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

Manufacturing Systems & Industrial Engineering
SABANCI UNIVERSITY Faculty of Engineering and Natural Sciences
http://fens.sabanciuniv.edu/msie
November 30, 2009

Solving A Robust Airline Crew Pairing Problem With Column Generation

İbrahim Muter, Ş. İlker Birbil, Kerem Bülbül, Güvenç Şahin, Hüsnü Yenigün
Sabancı University, 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 a local airline 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 is one of the most widely studied 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 to a single feasible schedule, let alone an optimal one under complicated cost structures.

Due to its complexity, the crew scheduling problem is tackled in two sequential phases both in practice and 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 [9].

A nonstop flight is called as 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 total flying time in a duty period is restricted by the *maximum flying time*. 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*, if the first duty period starts and the last duty period ends at a station designated as a *crew base*. 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 typically 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 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, using a set-covering type formulation, CPP is modeled as

$$\begin{aligned} \min \quad & \textstyle\sum_{p\in\mathcal{P}} c_p y_p \\ \text{s.t.} \quad & \textstyle\sum_{p\in\mathcal{P}} a_{ip} y_p \geq 1, \quad i \in \mathcal{F}, \\ & y_p \in \{0,1\}, \end{aligned} \tag{1}$$

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 constraint ensures that each flight leg is covered by at least one pairing. Note that if a flight is covered by multiple pairings then all but one of the crews assigned to this flight are deadheading. 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). The interested reader is referred to [1] for an excellent overview 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. On one hand, the brand image of an airline and the quality of service to the passengers depends much on how quickly and successfully the airline responds to such 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 as a priority by many 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 which 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 which 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 low-cost 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.

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 smaller local airlines in Turkey that sometimes have to add extra flights to their schedule at short notice, e.g., charter flights. 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 the 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. The reader is referred to [9] 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 [9], the authors introduce how an extra flight may be accommodated by modifying the existing pairings and formulate a set of integer programming models that provide natural recovery options without disrupting the existing flights. Their approach requires that all possible crew pairings are available and is only viable for relatively small problem instances. The proposed models in that work are solved by standard commercial solvers. In other words, in [9] 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. We point out further specific differences between these two studies in the next section.

2. Robust Crew Pairing Model. The complexity of the strategic, tactical and operational issues facing the airlines leads to theoretically and computationally challenging problems in operations research. The intractability of monolithic formulations that integrate all aspects of airline planning and operations has prompted the airlines to apply hierarchical frameworks to their planning cycles. In airline schedule planning, a well established cycle consists of flight schedule design, fleet assignment, aircraft routing, and crew scheduling, where the last one is comprised of the crew pairing and crew assignment phases as discussed in Section 1 (see [1] for more details). Various models have been designed and many different algorithms have been proposed for each of the components of this important tactical airline planning problem in the last few decades. The advances in the operations research techniques during this period of time and the leap in the speed of computers over the last two decades currently allows us to identify very good solutions to each individual problem. Clearly, this decomposition approach has one major disadvantage: the quality of the combined solution for the integrated overall problem is very hard to assess. Furthermore, another intrinsic drawback of this framework is the isolation of the planning cycle from operations. Airline operations are subject to several different types of disruptions as discussed in Section 1. In other words, plans are updated continuously due to irregular operations, and these updates are rarely carried out with an objective of minimizing costs in mind. Generally, the primary objective of recovery procedures is to find a feasible solution as soon as possible, and the adverse effect of disruptions and subsequent recovery actions on planned costs is unclear at the time of planning. The main goal of this paper is to take a step forward in closing this gap by focusing on a particular source of disruption and its impact on the crew pairing problem. To this end, we incorporate a specific concept of robustness into crew pairing generation and minimize the negative consequences of disruptions. A fundamental premise of our framework is that it provides natural paths to recovery for the particular type disruption under consideration and obviates the need for an explicit recovery process at the time of operation that pertains to crew pairing. Thus, the potential costs of disruptions and recovery are transparent during the planning phase leading to better decision making. While the sole source of disruptions in this paper is a set of extra flights added to the flight schedule during operation, the established framework may be generalized to other types of disruptions. This may be accomplished by defining the desirable properties of the pairings that facilitate recovery at a low cost for the specific disruptions under consideration and promoting such pairings either in the objective function or the constraints of the mathematical model as demonstrated in this section.

The main contribution of the current work is introducing a column generation algorithm that can handle the robust integer programming model proposed for RCPEF below. This model poses an interesting theoretical challenge (see Section 3) and is not amenable to a traditional column generation algorithm designed for the conventional crew pairing problem. We emphasize that in [9] the authors generate all possible crew pairings explicitly and solve the proposed integer programs by a commercial solver. This approach is clearly not computationally feasible for large crew pairing instances, and in this work we present our algorithms and results for large instances of RCPEF. Furthermore, the approach presented in [9] is a two-step approach. In the first phase, the conventional crew pairing problem is solved and the corresponding optimal objective value is obtained. Next, several robust models are formulated that promote pairings with certain properties. Such pairings facilitate the recovery process, if potential extra flights are actually inserted into the schedule at the time of operation. In these robust models, 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, i.e., 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. In the discussion below, we first introduce the types of pairings that yield natural recovery paths to recovery for potential extra flights and then formulate RCPEF as an integer program with exponentially many variables.

Several recovery options are explored in [9] for managing potential extra flights, which are classified 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.

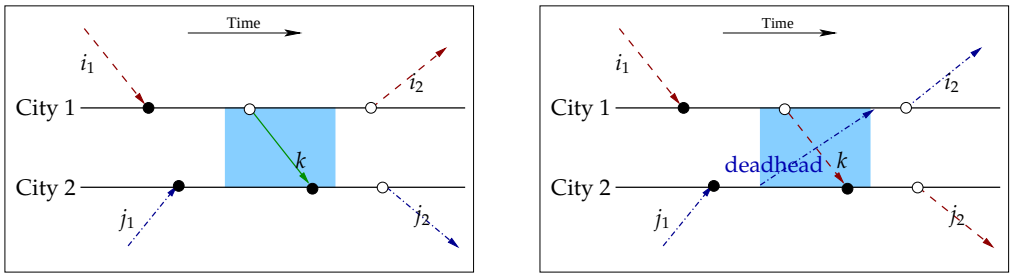
In this work, we restrict ourselves to the type A and type B solutions as illustrated in Figures 1 and 2, where the estimated time window of extra flight k is depicted by the shaded rectangles. The left and right end points of the estimated time window mark the earliest possible departure and latest 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 [10].

