 2 is re-invoked afterwards.

Algorithm 2: Column Generation for RSMP - Steps 3-9 of Algorithm 1

```
1:  repeat
2:    Solve RSMP
3:    Calculate the reduced costs of the pairings in the type A and buffer pools
4:    if a negatively priced pairing is identified in the type A or the buffer pools then
5:      Add column to RSMP, remove from the associated pool
6:    else
7:      Delete columns with reduced costs larger than a threshold value from the buffer pool
8:      Solve MLSP and mark pairings for potential type A solutions
9:      if at least one pairing with a negative reduced cost is identified then
10:       Add the column with the minimum reduced cost to RSMP
11:       Add all marked pairings that may potentially form type A solutions to the type A pool
12:       Add all remaining columns with negative reduced costs to the buffer pool
13:     end if
14:   end if
15: until all pairings have non-negative reduced costs or the optimality gap is sufficiently small
```

4. Computational Results.

We obtained three real-life instances from two Turkish airline companies. The first instance is a daily problem with 96 flights, and the other two instances are weekly problems with 135 and 490 flights, respectively. Only one crew base is present in all three instances. One extra

Table 1: Results for the conventional crew pairing problem.

# of Flights	IP OFV	# of Pairings in RMP	# of Selected Pairings	Time (sec.)
96	1626.67	206	48	1.94
135	8383.89	332	74	3.31
490	147819.00	1569	229	28.31

flight is added to the first two instances, and there exist two extra flights in the largest instance.

We conducted our computational experiments on a single core of an HP Compaq DX 7400 computer with a 2.40 GHz Intel Core 2 Quad Q6600 CPU and 3.25 GB of RAM running on Windows XP. Our column generation code is implemented in Visual C++, and IBM ILOG CPLEX 12.1/ Concert Technology 2.9 is employed to solve the linear programs in the column generation procedure. If the optimal solution of the final RSMP is not integral, an integer feasible solution to the original problem is obtained by solving (2)-(11) to optimality over the pairings included in the final RSMP (see Step 14 of Algorithm 1).

In order to determine a baseline cost for our robust model, we first solve the conventional crew pairing problem by column generation for all instances. In Table 1, we report the optimal integer objective function value (the optimal LP solution is integral in all cases), the number of pairings in the restricted master problem and in the solution, and the total solution time in columns 2-5, respectively.

We first run a set of preliminary experiments to determine a reasonable set of choices for α_k and β_k which prescribe the minimum requested number of recovery options for an extra flight $k \in \mathcal{K}$. We also decide on our pruning strategy in the MLSP algorithm. Then, we carry out two sets of experiments. In the first part, we fix the α and β values as determined above and investigate the effects of the parameters $N^{init}_{\max}$ and $N^{A}_{\max}$ both on the pairing solutions and the algorithmic performance. In the second part, we implement a strategy that limits the growth in the number of constraints of the RSMP with little or no effect on solution quality but with significant savings in solution time. We also explore the sensitivity of the solutions to varying α and β values and evaluate the impact of the column pools on the performance of the column generation procedure.

If α_k and β_k are set to large values, then many alternate solutions are provided to the decision maker, possibly at the expense of high pairing costs. However, it may not be possible to satisfy excessively large values of these parameters because the model may not provide that many recovery options. This situation would be detected by positive w^A_k and w^B_k variables in the final solution. Keeping this dichotomy in mind, we conduct a set of preliminary experiments for each extra flight to have a crude estimate of the number of type A and type B recovery solutions that may be obtained. As expected from a problem with a short planning horizon, the number of feasible recovery solutions for the first instance (daily problem with 96 flights) turns out to be quite low. We also observe that there does not exist any type B solution because the short planning horizon, coupled with strict feasibility conditions, rules out the possibility of inserting flights into the pairings. Therefore, for the first instance, we set α_1 and β_1 to 1 and 0, respectively. In contrast, for the instance with 135 flights, the number of type A and type B recovery solutions is extremely high. Consequently, it is computationally very demanding to generate all such solutions. We select $\alpha_1 = 1$ and $\beta_1 = 1$ for the second instance. In the last instance with 490 flights, we cannot identify any type A solution but a set of type B solutions for the first extra flight. However, the number of type A and type B solutions is excessively high for the second extra flight. Thus, initially we use the following parameter values for our largest instance: $\alpha_1 = 0$, $\beta_1 = 1$ and $\alpha_2 = 1$, $\beta_2 = 0$.

