COLUMN GENERATION APPROACHES TO A ROBUST AIRLINE CREW PAIRING MODEL FOR MANAGING EXTRA FLIGHTS

by

ELVİN ÇOBAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2008

COLUMN GENERATION APPROACHES
TO A ROBUST AIRLINE CREW PAIRING MODEL FOR MANAGING EXTRA FLIGHTS

APPROVED BY

Assoc. Prof. Dr. Ş. İlker Birbil (Thesis Supervisor)	.....
Assist. Prof. Dr. Kerem Bülbül (Thesis Co-supervisor)	.....
Assist. Prof. Dr. Güvenç Şahin	.....
Assist. Prof. Dr. Hüsnü Yenigün	.....
Assist. Prof. Dr. Dilek Tüzün Aksu	.....

DATE OF APPROVAL: .....

to my family

Acknowledgments

I would like to express my gratitude to my thesis advisor, Associate Professor Ş. İlker Birbil for his inspiration, enthusiasm, patience, and sincerity. I would like to thank my thesis co-advisor, Assistant Professor Kerem Bülbül whose enthusiasm has guided me through all my research. His inspiration and sincerity always motivates me. Without my advisors' guidance and support, completion of this thesis would be impossible. Their motivation and encouragement will always guide me through out my professional career. I would also like to thank Assistant Professor Hüsnü Yenigün for his guidance, his friendly attitude and his motivation. I am grateful to Assistant Professor Güvenç Şahin and Assistant Professor Dilek Tüzün Aksu for their enormous support and inspiration. I would also like to thank İlker Topçu for sharing his knowledge about the airline schedule planning.

I am grateful to my colleague and friend, İbrahim Muter for his invaluable support and guidance through my thesis. I want to thank Duygu Taş for her coordination and motivation throughout this research. I would also like to thank my dear friend Merve Peyiç for her trust, motivation and endless friendship. I also want to thank Sarper Göktürk for his support.

I would like to thank TÜBITAK for providing me financial support throughout my masters.

Lastly, I am very grateful to my mother Mehbüp Çoban, my father Ali Çoban and my sister Güler Çoban Çiçek for the concern, caring, endless love and support they provided throughout my life. Without them, I would not succeed.

COLUMN GENERATION APPROACHES TO A ROBUST AIRLINE CREW PAIRING MODEL FOR MANAGING EXTRA FLIGHTS

Elvin Çoban

Industrial Engineering, Master of Science Thesis, 2008

Thesis Supervisor: Assoc. Prof. Dr. Ş. İlker Birbil
Assist. Prof. Dr. Kerem Bülbül

Keywords: robustness, crew pairing, extra flights, column generation, row generation, column pools.

Abstract

A typical airline crew pairing problem aims at selecting a set of flight sequences (pairings) for crews such that each flight in the regular schedule is covered by one crew. In this thesis, we consider the management of potential extra flights that can possibly be introduced to the regular flight schedule during operation at a later point in time. Without delaying or canceling any existing flight, we try to handle these extra flights within the regular schedule and refer to the resulting mathematical model as a robust airline crew pairing model. The objective function of the robust model involves not only the regular pairing costs but also the opportunity costs for failing to cover the extra flights. Due to the large number of variables (pairings), a typical crew pairing model is usually solved by column generation methods. Before applying column generation to the proposed robust model, we first discuss several procedures to cover the extra flights by a given set of feasible pairings. However, these procedures introduce extra column-dependent constraints to the model. That is, as new columns are added by column generation to the model, the number of constraints may also increase. Similarly if a column is removed from the model, then some of these extra constraints may be deleted. To handle this dynamic change both in the number of constraints and variables we propose two approaches. The main idea behind these approaches is to generate a set of pairings (column pool) so that the number of constraints can be fixed. To this end, we flag the pairings that can be used for covering the extra flights and keep them in a special pool. We illustrate the proposed column generation approaches on a set of actual data acquired from a local airline.

DAYANIKLI EKİP EŞLEME PROBLEMİNDE KOLON YÖNETİMİ

Elvin Çoban

Endüstri Mühendisliği, Yüksek Lisans Tezi, 2008

Tez Danışmanı: Doç. Dr. Ş. İlker Birbil
Yrd. Doç. Dr. Kerem Bülbül

Anahtar Kelimeler: dayanıklı, ekip eşleme, kolon türetme, satır türetme, kolon havuzları

Özet

Klasik ekip eşleme probleminde amaç, uçuş çizelgesinde yer alan uçuşlar kapsayan, belirli kısıtlar altında en az maliyetli olan ekip eşlemelerinin bulunmasıdır. Son yıllarda hava yolu şirketleri planlanan çizelgelerini uygulamakta zorlanmaktadırlar. Çizelgeden sapılmasının nedenlerinden biri çizelgeye eklenen ek uçuşlardır. Çizelgeye yeni eklenen uçuşun kapsanması operasyonel seviyede gerçekleşir. Dayanıklı ekip eşleme problemimizi çizelgemizde gerçekleşebilecek olan bu ek uçuşları göz önünde bulundurarak tanımladık. Planlama aşamasında bilinen bu ek uçuşları olabildiğince uçuş rotalarını çıkarırken ele almaya çalıştık. Bu sayede ek uçuşlar gerçekleştirildikleri zaman, daha önceden çıkardığımız rotaları en az maliyetle kullanabilmeyi amaçladık. Önerdiğimiz dayanıklı ekip eşleme problemi modelinde yeni eklenen kolonlar (eşlemeler) satır ve sütun olarak problemin büyümesine neden olabilmektedir. Biz problemdeki satır ve sütun büyümelerini de dikkate alarak klasik ekip eşlemesinde de sık kullanılan bir metod olan kolon türetme metodu ile iki tane çözüm yöntemi önerdik. Bu çözüm yöntemlerinde amacımız ya kolon türetme metoduna başlamadan önce ya da kolon türetme metodunu bitirdikten sonra dinamik büyümeye neden olan kısıtları eklemektir. Etkili kolon türetme metodu için kolon havuzları da tanıttık. Önerdiğimiz yöntemleri yerel bir firmanın datalarında denedik.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Since 1950's Operations Research models have been widely used to solve complex operational and planning problems in the airline industry. With the highly competitive global market and the fierce competition between the airline companies, the crew scheduling has become a widely studied topic in the literature as the crew costs are the second major cost component in the airline industry. Building a cost efficient crew schedule is a challenging problem because of the complex regulations and the large scale of the problem. The crew scheduling problem is divided into two subproblems: firstly, a crew pairing problem is solved for constructing a sequence of flights (pairings); secondly, a crew assignment problem is solved for assigning each crew individually to the constructed pairings. In this thesis, we focus on the crew pairing problem [4, 7, 9].