In Figure 1, the original pairing p covering the flight legs i_1, 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 [9] 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 [9] 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 [9]. 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 appears to be quite difficult to keep track of the other two type A solutions in the pricing subproblem in the column generation algorithm.

The proposed robust mathematical model for RCPEF is given below:

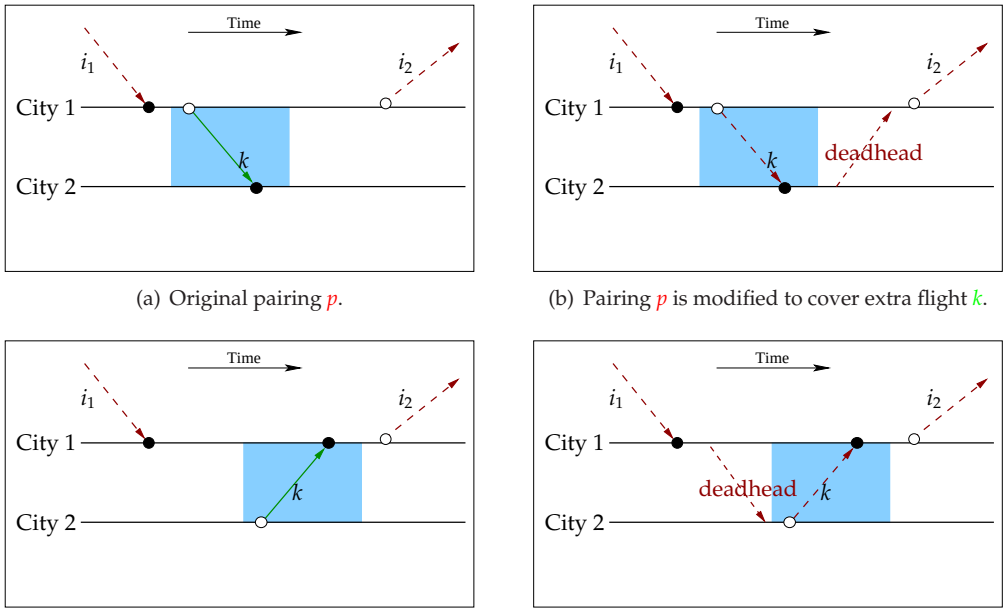
$$\min \sum_{p \in \mathcal{P}} c_p y_p + \sum_{i \in \mathcal{F}} (d_i s_i^+) \tag{2}$$



(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).



(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).

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

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

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

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

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

$$y_q \geq x^k_{(p,q)}, \qquad q \in \mathcal{P} : (p,q) \in \mathcal{P}_A(k), k \in \mathcal{K}, \tag{8}$$

$$y_p \in \{0,1\}, \qquad p \in \mathcal{P}, \tag{9}$$

$$x^k_{(p,q)} \in \{0,1\}, \qquad (p,q) \in \mathcal{P}_A(k), k \in \mathcal{K}, \tag{10}$$

$$s_i^+ \in \mathcal{Z}_+, \qquad i \in \mathcal{F}. \tag{11}$$

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 zero; 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 optimal solution, respectively. If pairing p is selected, the decision variable y_p takes the value 1 and zero; 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 zero; 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. 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 only if $y_p = y_q = 1$ and is set to zero; otherwise. The formulation (2)-(11) has exponentially many variables, one for each pairing. 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 (1) formulation 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 [6] for a review 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 an 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, etc. 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 amount of time. In Section 3.1, we shall cover the details of this difficult pricing problem. The column generation algorithm terminates when the MLSP yields no pairing with a negative reduced cost.

The formulation (2)-(11) for the RCPEF incorporates exponentially many variables, and poses an additional complexity over the CPP. 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 $x^k_{(p,q)}$ which depends on the 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) which is stated below:

$$\min \quad \sum_{p\in\bar{\mathcal{P}}} c_p y_p + \sum_{i\in\mathcal{F}} (d_i s^+_i + \sigma s^-_i) + \sum_{k\in\mathcal{K}} \tau(w^A_k + w^B_k) \tag{12}$$

$$\text{s.t.} \tag{13}$$

$$(u_i) \quad \sum_{p\in\bar{\mathcal{P}}} a_{ip} y_p - s^+_i + s^-_i = 1, \qquad i \in \mathcal{F}, \tag{14}$$

$$(z_k^A) \quad \sum_{(p,q)\in\bar{\mathcal{P}}_A(k)} x^k_{(p,q)} + w^A_k \geq \alpha_k, \qquad k \in \mathcal{K}, \tag{15}$$

$$(z_k^B) \quad \sum_{p\in\bar{\mathcal{P}}_B(k)} y_p + w^B_k \geq \beta_k, \qquad k \in \mathcal{K}, \tag{16}$$

$$(\delta^1_{(p,q),k}) \quad y_p + y_q - x^k_{(p,q)} \leq 1, \qquad (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{17}$$

$$(\delta^2_{(p,q),k}) \quad y_p - x^k_{(p,q)} \geq 0, \qquad p \in \bar{\mathcal{P}} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{18}$$

$$(\delta^3_{(p,q),k}) \quad y_q - x^k_{(p,q)} \geq 0, \qquad q \in \bar{\mathcal{P}} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{19}$$

$$(\gamma^y_p) \quad y_p \leq 1, \qquad p \in \bar{\mathcal{P}}, \tag{20}$$

$$(\gamma^x_{(p,q),k}) \quad x^k_{(p,q)} \leq 1, \qquad (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{21}$$

$$y_p \geq 0, \qquad p \in \bar{\mathcal{P}}, \tag{22}$$

$$x^k_{(p,q)} \geq 0, \qquad (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \tag{23}$$

$$s^+_i, s^-_i \geq 0, \qquad i \in \mathcal{F}, \tag{24}$$

$$w^A_k, w^B_k \geq 0, \qquad i \in \mathcal{K}. \tag{25}$$

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)-(5) in RCPEF by including the variables w^A_k and w^B_k, $k \in \mathcal{K}$, on the left hand sides of the constraints (15) and (16), 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 (14) and to advocate feasibility, we set the objective coefficients of these variables to a large value σ. Also note that the dual variables appear in parentheses to the left of their associated constraints in RSMP. We will need these in Section 3.1 while studying the pricing subproblem in detail.