In order to determine an effective pruning strategy in the MLSP algorithm, we solve the instance with 135 flights under various settings of the parameters $N^{init}_{\max}$ and $N^{A}_{\max}$ that we also employ later in Table 5. For each setting, we call the column generation algorithm three times with the following algorithmic choices in the MLSP: both rules are turned off, only the exact rule is active, and both pruning rules are used collaboratively. In the last case, MLSP is solved under the exact rule, if no pairing prices out favorably under the approximate rule. The exact pruning rule reduces the solution time (excluding the time for the IP solution) by approximately 3.5% on average over running MLSP with no pruning and the approximate rule provides an additional 3.5% improvement. These figures would grow further with an increase in the number of attributes. In this study, there is a relatively small number of attributes that we keep track of in the MLSP. This leads to the domination of many partial paths, which results in fast exact MLSP executions. In the remainder of our numerical experiments, we always invoke MLSP with both pruning rules.

In the first set of experiments, we assess the impact of the $N^{init}_{\max}$ and $N^{A}_{\max}$, which are the two primary parameters that we use in column management as explained in Section 3.3. Recall that the parameter $N^{init}_{\max}$ is used to restrict the number of partial paths that can accumulate on a node of the flight graph during the initialization procedure. When no restriction is imposed ($N^{init}_{\max} = \infty$), then all feasible type A solutions are generated before starting the column generation procedure. Clearly in this case, the parameter $N^{A}_{\max}$ becomes redundant and an optimal solution to the linear programming relaxation of RCPEF is obtained at the end of column generation. However, since each feasible type A solution adds three linking constraints (6)-(8), the size of the model may become excessively large when $N^{init}_{\max}$ increases. Naturally, this results in an increased computation time to solve RSMP. Therefore, if there is a large number of feasible type A solutions for an extra flight, it is only viable to work with small values of $N^{init}_{\max}$ and then incorporate some of the possible type A solutions during column generation by increasing the value of the parameter $N^{A}_{\max}$. These possible type A solutions are stored in the type A column pool, and they are checked for feasibility when column generation terminates. Nonetheless, this may cause another computational burden because when $N^{A}_{\max}$ is too large, the size of the type A pool becomes massive.

We first attempt to solve RCPEF to optimality by using the parameters $N^{init}_{\max} = \infty$ and $N^{A}_{\max} = 0$. We succeed for the smallest instance with 96 flights, but the initialization routine fails for both of the weekly problem instances due to an exponential growth in the number of paths in the flight network. The results for the daily problem are given in Table 2. In this table, $N^{init}_{\max}$ and $N^{A}_{\max}$ are specified in the first two columns, and the third and fourth columns report the objective function values of the final RSMP and the associated integer program, respectively. The number of pairings in the RSMP and the number of pairings in the solution are given in the next two columns. In the seventh column, the first number represents the number of type A solutions present in the RSMP followed by the number of type A solutions selected in the solution in brackets. Note that the total number of feasible type A solutions for this instance is only two. Columns 8 and 9 denote the number of distinct pairings that cover the extra flight and its associated deadhead in a type A solution, respectively. Column 10 is the number of type B recovery options present in the solution, and the next two columns report the total computation time and the computation time spent while solving RCPEF over the pairings in the final RSMP, respectively. The last column shows the number of times our column generation algorithm is called, where constraints for newly constructed type A solutions are introduced to the RSMP between two successive calls. We observe that the gap between the objective function value of the conventional problem CPP and that of RCPEF is 15%. That is, in this daily problem the connections in the conventional pairing solution are short and forcing a type A recovery option into the solution is rather costly.