The crew pairing problem aims at selecting a set of pairings for crews such that each flight in the regular schedule is covered by a crew. The conventional mathematical model of the crew pairing problem is a set partitioning model,

$$\begin{aligned} \min \quad & \sum_{p \in P} y_p c_p \\ \text{s.t} \quad & \\ & \sum_{p \in P} a_{ip} y_p = 1, \quad \forall i \in F, \\ & y_p \in \{0,1\}, \qquad \forall p \in P, \end{aligned} \tag{1.1}$$

where F and P are the sets of flights and pairings, respectively. Here, c_p is the cost of pairing p, $a_{ip} = 1$ if the flight leg i is covered by pairing p; 0, otherwise. The decision variable y_p is a binary variable, which is equal to 1 if pairing p is chosen; 0, otherwise. The objective function of problem (1.1) is the minimization of the total pairing cost, and the first set of constraints ensures that each flight is covered exactly once [3]. It is also common to model this problem as a set covering problem, in which each flight can be covered by more than one pairing. In this case the first set of constraint is replaced

with equation (1.2). Due to the large number of variables (pairings), a typical crew pairing model is usually solved by a column generation approach [21].

$$\sum_{p \in P} a_{ip} y_p \geq 1, \forall i \in F \tag{1.2}$$

There is a substantial cost difference between the actual and the planned costs in the flight schedules. Unprecedented events like maintenance problems, weather conditions, crew sickness can cause frequent disruptions resulting in delays or cancellations of flights. Crews and aircraft schedules are shuffled by these disruptions. Therefore, a crew recovery problem should be solved. However, the crew recovery problem aims at finding a good solution as fast as possible because the current flight schedule has already become operational.

There are two ways for minimizing the difference between the actual and the planned costs caused by these irregularities in the schedule. One of them is improving the quality of the recovery procedures at the operational level, and the other one is constructing more robust schedules at the planning level [7]. In this thesis, we are motivated by these disruptions and our work focuses on a cost minimization.

We consider an operational problem with the motivation of minimizing the gap between the actual and planned costs. However, we solve it at the planning level, not at the operational level.

One of the irregularities faced by the airline companies is the potential extra flights that will be introduced to the regular flight schedule. These extra flights will be realized during operation at a later point in time [31]. For instance, during summer there may be an increase in demand for popular holiday destinations, or certain types of customers like businessmen, sports teams may demand extra flights, which are not in the regular schedule. Even though many local companies estimate the time interval for these possible extra flights, they do not handle them at the planning level but at the operational level. The local companies do not treat these extra flights as regular flights because they cannot guarantee that these extra flights will be realized in the schedule. They revise their schedule during operations, if these extra flights are realized. They can even cancel or delay their regular flight schedule in order to cover these extra flights, and such disruptions increase the operational costs and the gap between the actual and the estimated cost. Therefore, robust airline crew pairing becomes an important topic to handle the disruptions in airline industry. In our study, we con-

sider disruptions due to the addition of extra flights to the regular flight schedule. We propose a mathematical model and two solution approaches for solving these proposed model. These are referred as dynamic and static approaches, respectively. The proposed mathematical model is an extension of the model in [31]. Unlike [31], we consider column generation as the solution approach in our study. In the proposed dynamic and static approaches, we try to handle the column-dependent constraints resulting from the proposed mathematical model. That is, as new columns are added to the model by column generation, the number of constraints may also increase. Similarly, if a column is removed from the model, then some of these extra constraints may be deleted. To implement an effective column generation algorithm, we introduce three column pools into our models. Moreover, we illustrate the proposed dynamic and static models on a set of actual data acquired from a local airline.

1.1 Contributions of This Research

Crew scheduling is very important for airline companies due to high costs. In particular crew pairing requires more attention, since a major cost component for an airline company comes from the crew costs. The conventional mathematical model for the crew pairing problem aims at minimizing the total cost of the pairings that cover the flights in the schedule. However, there can be disruptions in the regular flight schedule. To prevent an increase in the costs and to achieve a high-level of customer service, airline companies try to handle these disruptions effectively. One type of the disruptions is due to the extra flights inserted into the regular flight schedule. These extra flights are anticipated at the beginning of the schedule. Therefore, we have a chance to consider these extra flights at the planning level.

When extra flights are handled at the operational level, this may cause disruptions in the schedule, like delays or even cancellations of the regular flights. Therefore, we propose a robust airline crew pairing model where the regular pairing costs and the opportunity costs for failing to cover the extra flights appear in the objective function. This robust airline crew pairing model is solved at the planning stage. Due to the large number of pairings, a typical crew pairing model is usually solved by column generation methods. We also use column generation methods with our robust airline crew pairing model. However, we need to define some extra column-dependent constraints in our robust airline crew pairing model for handling the extra flights. Thus, our model grows dynamically both in the number of constraints and variables

Whenever new columns are added (deleted) to the model by column generation, the number of constraints may also increase (decrease). For handling this dynamic change both in the number of constraints and variables we propose two approaches. In the proposed approaches, we try to fix the number of constraints. We either consider these extra column-dependent constraints before the column generation by enumerating all columns causing the dynamic change in the model (static approach) or by ignoring the extra column-dependent constraints through column generation and considering them at the end of column generation (dynamic approach).

To sum up, we are motivated by the need for considering the robustness in the airline crew pairing problem at the planning level and propose a new mathematical model solved by two proposed approaches based on column generation. We observe that our model ends up with more flexible pairings in the sense that we can cover the extra flights as well as the scheduled flights. Whenever these extra flights are realized at the operational level, the airline company should utilize the pairings constructed at the planning stage without delaying or canceling the regular flights.

1.2 Outline

The outline of the thesis is as follows. Chapter 2 gives a brief literature review of the crew pairing problem, robustness and the column generation method. This literature review is followed in Chapter 3 by the proposed static and dynamic approaches for handling the extra flights in our flight schedule. A numerical study on a set of actual data acquired from a local airline company is given in Chapter 4. Chapter 5 contains the conclusion.

CHAPTER 2

LITERATURE REVIEW

The crew scheduling problem can be defined as assigning a group of workers (crews) to a set of tasks. This problem is realized in different areas such as bus and rail transit, truck and rail freight transport, or freight and passenger air transportation. Although these problems appear in different application areas, all crew scheduling problems share common features like covering all tasks and minimizing costs under the provision of safety regulations and labor negotiations [7]. In this thesis, we consider the first part of airline crew scheduling problem, the crew pairing problem.

Before reviewing literature on the crew pairing problem, we define some important terminology [7]:

- **Flight leg:** A nonstop flight (also called segment).
- **Duty period:** A sequence of flights that may be covered without violating the feasibility rules. The duty period defines a day's work for a crew. Each duty period should be followed by a rest time according to the elapsed time passed in the current duty period completed.
- **Sit time:** The time between two flight legs in a duty period. For each connection between two consecutive flight legs there are maximum and minimum sit time controls.
- **Elapsed time:** Time that has passed in the current duty period or in the current pairing.
- **Pairing:** A sequence of duty periods with overnight rests in between. A crew starts the duty at its base and returns to its base at the end of the pairing.
- **Deadhead flight:** A crew flying as passengers.