The challenge of solving (12)-(25) is rooted in our lack of ability to identify all constraints (17)-(19). 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 (12)-(25) optimally. 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 (17)-(19) in RSMP during each iteration and apply column generation to optimality with these fixed constraints. While we apply column generation, the candidate 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.

The main objective of the initialization procedure in Step 1 in 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 3 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 do 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

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 (17)-(19), 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 ϕ times.
14: Apply a primal heuristic method if column generation terminates with a fractional solution.

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 $\phi - 1$ times. Otherwise, we proceed to the final step and obtain an integer feasible solution for RCPEF in Step 14 by a primal heuristic method if the column generation provides us with a fractional solution at termination. We relegate the details of our primal heuristic to our computational study in Section 4. Note that the proposed approach is not necessarily 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 shall need 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 unpromising 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 Sections 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)-(25) 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 the SMP and the 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 formulation

DSMP of SMP stated in (26)-(39). DSMP is also required in Section 3.1.2 when we develop an optimality gap on the objective value of the current RSMP .

$$\max \sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}} (\alpha_k z_k^A + \beta_k z_k^B) + \sum_{p\in\mathcal{P}} \gamma_p^y + \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \gamma^x_{(p,q),k}) \tag{26}$$

$$\text{s.t} \sum_{i\in\mathcal{F}} a_{ip} u_i + \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B + \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \delta^2_{(p,q),k}) + \sum_{k\in\mathcal{K}} \sum_{\{(q,p):(q,p)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(q,p),k} + \delta^3_{(q,p),k}) + \gamma_p^y \le c_p, \quad p\in\mathcal{P}, \tag{27}$$

$$z_A^k - \delta^1_{(p,q),k} - \delta^2_{(p,q),k} - \delta^3_{(p,q),k} + \gamma^x_{(p,q),k} \le 0, \quad (p,q)\in\bar{\mathcal{P}}_A(k), k\in\mathcal{K}, \tag{28}$$

$$u_i \ge -d_i, \quad i\in\mathcal{F}, \tag{29}$$

$$u_i \le \sigma, \quad i\in\mathcal{F}, \tag{30}$$

$$z_k^A \le \tau, \quad k\in\mathcal{K}, \tag{31}$$

$$z_k^B \le \tau, \quad k\in\mathcal{K}, \tag{32}$$

$$u_i \text{ unrestricted}, \quad i\in\mathcal{F}, \tag{33}$$

$$z_k^A, z_k^B \ge 0, \quad k\in\mathcal{K}, \tag{34}$$

$$\delta^1_{(p,q),k} \le 0, \quad (p,q)\in\bar{\mathcal{P}}_A(k), k\in\mathcal{K}, \tag{35}$$

$$\delta^2_{(p,q),k} \ge 0, \quad p\in\mathcal{P}:(p,q)\in\bar{\mathcal{P}}_A(k), k\in\mathcal{K}, \tag{36}$$

$$\delta^3_{(p,q),k} \ge 0, \quad q\in\mathcal{P}:(p,q)\in\bar{\mathcal{P}}_A(k), k\in\mathcal{K}, \tag{37}$$

$$\gamma_p^y \le 0, \quad p\in\mathcal{P}, \tag{38}$$

$$\gamma^x_{(p,q),k}, \le 0, \quad (p,q)\in\bar{\mathcal{P}}_A(k), k\in\mathcal{K}. \tag{39}$$

The optimal dual variables provided by the current RSMP may violate one or several of the constraints (27) 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

$$\begin{aligned}\bar{c}_p &= c_p - \sum_{i\in\mathcal{F}} a_{ip}u_i - \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B \\ &\quad - \sum_{k\in\mathcal{K}} \left(\sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \delta^2_{(p,q),k}) + \sum_{\{(q,p):(q,p)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(q,p),k} + \delta^3_{(q,p),k}) \right) - \gamma_p^y \\ &= c_p - \sum_{i\in\mathcal{F}} a_{ip}u_i - \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B.\end{aligned} \tag{40}$$

Thus, the objective of the pricing subproblem is to determine at least pairing p with $\bar{c}_p < 0$. The last two terms in the first equality in (40) 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 all flights appear in at least one pairing and all extra flights are covered by at least one type B solution, respectively, early in the column generation procedure.

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 [10] for the duty network representation.

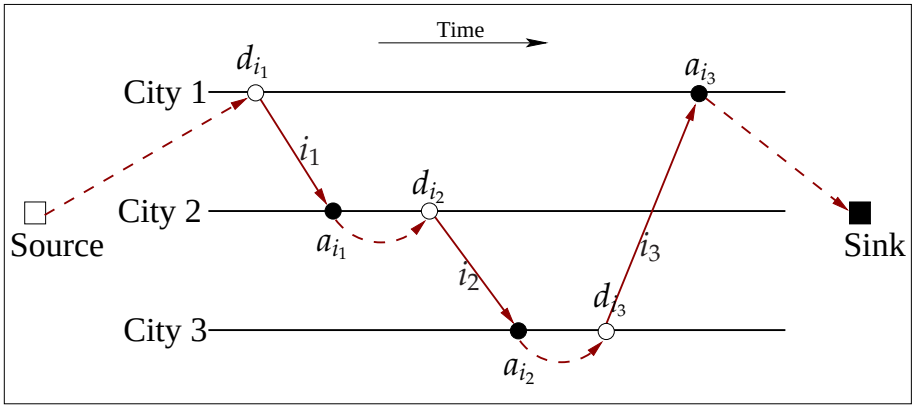


Figure 3: A flight network with crew base City 1.

Formally, for a given set of flights $\mathcal{F}$ our flight network $G = (V, E)$ is constructed as described below. 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 of arcs E in G 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 $E_F \cup E_C \cup E_B\}$. The cost of a pairing corresponding to a path that uses an arc $e \in E_0$ also incorporates the cost of deadheading along e.