The results for the first weekly instance with 135 flights are presented in Table 3. Note that the objective function value of the integer feasible solution is the same for each setting except for the first one and it is 21% larger than the baseline provided by the conventional model. As in the daily problem, the conventional solution features very tight connections and the cost of robustness is high. It is up to the decision maker to determine whether this solution is acceptable and take appropriate action. When $N^{init}_{\max}$ is fixed to 0 and $N^{A}_{\max}$ is increased, the number of feasible type A solutions grows considerably. This increase indicates that the type A pool is very effective in generating type A solutions. When $N^{A}_{\max} = 20$, the size of the model becomes huge because of the large number of existing type A solutions. Therefore, solving the integer program takes a substantial amount of time compared to the other settings. The same interpretation also applies to the cases where $N^{A}_{\max} = 0$ and $N^{init}_{\max}$ is increased gradually. In both cases, the computation time grows proportionally with the number of feasible type A solutions in RSMP. Recall that the number of type A solutions is related to the number of those pairings that can cover either the extra flight or its associated deadhead flight. In Table 3, we observe an extreme case where each pairing that can cover an extra flight can form a type A solution together with every pairing that can cover the deadhead of the same extra flight. Therefore, the number of type A solutions given in the seventh column is simply the product of the number of distinct pairings reported in the eighth and ninth columns. As these figures indicate, the large number of feasible type A solutions are created by the

Table 2: Optimal solution for the daily instance with 96 flights ($\alpha_1 = 1$ and $\beta_1 = 0$).

$N^{init}_{\max}$	$N^A_{\max}$	LP OFV	IP OFV	# of Pairings in RSMP	# of Selected Pairings	# of Type A [Selected]	Distinct Pairings (Extra)	Distinct Pairings (Deadhead)	# of Selected Type B	Time (sec.)	IP Sol. Time (sec.)	# of times Col. Gen. is repeated
∞	0	1814.17	1885.00	236	48	2[1]	1	2	0	2.28	0.16	1

Table 3: Sensitivity to $N^{init}_{\max}$ and $N^A_{\max}$ for the weekly problem with 135 flights ($\alpha_1 = 1$ and $\beta_1 = 1$).

$N^{init}_{\max}$	$N^A_{\max}$	LP OFV	IP OFV	# of Pairings in RSMP	# of Selected Pairings	# of Type A [Selected]	Distinct Pairings (Extra)	Distinct Pairings (Deadhead)	# of Selected Type B	Time (sec.)	IP Sol. Time (sec.)	# of times Col. Gen. is repeated
100	0	9646.64	10181.30	547	67	204[1]	12	17	1	7.43	0.41	1
200	0	9581.55	10181.40	531	65	840[1]	24	35	1	18.16	2.43	1
0	10	9679.07	10181.40	469	62	1380[1]	6	230	1	12.44	2.16	2
0	20	9535.94	10181.40	793	68	16926[1]	42	403	1	1400.77	1158.19	2

Table 4: Sensitivity to $N^{init}_{\max}$ and $N^A_{\max}$ for the weekly problem with 490 flights ($\alpha_1 = 0$, $\beta_1 = 1$, and $\alpha_2 = 1$, $\beta_2 = 0$).

$N^{init}_{\max}$	$N^A_{\max}$	LP OFV	IP OFV	# of Pairings in RSMP	# of Selected Pairings	# of Type A [Selected], $k = 1, 2$	Distinct Pairings (Extra)	Distinct Pairings (Deadhead)	# of Selected Type B, $k = 1, 2$	Time (sec.)	IP Sol. Time (sec.)	# of times Col. Gen. is repeated
0	5	148372.00	149576.00	3263	219	-, 1960[1]	14	140	1, 0	205.05	46.00	2
0	10	148411.00	149594.00	5582	227	-, 990[1]	9	110	1, 0	349.87	15.48	2

combination of a relatively small number of distinct pairings. This surprising fact can be explained by observing that a pairing that can cover an extra flight or its associated deadhead needs a long connection time around this extra flight, which typically appears between two duties. In this case, the extra flight or its deadhead is covered as the first or last flight of a duty. As long as there are no feasibility rules related to the length of pairings, two such pairings covering an extra flight and its deadhead form a type A solution when they are swapped.