The increasing popularity of the airline crew scheduling problem in the last few

of flights, the crew scheduling problem becomes a planning problem. Also airline crews receive higher salaries than equivalent personnel in other modes of transportation such as rail or truck. A small improvement in the schedule yields significant cost savings. In addition, there is a large number of restrictive rules in the airline industry, which makes the airline crew scheduling problem harder to solve than other scheduling problems [7]. Airline companies should follow the Federal Aviation Authority work regulations (FAA), which vary by crew type (pilot or flight attendant), crew size, aircraft type, type of operation (domestic or international). Other rules are the labor union rules. The main idea behind these rules is not only minimizing the crew fatigue but also ensuring the passenger safety. All these rules make the crew scheduling problem more complex and challenging. [1].

Airline schedule planning, as represented in Figure 2.1, is composed of four main steps. At the first stage, the timetable is constructed. At this stage of planning, we try to satisfy the expectations of the marketing department with the availability of fleets and without violating the constraints on the network, such as; time slots available for the airline at different airports. From this stage, the number of flight legs to be operated is given as the input to the fleet assignment stage. At the fleet assignment stage, we try to assign fleets to the flight legs. The main concern at this stage is to ensure the feasibility of the timetable according to the available fleet.

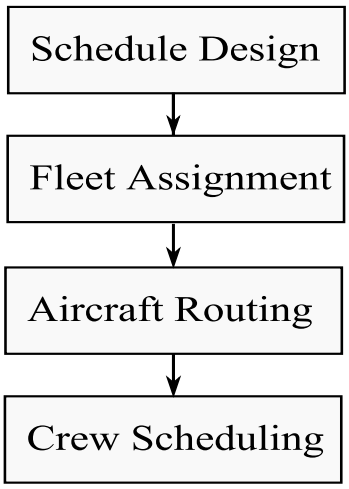


Figure 2.1: Airline schedule planning.

At the aircraft routing stage, each aircraft is assigned to flights taking into consideration the time spent at airports as well as the time spent for routine maintenance checks. If it is impossible to satisfy this main concern, then the timetable may have to be changed. At the crew scheduling stage, the timetable and the fleets assigned to

pairing and crew assignment(rostering) problems are handled at this stage. Under the crew scheduling problem, as mentioned previously in Section 1, we try to solve two subproblems: the crew pairing and the crew assignment problems. The crew pairing problem aims at selecting a set of flight sequences (pairings) for the crews such that each flight in the regular schedule is covered by one crew. In the crew assignment problem, the sequence of pairings is assigned to individual crew members [4].

2.1 Alternate Mathematical Model

The conventional mathematical model for crew pairing can be modeled in two ways: as a set partitioning problem (1.1) or as a set covering problem (constraint set is replaced with (1.2)).

When deadheading is permitted in the crew pairing problem, we need to model the problem as a set covering problem so that some flight legs may be covered more than once. In the study of Lavoie *et al.* [24] it is mentioned that the set covering model may represent the problem structure better since covering a flight leg more than once may reduce the cost since it provides an advantage of moving crews to another place. However, some of the airlines do not permit deadheading in their solutions, especially when solving their daily problems. Therefore, they use set partitioning problems [4]. Both set covering and set partitioning problems are utilized in different application areas.

For instance, in the study of Kohl [35], the mathematical model used for the crew pairing problem is a set covering problem which is utilized in *Carmen Systems*. *Carmen Systems* develop optimization systems for crew scheduling problem which are widely used in major airline companies such as Lufthansa, British Airways, Air France, Alitalia, and so on. Kohl also includes some extensions of the set covering problem for crew pairing like including the base constraints for ensuring pairings match with the distribution of crews or the global constraints for limiting the length of pairings or the variable crew satisfaction constraints (as each leg does not require same number of crews) [22].

Both set covering and set partitioning problems are common in Operations Research. However, set covering and set partitioning problems are NP-hard. Therefore, large-scale problems are not easily solved and long computation times are required to solve these problems [14]. As a result, there are many heuristics and exact methods to solve set partitioning and set covering problems. Most of the methods rely on

continuous relaxations but these solutions are known to be not only difficult but also frequently degenerate [2].

2.2 Pairing Generation

For small sized problems, the total generation of all possible columns may be possible. For large scale problems, however, generating all feasible columns require both time and computational effort. The computational effort also depends on the network structure used. There are two different network types: flight network and duty period network. In the flight network, we have flights represented with arcs showing possible connections. In the duty period network, we have arcs for duty periods and possible connections between duty periods [7]. It may seem easier to work on duty period network during pairing generation but in that case we need to spend time on duty period generation. On the other hand, we do not need to track most of the feasibility rules in a duty period network that we need to track in a flight network. A detailed study about working on duty period network including generating the duty periods is found in the work of Vance *et al.* [32]. The pairing generation method is similar in both networks. In this thesis, we focus on the flight network. Therefore, during generation of pairings, we need to track several criteria for feasibility of the connections. The restrictions on pairings, such as: the FAA rules, the operational considerations, the contractual restrictions, and so on, make the pairing generation problem more complex [33].

Most of the solution approaches depend on the partial enumeration of columns and work on this constructed subset of pairings. One of the previous methods is based on generating pairings on a subset of flights since covering all flights is difficult [1]. Another approach used in *Carmen Crew Pairing*, is limiting the number of possible connections on the network and constructing a subset of pairings. For instance, according to their experienced workers, the short connections are considered as more useful [4]. However, these partial enumerations may lead obtaining local minimums.

In the work of Klabjan *et al.* a new method is proposed for partial enumeration, which is based on the generation of random pairings. During enumeration, connections are chosen randomly. The probability of selecting a connection depends on the length of the connection. The smaller the connection time, the larger the probability of selecting this connection. One of the motivations is that low pairing costs may result in short connections [20]. Also in another method proposed by Klabjan *et al.*, parallel processing is utilized for pairing generation. The distribution of generated duties across

multiple processors on a parallel machine permits the parallel application of complex rules that should be mandatory during pairing generation [19].

2.3 Alternate Ways to Solve The Mathematical Model

There are different methods used in the literature for optimizing the mathematical model for the crew pairing problem. Branch-and-price, Langrangian relaxation and Benders decomposition are the most commonly used techniques. Before reviewing literature on specific methods to solve the crew pairing problem, a literature review of common methods is mentioned.