Observe that the flight network G is a directed acyclic graph, and without loss of generality we 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 which range from the lower and upper bounds on the connection times between two consecutive flights, to limits on the maximum flying time in a duty period, and to the maximum elapsed time of a pairing, etc. Moreover, frequently the cost structure of a pairing is non-linear, i.e., 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. The interested reader is referred to [1] and [5] for thorough reviews of the different cost structures and feasibility rules in use which are highly dependent on the airlines and governing aviation agencies. Therefore, while 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. Each attribute is referred to as a *label*, and in this work we consider a set of $L = 10$ labels for ensuring feasibility and computing cost. The labels l_1 through l_{L-1} are all commonly used in crew pairing problems: the total elapsed time or the time-away-from-base (l_1), the number of completed duty periods (l_2), the sum of the costs of the completed duty periods (l_3), the total number of flights covered (l_4), the sum of the dual values of the flights covered (l_5), the total elapsed time of the current duty period (l_6), the total flying time within the current duty period (l_7), the total number of flights covered by the current duty period (l_8), and the cost of the current duty period (l_9). We designate the final label l_L 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 label l_{10} are $\{A_k^e, A_k^d, B_k \mid k \in \mathcal{K}\} \cup \{none\}$, where

- ◇ *none* means that the partial pairing does not provide a type A or B solution for any extra flight,
- ◇ A_k^e 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_k^d 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 labels $l^p = \langle l_1^p, l_2^p, ..., l_L^p \rangle$, 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. 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 and is computed as $l_3^p + l_9^p$ 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 $\bar{c}_p = l_3^p + l_9^p - l_5^p = l_3^p - l_5^p$, where $l_9^p = 0$ for any such path because a feasible pairing is constituted by completed duty periods only. A pairing with the most negative reduced cost is added to the RSMP.

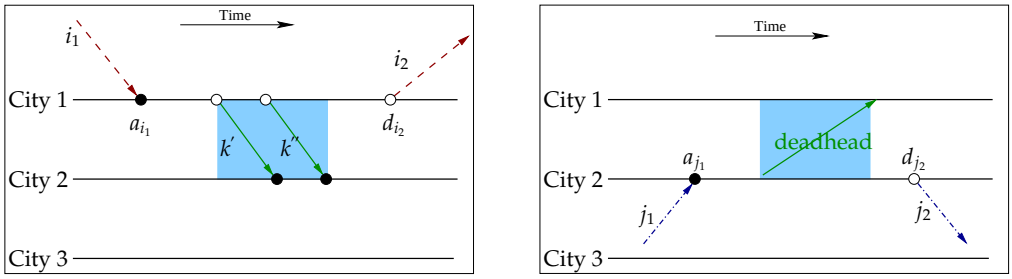
In our multi-label shortest path algorithm adapted from the algorithms by Desrosiers *et. al.* [4], the set of partial paths that terminate at $v \in V$ is denoted by $\mathcal{L}^v$, where the cardinality of this set is bounded from above by the number of all possible paths from the *source* node to 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 $v \in V$. If $l^p \preceq_d l^q$, then p is said to dominate q. In the context of this work, we state that $l^p \preceq_d l^q$, if $l_3^p + l_9^p \leq l_3^q + l_9^q$, $l_5^p \geq l_5^q$, and $l_{10}^p = l_{10}^q$.*

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 which 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 labels are identical, two partial pairings cannot dominate each other. In other words, the final label 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 labels l_{10} to one of the appropriate values $\{A_k^e, A_k^d, B_k \mid k \in \mathcal{K}\}$. Otherwise, the default value for l_{10} is *none* for all partial paths. In general, we allow a partial path to be marked several times for different extra flights. However, in the following discussion we limit ourselves to a single extra flight for the clarity of the presentation. The rules that we verify for type A solutions is 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 either a feasible sit or rest connection, regardless of where k is located in its estimated time window, then $l_{10}^p = A_k^e$. This is depicted in Figure 4(a) where k' and k'' denote the earliest and latest

possible extra flights 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 $l^q_{10} = A^d_k$. This is illustrated in Figure 4(b). Note that the limits on sit and rest times before and after deadhead flights may be different than those for regular flights. 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^d_k 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 (17)-(19).



(a) l^p_{10} is set to A^e_k for partial pairing p.

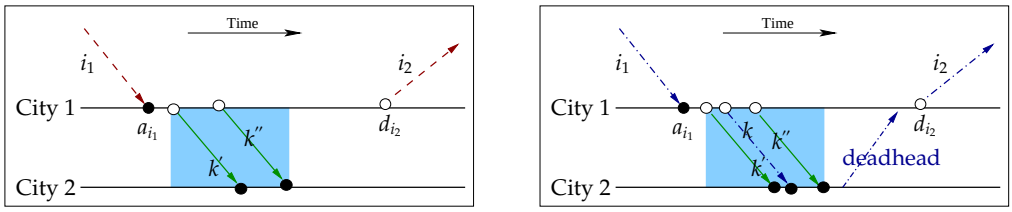
(b) l^q_{10} is set to A^d_k for partial pairing q.

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

The procedure described above does also shed light into 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 complement pairing q determines whether pairing p participates in a type A solution or not, and keeping track of such cross-effects seems impossible from a computational point of view 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 $l^q_{10} = B_k$. 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.

Intuitively, pairings that form type A or B solutions have longer connection times; that is, 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 $p = \langle \ldots, l^p_3, \ldots, l^p_5, \ldots, l^p_9, A^e_k \rangle$ and $p = \langle \ldots, l^p_3, \ldots, l^p_5, \ldots, l^p_9, A^d_k \rangle$. Such a partial pairing p incurs a relatively higher cost as reflected by the sum $l^p_3 + l^p_9$, but the sum of the dual values l^p_5 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 label. However, this approach creates a computational challenge. In preliminary experiments, we observed that a very large number of partial pairings with



(a) Original partial pairing p, l^p_{10} is set to *none*.

(b) Partial pairing q obtained from p, l^q_{10} is set to B_k.