Table 4 shows the results for the weekly instance with 490 flights. As we mentioned before, we cannot identify any feasible type A solution for the first extra flight. Therefore, the reported results related to type A solutions for the first extra flight are denoted by '-' in column 7. In column 10, the number of selected type B solutions for the first and the second extra flights is separated by commas. Due to the large size of this problem, we pick $N^{init}_{\max} = 0$ and $N^{A}_{\max} = 5, 10$. In this case, we achieve a robust solution at a relatively small incremental cost. The costs of the integer feasible solutions are 1.1 and 1.2% away from that of the conventional model, respectively. We also observe that the number of type A solutions decreases even though the parameter $N^{A}_{\max}$ is increased. The reason behind this unexpected behavior is as follows: When $N^{A}_{\max}$ is increased from 5 to 10, some of the pairings, which previously yielded type A solutions when $N^{A}_{\max}$ was set to 5, are dominated by the additional pairings that are introduced into the MLSP after increasing $N^{A}_{\max}$ to 10. Finally, as in the previous instance, we observe that the large number of feasible type A solutions are constructed by only a small number of distinct pairings.

In our last set of experiments we restrict the increase in the number of rows of RSMP in an effort to reduce the solution times for RSMP and the integer solution. To this end, our strategy at the termination of a column generation call is to only add type A solutions to the RSMP that are created by promising pairings with smaller reduced costs. Recall that a feasible type A solution is denoted by a pair (p, q), where pairings p and q cover the extra flight and its associated deadhead flight, respectively. For each pairing p that covers an extra flight k, we define a set $\{q|(p, q) \in P^k_A\}$ and sort the elements of this set in non-decreasing order of the reduced costs of pairing q in each pair. We introduce no more than $5\alpha_k$ type A solutions into the RSMP from this set. We test the effects of this strategy for the first weekly instance with 135 flights for each combination of the parameters $N^{init}_{\max} = 0, 100, 200$ and $N^{A}_{\max} = 0, 10, 20$ for $(\alpha_1, \beta_1) = (0, 1), (1, 1), (1, 0), (2, 1)$. Note that $N^{A}_{\max} = 0$ and $N^{init}_{\max} = 0$ are only relevant for $(\alpha_1, \beta_1) = (0, 1)$. For the second weekly instance with 490 flights, $N^{init}_{\max} = 0$ and $N^{A}_{\max} = 0, 5, 10$. Recall that we cannot identify any type A solution for the first extra flight and thus set $(\alpha_1, \beta_1) = (0, 1)$. For the other extra flight, $(\alpha_2, \beta_2) = (0, 1), (1, 1), (1, 0), (2, 1)$. Both $N^{init}_{\max}$ and $N^{A}_{\max}$ are set to zero if $\alpha_1 = \alpha_2 = 0$. The results are presented in Tables 5 and 6. The values of (α_k, β_k) are specified for $k = 1, 2$ in the first column. In the seventh and eighth columns, we report the number of pairings in the RSMP that are taken from the buffer and the type A pools, respectively. These figures attest to the positive effect of using the column pools. Under each parameter setting, a significant number of pairings come from the buffer pool and/or the type A pool, reducing the number of computationally costly calls to MLSP.

Tables 5 and 6 are helpful in illustrating the incremental costs of increasingly robust solutions and also the trade-offs among different recovery options. For instance, in Table 5 the cost of the integer solution for $(\alpha_1, \beta_1) = (0, 1)$ is 9490, while this cost increases to 10181 if in addition one type A solution is desired. Requesting one additional type A solution comes at an additional cost of 713. In Table 6, the cost of the integer solution for $(\alpha_2, \beta_2) = (0, 1)$ is 149259 while opting for a type A solution instead of a type B solution for this extra flight leads to a cost of 149663. Such analyses may be particularly important if a pairing participates in recovery options for different extra flights, and in this case choosing $\alpha_k + \beta_k > 1$ may prove advantageous.

For the instance with 135 flights, we compare the results in rows corresponding to $(\alpha_1, \beta_1) = (1, 1)$ in Table 5 to those in Table 3. Similarly, the results in rows corresponding to $(\alpha_2, \beta_2) = (1, 0)$ in Table 6 are compared to those in Table 4. In both cases, the solution quality is basically identical. However, the number of type A solutions in the RSMP is greatly reduced leading to significant savings in the computation time. Thus, we conclude that including a small subset of the type A solutions in the RSMP that are comprised of pairings with small reduced costs is a viable choice.

As a final remark, we note that the lower bounding mechanism discussed in Section 3.1.2 does not play a role in the convergence of the proposed approach. This is due to the fact that during the course of generating columns, MLSP with the approximate rule is very often successful in identifying pairings that price out favorably and MLSP is then solved exactly only a few times close to optimality.