In a branch-and-price algorithm, we try to solve the linear programming relaxations by column generation but at each iteration we can consider only a subset of columns. We allow the dynamic column generation in the branch-and-bound tree during branch-and-price [7]. Restricted master problem(RMP) is the problem in which we consider only a subset of the columns. In each iteration, RMP is solved and the optimal dual values are taken. Then, the subproblem is solved by using the dual values for calculating the reduced costs of new columns (new variables). If there is any column with a negative reduced cost, we cannot stop as we do not reach the optimal solution. However, if we are not able to find such a favorable column, we can stop the iterations and conclude that the optimal solution has been found. One of the computationally intensive parts of column generation can be stated as solving the subproblem because it needs to scan many columns [21].

Lagrangian relaxation is another common technique. During the design of Lagrangian relaxation based system, three questions are important: which constraints should be relaxed, how good dual variables should be computed and how should a good solution be deduced [12]. If we are able to partition the constraints into easy and difficult categories, the difficult constraints should be moved to the objective function with linear penalties. Therefore, we get rid of the difficult constraints in the model and we also discourage the violation of these constraints by the penalties in the objective function. With the Lagrangian method, the goal is finding best lower bounds found by the Lagrangian relaxation over the possible penalties. This problem is nonlinear and called the Lagrangian dual problem. The subgradient method is commonly used to solve the dual problem. One of the negative feedbacks about using subgradient approach is the difficulty to maintain feasibility [6, 21].

Benders decomposition is another common technique. At every iteration, we solve

a mixed integer RMP with a single continuous variable and obtain a lower bound for the optimal solution. Afterwards, we fix the variables according to the RMP result. The optimal dual values of the linear program provide a Benders cut. This cut is added to the RMP and the iterations are repeated [21].

Accordingly, in most of the techniques, it is common to solve the LP relaxation of the mathematical problem and then propose another approach to improve and arrive at integer solution. For instance, in [8] a combination of the interior point and the simplex methods is used for optimizing LP for a large-scale crew pairing problem.

One of the specific methods to solve a small number of constraints with a very large number of columns is the *SPRINT* method. In the *SPRINT* method, the linear program is solved at the beginning using a small subset of columns. Then, the dual values are used for pricing all columns of the original problem. Then, we check whether the solution is optimal. If it is not, we should have at least one column with negative reduced cost. Therefore, the columns with the negative reduced cost are added to the problem whereas the other columns are randomly selected. This is repeated until there is no column remaining with negative reduced cost. In this case, the problem is optimized. Low solution time with good performance is achieved even if the number of iterations of *SPRINT* method increases [13]. As an addition, there is an algorithm called the volume algorithm in which the feasible dual solutions can be found. The volume algorithm is utilized for finding a feasible dual solution. The feasible dual solutions found by the volume algorithm provides an approximation to the optimum value [5]. In [3], the authors utilize the dual solutions found by the volume algorithm in column generation phase whenever *SPRINT* method performs poorly. Using the volume algorithm also results in faster solutions than not only the dual simplex, or the primal simplex algorithms but also than the interior point algorithm.

Another system developed for solving the crew scheduling problem is proposed in [1]. This system is called trip reevaluation and control (*TRIP*). *TRIP* is based on iterative pairing generation and solving subproblems. Firstly, an initial solution is considered. If it is not possible to easily find a feasible initial solution, the authors suggest modifying the previous period's solution. Secondly, the set partitioning problem is solved and an IP solution is found. One of the main ideas is increasing the size of the subproblem and the number of iterations. The motivation behind this work is greater the number of iterations and the size of the subproblem, the better the solutions obtained. Therefore, the author looks if there exists a better subset of pairings that

result in a better objective function value. If there is, this subset of pairings is added to the subproblem and the problem is resolved. However, after some iterations or after meeting an improvement criteria, iterations are stopped and it is concluded that the obtained solution is the best found solution until now. However, this solution should be local minima. Therefore, one of the authors' major objectives is increasing the speed of *TRIP* iteration especially for solving large subproblems. As a result, they can do more iterations than before in the same time period. With *TRIP*, the authors conclude that there is a great improvement in cost such as annual savings of $20 millions. *TRIP* has been used by some big airline companies like American Airlines and Continental Airlines [1].

One popular method for solving the airline crew pairing problem is utilizing a column generation approach [3].As stated Lagrangian duality is another alternative for solving the relaxation of the mixed integer problems. Depending on which method produces the optimal dual solutions more efficiently, during optimization one can use Lagrangian duality or the column generation. Subgradient algorithm is accompanied with both Lagrangian duality and the column generation. Subgradient method is detailed in the study of Held et al [18]. Subgradient algorithm is easy to code but besides its simplicity, slowness and convergence troubles for large problems are reported. The limitations of the column generation method is also given by Held and Karp [16, 17]. From their study, they conclude that subgradient algorithm should be preferred to any linear methods.

As the solvers and the computation power improve over time, the dynamic column generation method becomes an interesting tool for considering all possible pairings during solving the LP relaxation. The set partitioning or the set covering problem are referred as the master problem if it is solved considering all possible pairings. In this thesis, column generation method is used. Therefore, for gaining a brief insight about a column generation approach, we handle the steps of the method.

2.3.1 Column Generation

The column generation algorithm is studied in detail by Desrosiers and Lubbecke [10, 26]. The column generation method involves three general steps [7]:

1. Solving RMP

2. Solving the pricing subproblem

3. Updating the RMP if a new column can be added to the solution

When row and column generation are all conducted simultaneously, as pointed out by Barnhart *et al.*, combining row and column generation is a challenging topic. The authors mention that whenever additional rows are added to the model, the pricing problem's structure changes. One of the possible ways to handle such dynamic row and column generation is by doing the pricing over the original set of columns without considering the additional rows. However, there are cases in which the newly added row does not also destroy the pricing problem structure such as in the problem of minimizing the number of vehicles required to satisfy the demands of the customers. If a constraint for cutting the fractional objective function value is added, then the pricing problem structure will not change because the newly added constraint has coefficients of one at each column [6].

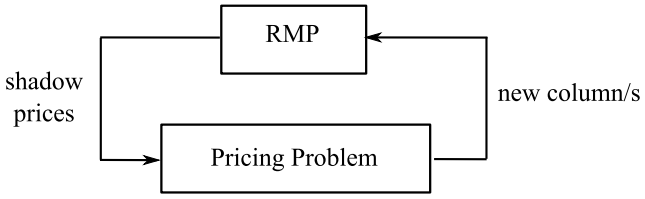


Figure 2.2: Relation between RMP and pricing subproblem.

In the first step of column generation, an optimal solution is found to current RMP which contains only a subset of columns. The primal simplex algorithm for solving RMP is believed to be the most efficient algorithm. However, there are disadvantages of using the primal simplex method. One of the disadvantages is the degeneracy that the LP relaxation will face. Another disadvantage is the optimal extreme point dual solutions that give misleading information about the reduced costs. Therefore, several iterations of column generation are done. One of the major benefits of using the primal simplex method is being able to warm start. Interior point algorithms are also used but they lack warm start [7].