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

high costs may be carried all the way to the *sink* node, slowing down our algorithms substantially and providing no or very slim benefits, if we keep all non-dominated partial paths according to Definition 3.1. 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 $N^A_{\max}$ non-dominated partial pairings at a node $v \in V$ during MLSP whose final labels are either set to A^e_k or A^d_k for some extra flight k. The sensitivity of our solution quality and the computational time to this parameter is explored in our computational study 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^q_5 incorporates the dual value of constraint (16) 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 the fact that there may be exponentially many paths from the *source* node to a node $v \in V$. Even though not all of these paths need to be computed thanks to the feasibility rules and the domination between partial paths based on the ordering relation $\preceq_d$ (see Definition 3.1), in practice still a huge number of feasible partial paths that are mutually incomparable by $\preceq_d$ may accumulate on a node 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 [7] in terms of the underlying intuition of the rules and their types. Like in [7] 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, it means that 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 simple 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, that might be due to an overpruning 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 below

$$\begin{aligned} \bar{c} &= \frac{\sum_{p\in\bar{\mathcal{P}}} c_p}{\sum_{p\in\bar{\mathcal{P}}} |p|}, \\ \lambda &= \max_{p\in\bar{\mathcal{P}}}\{|p|\}, \end{aligned} \tag{41}$$

where $|p|$ is 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 pairings

generated so far as

$$\begin{aligned} \bar{u} &= \frac{\sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{F}} a_{ip}u_i}{\sum_{p\in\mathcal{P}}|p|}, \text{ and} \\ \bar{z} &= \frac{\sum_{k\in\mathcal{K}} z_k^B|\bar{\mathcal{P}}_B(k)|}{\sum_{p\in\mathcal{P}}|p|}, \end{aligned} \tag{42}$$

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 in Equation (43) by using the average values given in Equations (41) and (42).

$$\tilde{e}_p = \underbrace{\left(l_3^p + l_9^p - l_5^p\right)}_{\text{known}} + \underbrace{(\lambda - |p|)(\bar{c} - \bar{u} - \bar{z})}_{\text{estimated}} \tag{43}$$

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; 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 keep only 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. We denote the length of the shortest and longest paths from a node $v \in V$ to the *sink* node in a flight network $G'(V, E \cup \{(d_k, a_k) | k \in \mathcal{K}\})$ by δ_v and Δ_v, respectively. G' is obtained from the original flight network G by inserting the flight arcs for the extra flights. In the shortest path problem, the arc costs are the flight, connection, and deadheading costs. In the longest path problem, arcs $e \in E_C \cup E_B \cup E_0$ incur no cost. The cost assigned to a flight arc $(d_i, a_i) \in E_F$ is u_i and the cost assigned to an extra flight arc $(d_k, a_k), k \in \mathcal{K}$, is z_k^B, where $u_i, i \in \mathcal{F}$, and $z_k^B, k \in \mathcal{K}$, are the optimal dual values from the current RSMP associated with the constraints (14) and (16), respectively. Note that δ_v is the same for all iterations of MLSP since the flight, connection, and deadheading costs do not change. However, 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 as

$$e_p = l_3^p + l_9^p - l_5^p, \quad p \in \mathcal{L}^v. \tag{44}$$

The following claim explains 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 $e_p \geq \Delta_v - \delta_v$, then the reduced cost of q is non–negative.*

PROOF. Suppose r is the path from the node v to the *sink* node such that when r is appended to p we get q. Let c_r be the total cost of r and d_r be the sum of the dual values of the regular and extra flights covered in r. By using the definitions of δ_v and Δ_v given above we have

$$\begin{aligned} c_r &\geq \delta_v, \\ d_r &\leq \Delta_v. \end{aligned} \tag{45}$$

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

$$c_q = l_3^q + l_9^q = l_3^p + l_9^p + c_r \geq l_3^p + l_9^p + \delta_v. \tag{46}$$

The following expression calculates an upper bound on the sum of the dual values of the regular flights covered in q and the sum of the dual values associated with the extra flights for which q provides a type B solution:

$$\sum_{i\in\mathcal{F}} a_{iq}u_i + \sum_{\{k\in\mathcal{K}: q\in\mathcal{P}_B(k)\}} z_k^B = l_5^q = l_5^p + r_d \leq l_5^p + \Delta_v. \tag{47}$$

By using Equations (40), (46), and (47), we have the following chain of reasoning for the reduced cost $\bar{c}_q$ of q:

$$\begin{aligned} \bar{c}_q &= c_q - \sum_{i\in\mathcal{F}} a_{iq}u_i - \sum_{\{k\in\mathcal{K}:q\in\mathcal{P}_B(k)\}} z_k^B \\ &\geq (l_3^p + l_9^p + \delta_v) - (l_5^p + \Delta_v) \\ &= (l_3^p + l_9^p - l_5^p) - (\Delta_v - \delta_v) \\ &= e_p - (\Delta_v - \delta_v). \end{aligned} \tag{48}$$

Finally by using the premise of the proposition and Equation (48), we can deduce $\bar{c}_q \geq 0$. □

Intuitively, δ_v and Δ_v give 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.

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 [2] for a discussion of this topic. 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 number of pairings are 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.

The optimal objective value z_{RSMP} of the current restricted 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 start 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 value of the MLSP which yields the maximum infeasibility over all constraints in DSMP. An optimality gap is then computed by employing the objective value of this dual feasible solution. The formulation (49)-(52) below is equivalent to DSMP and is obtained from (26)-(39) by applying the transformation $u_i = u'_i - d_i, i \in \mathcal{F}$. Our motivation for this transformation will be clear later in this section.

$$\max \sum_{i\in\mathcal{F}} u'_i + \sum_{k\in\mathcal{K}} (\alpha_k z_k^A + \beta_k z_k^B) + \sum_{p\in\mathcal{P}} \gamma_p^y + \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \gamma^x_{(p,q),k}) - \sum_{i\in\mathcal{F}} d_i \tag{49}$$

$$\text{s.t} \sum_{i\in\mathcal{F}} a_{ip}u'_i + \sum_{\{k\in\mathcal{K}:p\in\mathcal{P}_B(k)\}} z_k^B + \sum_{k\in\mathcal{K}} \sum_{\{(p,q):(p,q)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(p,q),k} + \delta^2_{(p,q),k}) + \sum_{k\in\mathcal{K}} \sum_{\{(q,p):(q,p)\in\bar{\mathcal{P}}_A(k)\}} (\delta^1_{(q,p),k} + \delta^3_{(q,p),k}) + \gamma_p^y \leq c_p + \sum_{i\in\mathcal{F}} a_{ip}d_i, \qquad p \in \mathcal{P}, \tag{50}$$

$$(28), (31), (32), (34) - (39),$$

$$u'_i \leq \sigma + d_i, \qquad i \in \mathcal{F}, \tag{51}$$

$$u'_i \geq 0, \qquad i \in \mathcal{F}. \tag{52}$$

Now, assume that the optimal dual values corresponding to constraints (14)-(21) 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 (40), we define a score value [3] $\bar{s}_p$ based on (50):