5. Conclusions and Future Research

For a commercial airline, recovering from disruptions is critical from both cost effectiveness and customer service aspects. Recovery procedures often result in significant deviation from the planned airline operation and incur additional operational costs. The quality of service also suffers due to operational changes made at short notice. Robust airline planning, which aims at reducing the effect of disruptions on the airline performance, is an area that has recently received attention from academicians and practitioners. In this research, we focus on the robust crew pairing problem, where effective recovery can be achieved through a solution approach that explicitly incorporates disruptions and corresponding recovery options at the planning stage. We consider robustness from a particular perspective of managing extra flights. These extra flights do not appear in the regular flight schedule at the planning level but they may be added to the flight schedule during operations.

Due to the size and complexity of real-life instances, the conventional crew pairing problem is already known to be intractable; it is usually solved by a column generation procedure along with a branch-and-price algorithm for optimal solutions or a heuristic method for near-optimal solutions. Our problem is even harder to solve since the column generation procedure requires adding both columns and rows simultaneously to the restricted master problem. We show that this problem can be solved heuristically by enhancing the column generation procedure through effective pruning strategies in the solution of the pricing subproblem and managing the generated pairings to be introduced to the restricted master problem via column pools.

A major challenge within the column generation procedure is solving the pricing subproblem MLSP in order to find a negative reduced cost pairing. In contrast to the conventional column generation procedure, the potential benefit of covering extra flights with a new pairing cannot be evaluated accurately in the pricing subproblem. In other words, the reduced cost of the new column cannot be calculated correctly due to the incomplete set of constraints in the RSMP. Our algorithm overcomes this challenge by using a two-level iterative heuristic approach that fixes the number of constraints in the RSMP at each iteration of the pricing problem, but stores the pairings that can cover the extra flights to be added to the RSMP subsequently. Since the number of paths in the MLSP is exponential in the number of nodes, it is necessary to use effective pruning strategies. In this study, we propose two pruning rules, exact and approximate, which are used in a collaborative manner. Nonetheless, the overall framework provides a heuristic solution to the restricted master problem because we cannot guarantee that all relevant constraints are generated before column generation terminates.

Our computational experiments have demonstrated that in successive iterations there are only minor changes in the data used within the pricing subproblems. Therefore, as a future research we intend to work on algorithmic improvements in order to use the computer memory efficiently for storing the static information from one iteration to the next. Moreover, we have also observed that the underlying structure of the flight network lends itself to parallelization, and hence by using the recent developments in the field of parallel programming, we may obtain performance improvements. This is another line of work that we want to pursue in the future.

The issue of adding both columns and rows simultaneously can be encountered in other large-scale problems that would be solved by column generation. As exemplified by RCPEF, the main challenge in using column generation for these problems is the missing information in the pricing subproblem due to the absence of a set of constraints that can be added only after the newly generated columns are known. The algorithm we propose here can be applied to other problems with the same structure, where newly generated columns lead to the generation of new constraints in the master problem. Our future research also includes developing a methodological framework that targets this general class of problems.

Acknowledgments. This research has been supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 106M472.

Table 5: Sensitivity to α_k, β_k, $N_{\max}^{init}$, and $N_{\max}^{A}$ for the weekly problem with 135 flights (# of type A solutions restricted).