The volume algorithm mentioned previously is another alternative for producing dual vectors. The volume algorithm is claimed to overcome the simplex algorithms and interior point algorithms [5, 3].

As Barnhart *et al.* refers, there are two main questions: the criteria for selecting the pairings and how can we find the pairings meeting the criteria. The criteria about the reduced costs are the most common ones. Such as in [8] the cost of each pairing divided by the respective reduced cost may be considered as a criterion.

We search through the network by multilabel shortest path algorithm to find the pairings meeting the criteria or enumerate by the brute force approach [7]. Generating all pairings is an alternative solution. However, it is costly. In [3], partial enumeration in pricing is done with a criterion based on the reduced costs. A detailed study about the pricing problem can be found in [30].

Whenever we find a favorable column considering the criteria chosen, we add this column to the RMP model. We insert the column and update the coefficient matrix about the coverage of the corresponding flight legs.

2.4 Crew Recovery

Due to some disruptions in the schedule, such as maintenance problems, severe weather conditions, and so on, an airline schedule rarely operates as planned. One needs to consider the modification of a crew schedule that has been affected by disruptions. The resulting model is called the crew recovery problem (CRP). CRP is solved at the operational level. The objective of CRP is reassigning crews such that a schedule is maintained with the minimum additional crew costs and the minimum cancellations of the regular flights [25, 7].

The planning problem differs from CRP in several ways. The major difference is the time horizon for solving the problems. Barnhart *et al.* points out the goal of CRP as finding a good solution in a short time period. The second difference is the necessity of considering the recent flying history of the active crews in the CRP. Third difference is including the reserve crews in the CRP. Reserve crews can be defined as the crews that have a minimum guaranteed pay (measured in flying hours) and have hour limitations for flying in a month. Another difference is the feasibility rules for the pairings in each problem. In the planning problem, most airlines try to construct flexible pairings that may recover the possible disruptions. However, in CRP the rules on flying times are often pushed to their legal upper bounds. The last and the more significant difference is the objective functions of the two problems. CRP should have different objective functions like returning to the original plan as quickly as possible and minimizing the passenger disruptions or keeping down the additional cost of reserve crews. Even deciding on the objective function can be difficult in the CRP [7].

In the works of Lettovsky *et al.*,a new optimization based solution approach is introduced. The authors provide a computationally inexpensive deadhead selection heuristic, propose an integer model for the CRP and solve this problem using primal-

dual subproblem simplex method [25].

2.5 Robustness

The crew pairing problem is solved at the planning stage. Therefore, all flights are assumed to have fixed and known departure times. However, at the operational level, as mentioned previously in Section 1, there may be disruptions in the regular planned schedule. In US it is reported that at least 25 % of the flight legs were delayed by 15 minutes or more in the summer of 2000 due to the disruptions. Clearly, the crew pairing problem gains more attention and airline companies try to focus on solving the robust crew pairing problem to handle the distruptions.

There are three existing approaches for finding the robust crew pairings. In the first approach, the expected pairing costs are included in the model. The costs are computed by running *Simair* (Simulation of Airline Operations), which is a modular airline simulator used for evaluating plans and recovery policies[27]. The expected costs are assumed as independent of the other pairings in the schedule. This assumption holds if we use the push-back recovery procedure. In the push-back recovery procedure, we delay the flights until all the resources are available. After computing the cost components, the crew pairing problem is solved and the solution is evaluated again by *Simair* [28].

The second approach for the robust airline crew pairing problem is based on maximizing the connection times. Both in the works of Ehrgott *et al.* [11] and Yen *et al.* [34], robustness is measured in terms of excess sit connection time above the minimum sit connection time. The authors define penalty with a penalty factor times the difference between the connection time and minimum required sit connection time. If we sum these penalties over all sit connections in a pairing, we define the robustness cost of a pairing. Then, the model which minimizes the robustness cost is solved. In [34], a stochastic integer programming model is studied. It is assumed that the connection time is a random variable in the model. As the model has a large-scale, the authors propose a heuristic based on follow-on branching rule for solving the model. In [11], the connection times are assumed to be deterministic and the connection times are taken with respect to the planned flight schedule.

In the work of Shebalov and Klabjan [29], move-up crews are defined as a resolution to the disruptions. Move-up crews can be defined as the crews available for swapping one or more flights when another crew is delayed or has reached the bounds of the fea-

sibility rules, such as reaching the limit on the flying time. When the move-up crews are used in the recovery procedure, besides the traditional objective function of minimizing pairing cost, they consider the maximization of the number of opportunities for crew swapping. The model proposed is a bicriteria optimization model. A Langrangian relaxation approach is used for solving the model.

Additionally, in the work of Tekiner *et al.* [31], a new robust mathematical model is proposed. Their work can be considered as an extension of the work of Shebalov and Klabjan [29] in which only the original crews are used for covering the extra flights.

CHAPTER 3

STATIC AND DYNAMIC APPROACHES

3.1 Problem Statement

Airline crew pairing problem is one of the challenging scheduling problems. One of the reasons is the existence of different crew categories in the airline crew pairing problem, such as; captains, first officers, flight attendants, and so on. Therefore, we need to solve the crew pairing problem considering this crew category structure as none of the crew can substitute another crew if they belong to different categories [3, 4].

Moreover, cockpit crews receive higher salaries than cabin crews. Therefore, in our study we consider crew pairing problem for cockpit crews as solving the crew pairing problem for cockpit crews and optimizing the solution may decrease the cost of the airline company more than optimizing the crew pairing problem for cabin crews. As a result, in the mathematical model we propose, each flight should be covered at least once. If we consider also the crew pairing problem for the cabin crew, then we will either increase this right hand side of this coverage constraint of each flight from one or solve the problem for the required number of cabin crew times.

The crew pairing problem is solved according to the fleet type as each pilot is commonly licensed to fly only one type of fleet. Therefore, we concentrate on the flight schedule of the same fleet.

In the study of Tekiner *et al.* [31], the authors examine several ways of recovery options for managing extra flights at the planning level. They classify the possible solutions into two types:

- **Type A.** Two pairings are selected and swapped for covering the extra flights.
- **Type B.** One pairing is selected such that there is enough connection time between two consecutive legs for covering the extra flight.

Nine possible Type A and Type B solutions are proposed, here we only consider

Type A solution. The method proposed in [31] is different than ours. We only introduce three types of solutions.

In all solution types we consider, the potential extra flights that can possibly be introduced to the regular flight schedule do not have exact departure times. According to estimates done by the airline company, these extra flights are scheduled within a range of time. As shown in Figure 3.1 and 3.2 the extra flights are shown with two parallel arrows which represent the time range defined for the extra flights.