$$\bar{s}_p = \frac{c_p + \sum_{i\in\mathcal{F}} a_{ip} d_i}{\sum_{i\in\mathcal{F}} a_{ip} u'_i + \sum_{\{k\in\mathcal{K}: p\in\mathcal{P}_B(k)\}} z^B_k} = \frac{c_p + \sum_{i\in\mathcal{F}} a_{ip} d_i}{\sum_{i\in\mathcal{F}} a_{ip} u_i + \sum_{\{k\in\mathcal{K}: p\in\mathcal{P}_B(k)\}} z^B_k + \sum_{i\in\mathcal{F}} a_{ip} d_i}, \tag{53}$$

where $0 < \bar{s}_p < 1$. Computing $\bar{s}_{\min} = \min_{\{p\in\mathcal{P}: \bar{c}_p<0\}} \bar{s}_p$, where $0 < \bar{s}_{\min} < 1$, we can construct a feasible solution for (49)-(52):

$$\begin{array}{ll}
\hat{u}_i = \bar{s}_{\min} u'_i = \bar{s}_{\min}(u_i + d_i), & i \in \mathcal{F}, \\
\hat{z}^A_k = z^A_k, & k \in \mathcal{K}, \\
\hat{z}^B_k = \bar{s}_{\min} z^B_k, & k \in \mathcal{K}, \\
\hat{\delta}^1_{(p,q),k} = \delta^1_{(p,q),k}, & (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \\
\hat{\delta}^2_{(p,q),k} = \delta^2_{(p,q),k}, & p \in \mathcal{P} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \\
\hat{\delta}^3_{(p,q),k} = \delta^3_{(p,q),k}, & q \in \mathcal{P} : (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K}, \\
\hat{\gamma}^y_p = \gamma^y_p, & p \in \mathcal{P}, \\
\hat{\gamma}^x_{(p,q),k} = \gamma^x_{(p,q),k}, & (p,q) \in \bar{\mathcal{P}}_A(k), k \in \mathcal{K},
\end{array} \tag{54}$$

where $0 \leq \hat{u}_i \leq u'_i = (u_i + d_i), i \in \mathcal{F}$, and $0 \leq \hat{z}^B_k \leq z^B_k, k \in \mathcal{K}$. In order to see why the values in (54) are feasible for (49)-(52), observe that we decrease the left hand sides of the violated constraints in (50) by scaling down the non-negative variables $u'_i, i \in \mathcal{F}$, and $z^B_k, k \in \mathcal{K}$, 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 objective value associated with the dual feasible solution (54) is given by

$$\hat{z}_{\text{DSMP}} = z_{\text{RSMP}} - (1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u'_i + \sum_{k\in\mathcal{K}} \beta_k z^B_k \right) = z_{\text{RSMP}} - (1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}} \beta_k z^B_k + \sum_{i\in\mathcal{F}} d_i \right) \tag{55}$$

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

$$\frac{z_{\text{RSMP}} - z_{\text{SMP}}}{z_{\text{SMP}}} \leq \frac{z_{\text{RSMP}} - \hat{z}_{\text{DSMP}}}{\hat{z}_{\text{DSMP}}} = \frac{(1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}} \beta_k z^B_k + \sum_{i\in\mathcal{F}} d_i \right)}{z_{\text{RSMP}} - (1 - \bar{s}_{\min}) \left(\sum_{i\in\mathcal{F}} u_i + \sum_{k\in\mathcal{K}} \beta_k z^B_k + \sum_{i\in\mathcal{F}} d_i \right)}. \tag{56}$$

The relation (56) yields on 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 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 start 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 $y_p, y_q, x^k_{(p,q)}$ and the three constraints of type (17)-(19) 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 labels 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_k^e or A_k^d in their final labels 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 $N_{\max}^{init}$ 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 only possible for small problem instances, and we investigate the sensitivity of the objective value and the ultimate number of type A solutions identified by our algorithms to the value of $N_{\max}^{init}$ 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.* [2]. In this research, column management has two primary goals. First, solving the 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 the 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 MLSP 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.

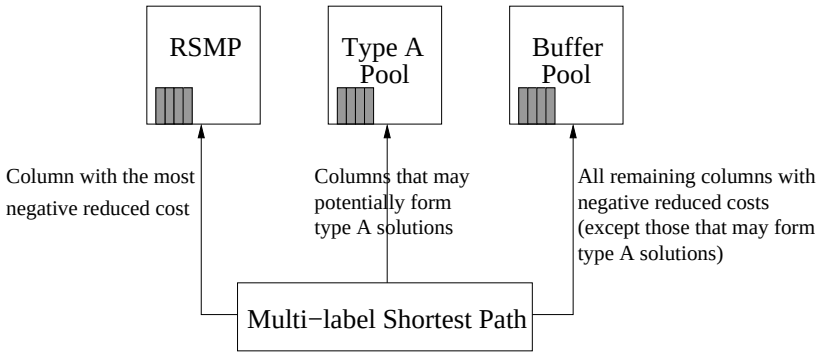


Figure 6: Constructing the column pools.

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. The 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 computational time expended 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 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 [8]. Moreover, any pairing with its final label 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 $N_{\max}^{init} = \infty$ 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 MLSP algorithm 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. See Algorithm 1.

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

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

4. Computational Results. We tested our algorithm on three instances which were obtained from two Turkish airline companies. The first instance is daily and contains 96 flights. The other two instances are weekly with 135 and 490 flights, respectively. For these three instances, there is only one crew base. The time-windows for the possible extra flights are taken as input in addition to the regular flight schedule. Only one extra flight is considered in the data with 96 flights and 135 flights and two extra flights are considered in the data with 490 flights. We conduct the computational experiments on a machine with Intel(R) Core(TM) 2.13GHz CPU and 1 GB of RAM running Windows XP. The column generation code is written in Visual C++ and the restricted short master problem is optimized by ILOG CPLEX 11.0 using ILOG Concert Technology 2.5.

As the initialization procedure in our algorithm provides some pairings to start with, we pointed out that they might not cover all the flights. In that case, the artificial variables in each covering constraint acts as an artificial pairing with a large cost so that the column generation tries to cover them before terminating. Also, when our algorithm exits the main loop, we have to check whether the solution is integral. If yes, then our algorithm ends. Otherwise, there is a problem of finding a good integer solution. We choose to solve an integer program (IP) with the columns in the RSMP. The IP Solver of CPLEX 11.0 is used.