(α_1, β_1)	$N_{\max}^{init}$	$N_{\max}^{A}$	LP OFV	IP OFV	# of Pairings in RSMP	From Buffer Pool	From Type A Pool	# of Selected Pairings	# of Type A [Selected]	# of Selected Type B	Time (sec.)	IP Sol. Time (sec.)	# of times Col. Gen. is repeated
(0,1)	0	0	9490.14	9490.14	386	356	8	73	0	1	5.95	0*	1
(1,1)	100	0	9751.41	10181.40	547	476	0	71	60[1]	1	8.17	0.21	2
(1,1)	200	0	9733.51	10181.40	533	432	0	73	120[1]	1	13.59	0.15	2
(1,1)	0	10	9776.56	10181.40	468	168	69	73	30[1]	1	9.62	0.10	2
(1,1)	100	10	9738.92	10181.40	519	140	40	73	60[1]	1	9.97	0.12	2
(1,1)	200	10	9723.80	10181.40	466	134	32	68	120[1]	1	14.61	0.235	2
(1,1)	0	20	9712.46	10181.40	786	164	95	63	210[1]	1	24.83	0.31	2
(1,1)	100	20	9709.51	10181.20	737	132	42	70	200[1]	1	23.08	0.37	2
(1,1)	200	20	9709.51	10181.30	700	134	43	69	240[1]	1	30.64	0.41	2
(1,0)	100	0	8635.03	9075.14	544	478	0	73	60[1]	0	6.77	0.22	2
(1,0)	200	0	8619.71	9075.14	521	426	0	73	120[1]	0	11.95	0.25	2
(1,0)	0	10	8639.74	9075.14	516	165	58	74	30[1]	0	8.53	0.23	2
(1,0)	100	10	8623.57	9075.04	509	135	33	70	60[1]	0	9.45	0.25	2
(1,0)	200	10	8610.47	9075.14	515	139	36	72	120[1]	0	14.80	0.28	2
(1,0)	0	20	8601.17	9075.14	795	154	99	72	185[1]	0	23.78	0.33	2
(1,0)	100	20	8598.36	9075.14	714	113	41	73	200[1]	0	23.09	0.34	2
(1,0)	200	20	8598.36	9075.04	742	129	37	73	240[1]	0	30.83	0.42	2
(2,1)	100	0	9857.51	10894.50	517	448	0	71	120[2]	1	7.22	0.28	2
(2,1)	200	0	9820.77	10894.50	532	432	0	73	240[2]	1	13.62	0.64	2
(2,1)	0	10	9917.26	10982.00	468	168	69	71	60[2]	1	8.94	0.22	2
(2,1)	100	10	9829.21	10894.50	519	140	40	73	120[2]	1	10.75	0.39	2
(2,1)	200	10	9804.84	10894.50	465	134	32	67	240[2]	1	15.53	0.80	2
(2,1)	0	20	9778.83	10894.50	787	164	95	72	420[2]	1	27.62	1.17	2
(2,1)	100	20	9775.70	10894.50	738	132	42	62	400[2]	1	26.34	1.25	2
(2,1)	200	20	9775.70	10894.50	702	136	43	73	480[2]	1	37.92	5.28	2

*LP solution is integral.

Table 6: Sensitivity to α_k, β_k, $N_{\max}^{init}$, and $N_{\max}^{A}$ for the weekly problem with 490 flights (# of type A solutions restricted).

(α_1,β_1), (α_2,β_2)	$N_{\max}^{init}$	$N_{\max}^{A}$	LP OFV	IP OFV	# of Pairings in RSMP	From Buffer Pool	From Type A Pool	# of Selected Pairings	# of Type A [Selected], $k=1,2$	# of Selected Type B, $k=1,2$	Time (sec.)	IP Sol. Time (sec.)	# of times Col. Gen. is repeated
(0,1),(0,1)	0	0	149259.00	149259.00	1914	1843	16	229	-,0	1,1	49.76	0*	1
(0,1),(1,1)	0	5	149544.00	150508.00	3085	670	228	219	-,24[1]	1,1	154.98	0.70	2
(0,1),(1,1)	0	10	149441.00	151341.00	6839	861	396	220	-,285[1]	1,1	380.81	10.87	2
(0,1),(1,0)	0	5	148524.00	149576.00	2570	782	282	230	-,70[1]	1,0	143.58	0.91	2
(0,1),(1,0)	0	10	148554.00	149663.00	5581	721	343	220	-,45[1]	1,0	342.64	1.25	2
(0,1),(2,1)	0	5	149563.00	151399.00	3059	675	228	230	-,80[2]	1,1	163.92	3.33	2
(0,1),(2,1)	0	10	149497.00	152108.00	6928	851	396	222	-,570[2]	1,1	382.00	14.56	2

*LP solution is integral.

References

[1] R. Anbil, J.J. Forrest, and W.R. Pulleyblank. Column generation and the airline crew pairing problem. *Documenta Mathematica, Journal der Deutschen Mathematiker Vereinigung*, pages 677–686, 1998.

[2] C. Barnhart, A.M. Cohn, E.L. Johnson, D. Klabjan, G.L. Nemhauser, and P.H. Vance. Airline crew scheduling. In R.W. Hall, editor, *Handbook of Transportation Science*, pages 517–560. Springer, 2003.