The Type A solution is illustrated in Figure 3.1.

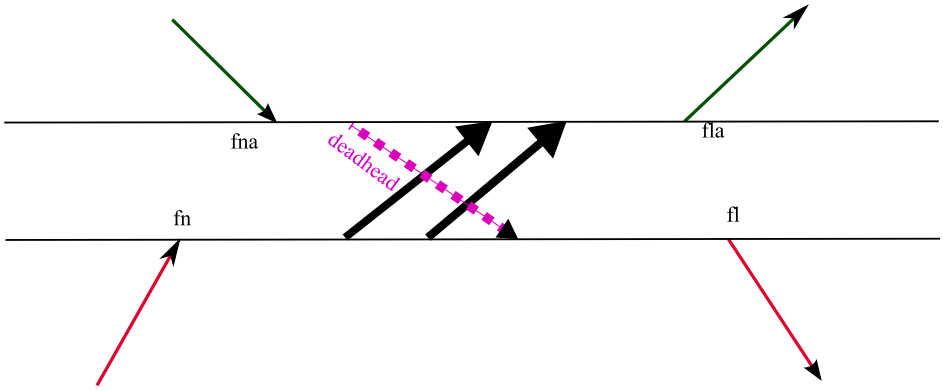
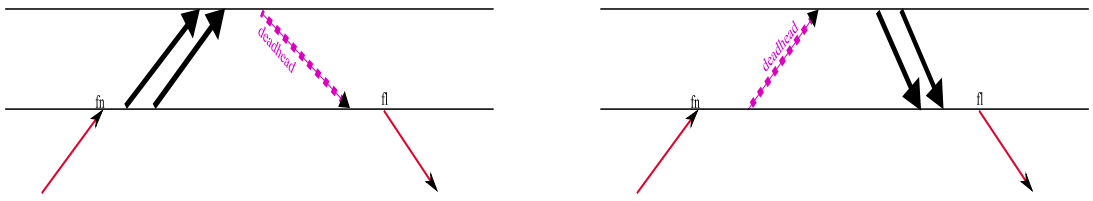


Figure 3.1: Type A solution.

Similarly, in Figure 3.2, two Type B solutions are illustrated.



(a) (b)

Figure 3.2: Type B solution.

Difference between the two Type B solutions is the location of the deadhead flight. Deadhead may appear in the schedule before or after the extra flight.

3.2 Mathematical Model

We propose a new mathematical model for the robust airline crew pairing problem.

major differences between the two models. One of the main differences is the objective functions. In the study of Tekiner *et al.*, the objective function is maximization of the extra flight coverage but we try to minimize the total cost including not only the regular pairing costs but also the opportunity costs for failing to cover the extra flights. In [31], a constraint is introduced for not deviating from the objective function value of the conventional crew pairing model. Therefore, the authors first solve the conventional mathematical model and then solve the robust mathematical model they propose. However, we do not solve our problem twice as we include all cost components in our objective function including the opportunity costs for failing to cover the extra flights.

The notation used in the proposed mathematical model is given in Table 3.1.

P	:	set of pairings
F	:	set of flights
K	:	set of extra flights
c_p	:	cost of pairing p
d_k	:	opportunity cost of failing to cover extra flight k
a_{ip}	:	1 if flight leg i is covered by pairing p, 0 otherwise
$\bar{a}_{kp}$	:	1 if pairing p covers extra flight k as Type B solution, 0 otherwise
$\bar{a}_{pqk}$	:	1 if pairings p and q cover extra flight k as Type A solution, 0 otherwise

Table 3.1: The notation used in the mathematical model.

Our decision variables are:

$$y_p = \begin{cases} 1, & \text{if pairing } p \text{ is chosen,} \\ 0, & \text{otherwise;} \end{cases}$$

$$z_k = \begin{cases} 1, & \text{if extra flight } k \text{ is not covered,} \\ 0, & \text{otherwise;} \end{cases}$$

$$x^k_{(p,q)} = \begin{cases} 1, & \text{if extra flight } k \text{ is covered by pairing } p \text{ and pairing } q \text{ as Type A solution,} \\ 0, & \text{otherwise.} \end{cases}$$

The proposed mathematical model is given as follows;

$$
\begin{aligned}
\min \quad & \sum_{p\in P} y_p c_p + \sum_{k\in K} d_k z_k + \sum_{k\in K} d_k \Big(\sum_{p\in P}(1-y_p)\bar{a}_{kp} + \\
& \sum_{p,q\in P}(1-x^k_{(p,q)})\bar{a}_{pqk}\Big) \\
\text{s.t} \quad & \\
& \sum_{p\in P} a_{ip} y_p \geq 1, && \forall i \in F, \\
& \sum_{p\in P} \bar{a}_{kp} y_p + \sum_{(p,q)\in P} \bar{a}_{pqk} x^k_{(p,q)} \geq 1 - z_k && \forall k \in K, \\
& 2\bar{a}_{pqk} x^k_{(p,q)} \leq y_p + y_q, && \forall p,q \in P, \forall k \in K \\
& y_p \in \{0,1\}, && p \in P \\
& z_k \in \{0,1\}, && k \in K \\
& x^k_{(p,q)} \in \{0,1\}, && p,q \in P, k \in K.
\end{aligned}
\tag{3.1}
$$

The regular pairing cost of a pairing p is defined as

$$
c_p = \max\{f_p.TAFB,\ ndp.mg,\ \sum_{d\epsilon p} b_d \tag{3.2}
$$

where mg and f_p are constants. $TAFB$ represents time away from base. ndp is the number of duty periods in pairing p whereas b_d is the sum of the duty periods' costs of pairing p. Pairing cost is, simply, the maximum of three components; sum of costs of duties in pairing p, some fraction of total elapsed time, and minimum guaranteed number of minutes per pairing.

In problem (3.1), we try to minimize the total cost which is the summation of four terms: cost of regular pairings, cost of failing to cover the extra flights, and the opportunity costs for failing to cover the extra flights with a Type A solution or with a Type B solution. Our objective function promotes the coverage of the extra flights as we incur costs of failing to cover the extra flights. The first set of constraints ensures that each flight leg should be covered at least once. The second set of constraints is for covering each extra flight with Type A or Type B solutions. The last constraint set is defined for all feasible Type A solutions. When a feasible Type A solution exists, the respective extra flight should be covered by the pairings forming the Type A solution. Finally, we have constraints for defining our decision variables as binary variables.

3.3 The Column Pools

Before starting with the dynamic and static models, we explain the column pools and their impact on solving the problem. As mentioned in Section 1, we have introduced the column pools for an effective column generation algorithm. The proposed column pools are illustrated in Figure 3.3.