Before testing our robust model, we solve the data sets without incorporating the extra flights; i.e. the problem is solved as a conventional crew pairing problem. The objective function value of the crew pairing problem will provide us the information on the difference in the cost caused by handling the

Table 1: Optimal solutions for the conventional crew-pairing problem for the instances

Instance	Objective Function	Total Time(sec.)
96	1626.67	1.2
135	8383.89	3.14
490	147496	2236.55

Table 2: Optimal Solution for the data with 96 flights

M	N	Objective Function	Gap(%)	Number of Deadheads	Total Time(sec.)
100000	0	2051.67	26	7	1.2

extra flights at the planning level. The results are given in Table 1.

The set of parameters in our robust mathematical model that indicate the number of recovery alternatives provided to the planner are α and β. If these values are chosen to be large, a high degree of flexibility is provided to the planner. When any disruptions occurs in any of the selected pairings that form a type A or type B solution, the planner can assign the extra flight to another alternative solution. However, it decreases the flexibility of the model to select lower cost pairings so that the objective function increases. Also, the large values may result in infeasibility since the number of feasible type A and type B solutions may be lower than these values. We conduct preliminary experiments for each extra flight to have a crude estimate of the feasible type A and type B solutions. The number of feasible type A and type B solutions for the data with 96 flights is expected to be low since the planning horizon is very short. We detect only two feasible type A solutions for this data set and no type B solution can be formed because of the feasibility rules. Thus, the α and β parameters are set to two and zero, respectively. For the data with 135 flights, the number of type A and type B solutions are very large so that it is very time-consuming, if not impossible, to detect them. For the first extra flight in the data with 490 flights, there is no possible type A solution but type B solutions. However, for the second extra flight, the number of type A and type B solutions are excessive. Initially, we select only one value for each of the parameters α and β for the weekly data sets: $\alpha = 6$ and $\beta = 2$.

To test our algorithm on the data sets, first we have to determine the parameters that characterize it. The defining parameters of our algorithm are $N^{init}_{\max}$ and $N^{A}_{\max}$. The initialization procedure that generates feasible type A solutions has a parameter $N^{init}_{\max}$ that restricts the number of partial paths that can accumulate on each node of the flight graph. When no restriction is imposed on the initialization procedure, all feasible type A solutions are generated before starting the column generation procedure. Thus, forcing possible type A solutions during MLSP through $N^{A}_{\max}$ is unnecessary. This parameter selection gives us the optimal solution. When optimal solution cannot be found, a fine-tuning for the parameters $N^{init}_{\max}$ and $N^{A}_{\max}$ is essential. Since each feasible type A solution adds three linking constraints, the size of the model may become excessively large when $N^{init}_{\max}$ increases. This causes the solution of RSMP to take longer. Hence, if there is a large set of feasible type A solutions for an extra flight, it is reasonable to work with small values of $N^{init}_{\max}$ and 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 fixed column pool and are checked for feasibility when column generation terminates. Accordingly, when $N^{A}_{\max}$ increases, the size of the fixed pool grows large and may result in lack of memory. Also, since we observed that ϕ rarely exceeds two, it is fixed to three.

We first attempt to solve the instances to optimality by our robust model which means that no restriction is imposed on the initialization procedure, ϕ is fixed to a large value and the fixed pool is not utilized. In the data with 96 flights, we could find the optimal solution of the robust model which is given in Table 2. The fourth column shows the gap between the optimal solution of the CPP and the RCPEF. It indicates that there is a 26% increase in the cost caused by covering the extra flight.

For the weekly data sets, the processor ran out of memory because of the large number of paths generated during the initialization procedure. For these cases, we conduct a preliminary test to evaluate how the changes in the values of $N^{init}_{\max}$ and $N^{A}_{\max}$ affect the results. The value of exactly one of $N^{init}_{\max}$ and $N^{A}_{\max}$ is fixed to zero and the other one is increased.

Table 3: Evaluation Tests for the data with 135 flights

M	N	ObjFun	NumberofA	SelectedA	DistEx	DistDh	SelDistEx	SelDistDh	SelectedTypeB	TotalTime(sec.)	IntTime(sec.)
0	5	14204.5	2112	6	24	88	3	2	2	149.391	135.719
0	10	14168.9	4671	6	27	173	3	2	2	750.203	718.765
0	20	-	32643	-	117	279	-	-	-	-	-
100	0	14023.9	204	6	12	17	3	2	2	7.094	1.625
200	0	14023.9	840	6	24	35	3	2	2	88.171	78.891
300	0	14023.9	3408	6	48	71	3	2	2	4449.56	4393.39

Table 4: Evaluation Tests for the data with 490 flights

M	N	NumberofA	DistEx	DistDh
0	5	21710	117	279
0	10	27594	189	146
0	20	20544	214	96

Table 3 shows the results for the data with 135 flights where the third column shows the objective function. The fourth and fifth columns represent the number of type A solutions in the RSMP and the number of selected type A solutions at the optimal solution, respectively. While DistEx and DistDh represent the number of distinct pairings that cover the extra flight and the deadhead in the feasible type A solutions (columns 6 and 7), respectively, SelDistEx and SelDistDh represent the number of distinct pairings that cover the extra flight and the deadhead in the selected type A solutions (column 8 and 9), respectively. Column 10 is the number of type B solutions in the optimal solution and last two columns are the computational time and the duration of the IP solver in seconds. When $N^{init}_{\max}$ is fixed to zero and $N^{A}_{\max}$ is increased, the number of feasible type A solutions grow considerably, indicating that the fixed pool is very effective in generating type A solutions. When $N^{A}_{\max} = 20$, the size of the model becomes excessively large because of the feasible type A solutions, making it impossible for the IP to find the optimal solution in a reasonable time limit (the instances where the optimal could not be found is indicated by '-'). The same interpretation also applies to the instances where $N^{A}_{\max} = 0$ and $N^{init}_{\max}$ is increased that the initialization procedure is very effective in generating type A solutions. In both cases, the computation time grows proportionately with the number of feasible type A solutions in the RSMP. It is clear that the number of type A solutions in column 3 is the multiplication of the number of distinct pairings that cover the extra flight and the deadhead, meaning that each pairing that can cover the extra flight form a type A solution with each pairing that can cover the deadhead. Therefore, the large numbers for the feasible type A solutions is caused by the combination of small number of distinct pairings. Lastly, the number of selected type A and type B solutions in the optimal solution is equal to the α and β values except for one instance. This proves the intuitive result that the costs of the pairings that form either a type A or a type B solution is larger than the regular pairings.