[3] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[4] R. Bixby, J. Gregory, I. Lustig, R. Marsten, and D. Shanno. Very large-scale linear programming: A case study in combining interior point and simplex methods. *Operations Research*, 40:885–897, 1992.

[5] K. Bulbul, P. Kaminsky, and C. Yano. Flow shop scheduling with earliness, tardiness, and intermediate inventory holding costs. *Naval Research Logistics*, 51(3):407–445, 2004.

[6] J. Clausen, A. Larsen, J. Larsen, and N.J. Rezanova. Disruption management in the airline industry-concepts, models and methods. *Computers & Operations Research*, 37(5):809–821, 2010.

[7] T.G. Crainic and J.M. Rousseau. The column generation principle and the airline crew scheduling problem. *Infor*, 25(2):136–151, 1987.

[8] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Handbooks in Operations Research and Management Science: Network routing*, pages 35–139. Elsevier, October 1995.

[9] J. Desrosiers and M.E. Lübbecke. A primer in column generation. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, GERAD 25th Anniversary Series, pages 1–32. Springer, 2005.

[10] M. Ehrgott and D.M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11:139–150, 2002.

[11] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, GERAD 25th Anniversary Series, pages 33–65. Springer, 2005.

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

[13] D. Klabjan. Large-scale models in the airline industry. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, GERAD 25th Anniversary Series, pages 163–195. Springer, 2005.

[14] S. Lan, J-P. Clarke, and C. Barnhart. Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28, 2006.

[15] L. Lettovský, E.L. Johnson, and G.L. Nemhauser. Airline crew recovery. *Transportation Science*, 34(4):337–348, November 2000.

[16] M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2002.

[17] A. Makri and D. Klabjan. A new pricing scheme for airline crew scheduling. *Informs Journal on Computing*, 16(1):56–67, 2004.

[18] J.M. Rosenberger, E.L. Johnson, and G.L. Nemhauser. A robust fleet-assignment model with hub isolation and short cycles. *Transportation Science*, 38(3):357–368, 2004.

[19] M. Savelsbergh and M. Sol. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490, 1998.

[20] A.J. Schaefer, E.L. Johnson, A.J. Kleywegt, and G.L. Nemhauser. Airline crew scheduling under uncertainty. *Transportation Science*, 39(3):340–348, August 2005.

[21] S. Shebalov and D. Klabjan. Robust airline crew pairing: Move-up crews. *Transportation Science*, 40(3):300–312, August 2006.

[22] B.C. Smith and E.L. Johnson. Robust airline fleet assignment: Imposing station purity using station decomposition. *Transportation Science*, 40(4):497–516, 2006.

[23] H. Tekiner, Ş.İ. Birbil, and K. Bülbül. Robust crew pairing for managing extra flights. *Computers & Operations Research*, 36:2031–2048, 2009.

[24] J.M. Valério De Carvalho. A note on branch-and-price algorithms for the one-dimensional cutting stock problems. *Computational Optimization and Applications*, 21(3):339–340, 2002.

[25] P.H. Vance, A. Atamturk, C. Barnhart, E. Gelman, E.L. Johnson, A. Krishna, D. Mahidhara, G.L. Nemhauser, and R. Rebello. A heuristic branch-and-price approach for the airline crew pairing problem. Technical Report TLI/LEC-97-06, Georgia Institute of Technology, Atlanta, GA, June 1997. http://www.crewingsolutions.com/docs/branch&price%20ACPO%20Vance%20lec9706.pdf.

[26] P.H. Vance, C. Barnhart, E.L. Johnson, and G.L. Nemhauser. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, 3(2):111–130, 1994.

[27] P.H. Vance, C. Barnhart, E.L. Johnson, and G.L. Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200, March-April 1997.

[28] F. Vanderbeck and L.A. Wolsey. An exact algorithm for IP column generation. *Operations Research Letters*, 19(4):151–159, 1996.

[29] O. Weide, D. Ryan, and M. Ehrgott. An iterative approach to robust and integrated aircraft routing and crew scheduling. *Computers & Operations Research*, 37(5):833–844, 2010.

[30] J.W. Yen and J.R. Birge. A stochastic programming approach to the airline crew scheduling problem. *Transportation Science*, 40(1):3–14, February 2006.