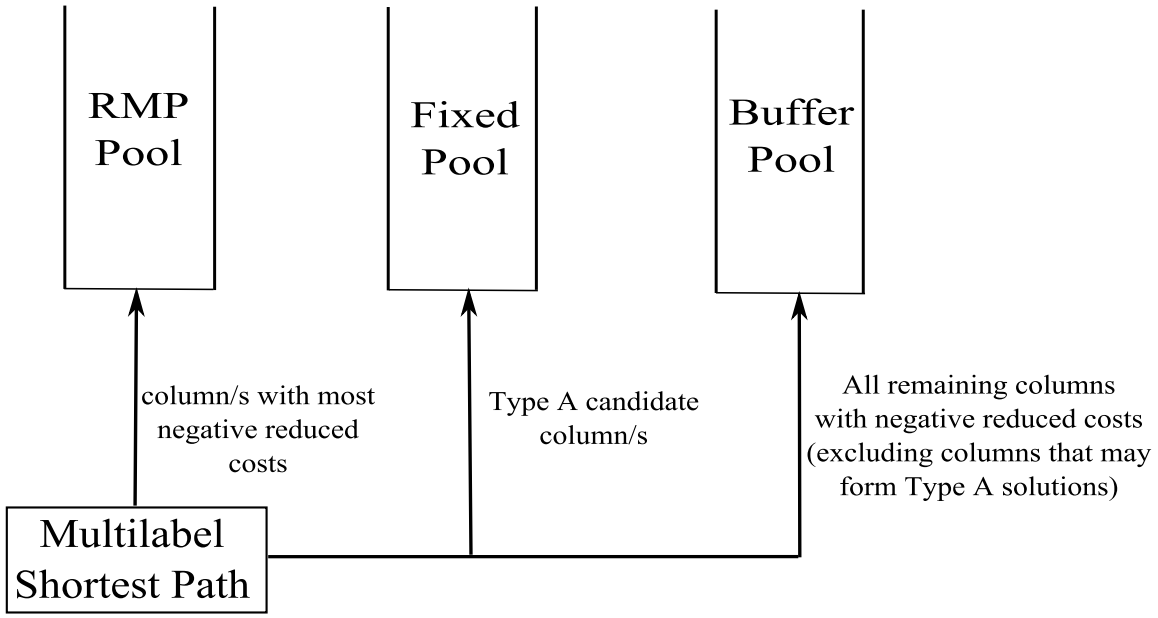


Figure 3.3: Constructing column pools.

We fill these pools with columns after solving the multilabel shortest path either at the intermediate iterations or at the beginning of the column generation procedure. Initially, the buffer pool and the fixed pool are empty. Before solving the LP relaxation, RMP column pool is filled either with the initial feasible pairings (dynamic approach) or with all possible columns forming Type A solutions (static approach).

The first column pool is the RMP. During our column generation algorithm, we send the columns with the most negative reduced cost to this pool as shown in Figure 3.3. Then, we solve the RMP again, since a new column is added to our model and hence the dual prices are altered. Before searching for another column with negative reduced cost, we need to update the dual prices; therefore, the reduced costs.

The second column pool is the fixed column pool which holds the columns of the candidate Type A solutions. In the multilabel shortest path, each column is tagged. These tags show whether this column can cover the extra flight as a Type A solution (extra flight or deadhead) or as a Type B solution. In the fixed column pool, all Type A solution candidates are held even if they have positive reduced costs. Fixed column

any column may form feasible Type A solution with another column is preserved. We want to conclude with as many alternative solutions as possible for covering the extra flights.

The third column pool is the buffer pool which holds the columns with negative reduced costs apart from the ones in the fixed column pool. After solving the multilabel shortest path, we send the column(s) with the minimum reduced cost to RMP pool. All columns with Type A tags are sent to the fixed column pool. All remaining columns with negative reduced costs are sent to the buffer column pool.

The main advantage of using the column pools is to do the column generation effectively without solving the multilabel shortest path at each iteration. Solving the multilabel shortest path is an expensive and exhaustive process [30]. Therefore, we construct the column pools, and at every iteration we check the fixed column pool, and the buffer column pool if there exists a column with negative reduced cost. If there is, we do not solve the multilabel shortest path which saves us from computational burden. If there is not, we solve the multilabel shortest path again and update the column pools. In the buffer column pool, we clean some of the columns with positive reduced costs [30]. However, the column management is not permitted for the fixed pool, as we do not want to lose any candidate Type A solutions.

3.4 Static Approach

We call this approach static because whenever we fix all Type A solutions, we also fix the constraints $2\bar{a}_{kp}x^k_{(p,q)} \leq y_p + y_q, \forall (p,q) \in P, \forall k \in K$. Fixing these constraints are important because during column generation we do not need to pay special attention to dynamic growth of rows in the problem (3.1) with newly added variables. Column generation is applied to the relaxation of the problem (3.1).

In the static approach, we do backward search from each extra flight (Figure 3.7) and forward search from the source node to sink node (Figure 3.8) to find the candidate nodes of pairings that may form Type A solutions and to form candidate Type A solutions for covering the extra flights or the deadheads. For each extra flight, we first find the nodes which will be visited just before the extra flight or the associated deadhead. Then from these nodes, we try to find which nodes can be visited before these tagged nodes. During tagging, we use the information from topological sorting which is done at the beginning of the static model. We check the topological order of each node and connect each node with another node in the flight network with a higher

topological order.

After tagging all previous candidate nodes as shown in Figure 3.7, regular multilabel shortest path [30] is modified and solved. A multilabel shortest path algorithm is used to find new pairings. The labels represent the state of each path in the flight network with respect to the cost and the rules of the pairing and the duty periods. Besides the regular multilabel shortest path, we first go over the tagged nodes until the extra flight or the deadhead is covered, then we update the information about whether the path can cover the extra flight or the deadhead. Then, we continue solving the regular multilabel shortest path with the rest of the graph. When the multilabel shortest path is finished, we check all the paths(pairings) on the sink node coming from the source node for taking the tagged paths. After, we check whether these tagged paths (Type A solution candidates) are still feasible after they are swapped. If they are, these pairing pairs are fixed in the model. Next, we do this backward and forward search for all extra flights. At the end, we fix all Type A solutions at the beginning of column generation if the swapped pairings are still feasible. In Figure 3.4 and Figure 3.5, the proposed static approach is summarized.

In the static approach, the fixed column pool is not required since we fix all feasible Type A solutions before starting the column generation. The fixed column pool holds the columns with Type A tags but in the static approach, we fix all feasible Type A solutions in RMP pool. Therefore, there is no reason for using the fixed column pool in the static model.