Table 4 shows the results for the data with 490 flights. Since the optimal solution could not be found by relaxing the restriction on the initialization procedure, tests for the various values of $N^{init}_{\max}$ and $N^{A}_{\max}$ are also conducted for this case. As we mentioned above, there is no feasible type A solution for the first extra flight so that we did not report the columns respected to the type A solutions for this extra flight. Our first observation is that increasing the parameter $N^{init}_{\max}$ until the initialization procedure is terminated by lack of memory does not result in feasible type A solution for the second extra flight. The reason is that the number of feasible pairings is so large that the restricted number of paths on each node does not result in feasible type A solutions. Increasing $N^{A}_{\max}$ causes sudden increases in the feasible type A solutions as given in Table 5. Since the number of type A solutions in the RSMP is very large, the instances could not be solved to optimality using an IP solver. Again, it can be observed that the large numbers for the feasible type A solutions is caused by the combination of small number of distinct pairings.

To reduce the size of the large number of type A solutions, we restrict the number of alternatives for the pairings that can cover the extra flight. For each of these pairings, if the number of alternatives is larger than 10, we sort these pairings that can cover the deadhead according to their reduced costs and delete until only 10 alternatives remain. We test the effect of this strategy using three values for each parameter $N^{init}_{\max}$ and $N^{A}_{\max}$ (except zero) and also their combinations (both $N^{init}_{\max} > 0$ and $N^{A}_{\max} > 0$) which utilize both the initialization procedure and the fixed pool to detect the type A solutions. Table 5 and Table 6

Table 5: Tests for 135

M	N	ObjFun	NoofPairings	countBuffer	countFixed	NumberofA	SelectedA	SelectedTypeB	TotalTime(sec.)	IntTime(sec.)
100	0	14023.9	302	236	0	120	6	2	6.829	0.642
200	0	14023.9	322	228	0	240	6	2	11.826	1.363
300	0	14023.9	360	204	0	480	6	2	58.426	3.007
0	5	14204.5	372	134	26	240	6	2	8.913	1.049
100	5	14023.9	374	160	29	189	6	2	10.245	0.861
200	5	14023.9	344	157	21	350	6	2	16.369	2.192
300	5	14023.9	426	144	21	590	6	2	78.853	4.057
0	10	14168.9	523	112	37	270	6	2	14.88	1.143
100	10	14023.9	516	143	35	220	6	2	16.541	1.175
200	10	14023.9	590	149	35	410	6	2	24.42	2.162
300	10	14023.9	610	131	24	580	6	2	123.385	5.153
0	20	14023.9	858	113	51	1170	6	2	61.826	8.443
100	20	14023.9	962	147	42	1510	6	2	105.857	12.687
200	20	14023.9	966	136	37	1310	6	2	98.589	12.61
300	20	14023.9	1037	122	36	1630	6	2	195.941	17.341

Table 6: Tests for 490

M	N	ObjFun	NoofPairings	countBuffer	countFixed	NumberofA	SelectedA	SelectedTypeB	TotalTime(sec.)	IntTime(sec.)
0	5	156204	3697	867	293	1790	6	2(2)	5178.05	140.953
0	10	155792	4604	806	331	1890	6	2(2)	2480.33	201.766
0	20	158413	6276	833	382	2110	6	2(2)	12686.9	213.172

shows the results for this setting. Three new columns are added to the table; column 4 is the number of pairings in the RSMP, column 5 is the number of pairings in the RSMP that are taken from the buffer pool and column 6 is the number of pairings in the RSMP that are taken from the fixed column pool. The latter two columns in both Table 5 and Table 6 justify the use of the column pools in avoiding the MLSP problem whenever possible since a remarkable proportion of the pairings in the RSMP is extracted from the buffer pool or the fixed pool. There is no discernable difference between the objective function values in Table 5 and Table 3, but the number of type A solutions and the computational time used in solving the IP decreases in parallel. Table 6, where the number of type A solutions is considerably decrease, indicates that the instances can be solved with our deletion procedure (the number in parentheses gives the number of selected type B solutions for the first extra flight for which there is no type A solution). Hence, we can say that our deletion procedure is successful.

At this point, we can test the effect of the parameters α and β on the results. Tables 7 and 4903 shows the results for $\alpha = 2$ and $\beta = 1$. Compared to the results in Tables 5 and 6, the objective function decreases considerably.

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

Table 7: Tests for 135 with $\alpha = 2$ and $\beta = 1$

M	N	ObjFun	NoofPairings	countBuffer	countFixed	NumberofA	SelectedA	SelectedTypeB	TotalTime(sec.)	IntTime(sec.)
100	0	10894.5	295	233	0	120	2	1	5.671	0.517
200	0	10894.5	319	228	0	240	2	1	11.905	0.924
300	0	10894.5	363	210	0	480	2	1	57.472	3.243
0	5	11298.9	367	120	37	270	2	1	10.699	1.378
100	5	10894.5	399	145	32	350	2	1	14.928	1.159
200	5	10894.5	396	137	23	370	2	1	17.81	1.645
300	5	10894.5	467	135	25	720	2	1	79.307	2.695
0	10	10982	545	107	49	210	2	1	15.131	0.925
100	10	10894.5	559	130	43	240	2	1	18.123	0.972
200	10	10894.5	565	125	37	440	2	1	27.162	1.801
300	10	10894.5	638	124	43	700	2	1	94.767	4.683
0	20	10894.5	645	136	56	880	2	1	41.87	6.375
100	20	10894.5	980	137	47	1500	2	1	112.295	10.307
200	20	10894.5	997	150	34	1470	2	1	122.07	13.785
300	20	10894.5	960	159	29	1210	2	1	150.327	12.328

Table 8: Tests for 490 with $\alpha = 2$ and $\beta = 1$

M	N	ObjFun	NoofPairings	countBuffer	countFixed	NumberofA	SelectedA	SelectedTypeB	TotalTime(sec.)	IntTime(sec.)
0	5	151697	4299	910	282	2140	2	1	7573.31	58.031
0	10	152053	4861	815	327	2200	2	1	3939.44	927.921
0	20	152637	6051	791	510	2450	2	1	4527.47	283.609

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

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

[9] H. Tekiner, S.I. Birbil, and K. Bulbul. Robust crew pairing for managing extra flights. *Computers & Operations Research*, 36:2031–2048, 2009.

[10] 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.