In the following figures, backward and forward search over the network is visualized. In Figure 3.6, step 1 of tagging is done. The nodes that may be visited just before the extra flight or its associated deadhead flight are tagged in this step. These nodes are decided according to their destinations and the feasibility rules. Then, as shown in Figure 3.7, tagging of all other nodes with different style dashed lines, that have topological orders less than the tagged nodes, are searched for whether there may be feasible connections between the previously tagged nodes. After all, as shown in Figure 3.8 the multilabel shortest path is done and we end up with the pairings which are candidates of Type A solutions.

Suppose all found pairings composed of the flight legs drawn with dashed lines in Figure 3.9 are feasible and the base for the pairings is C. There are a couple of example of pairings on the network:

- Pairing 1 consists of flights with topological orders 1 - 4 - 7 - 10

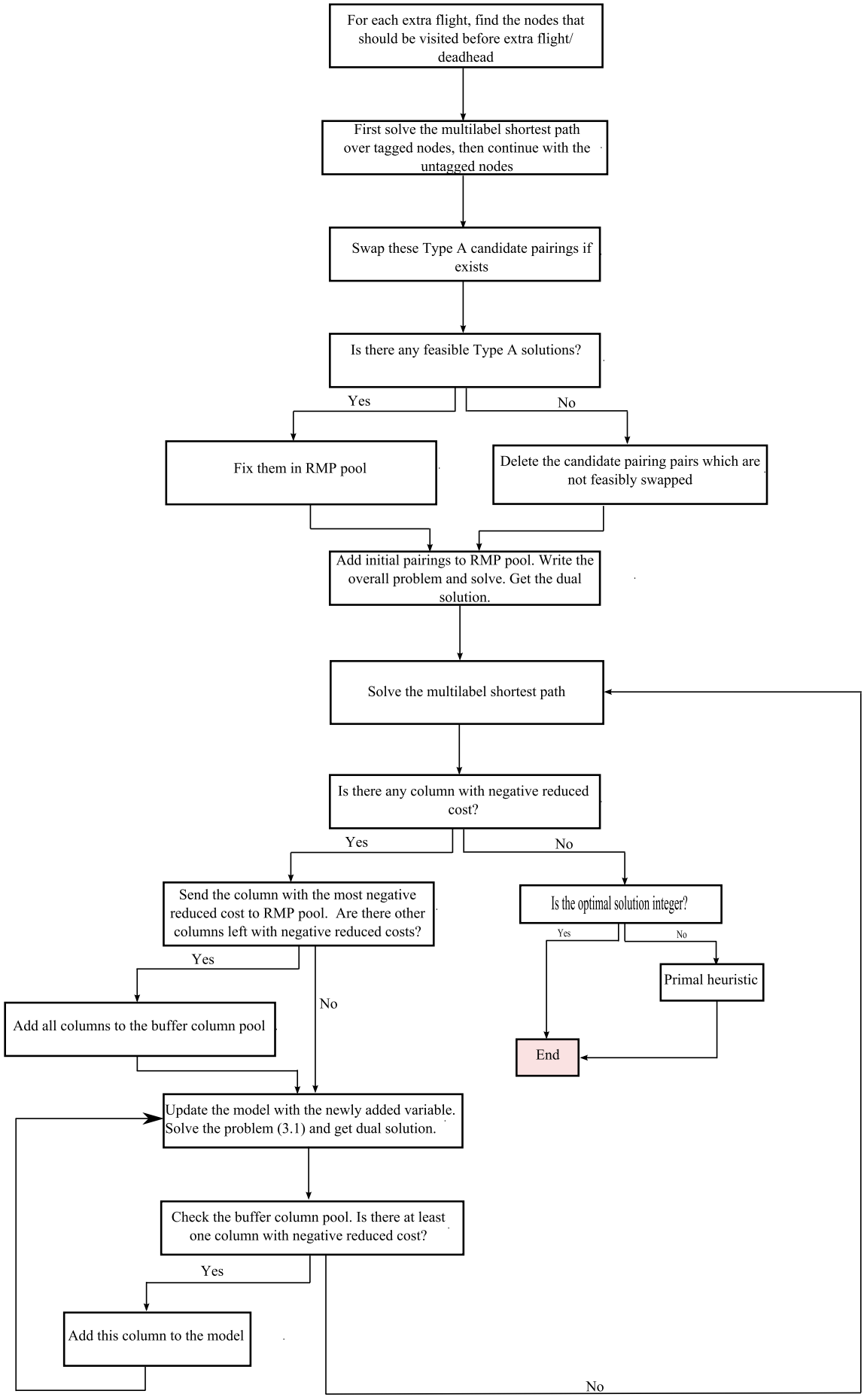


Figure 3.4: Flowchart of static model

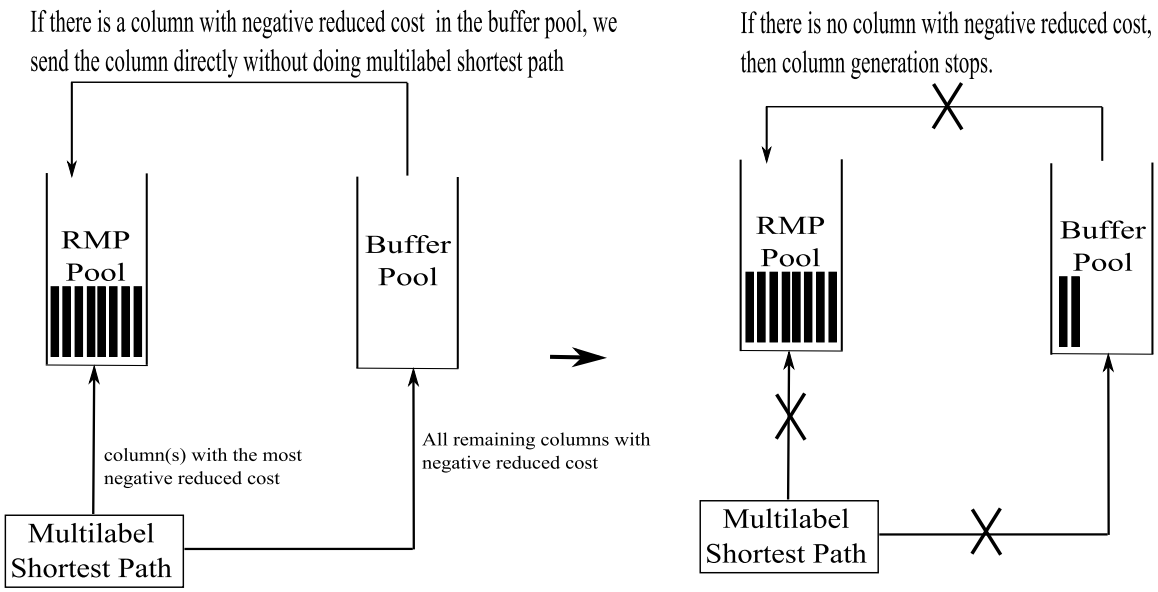


Figure 3.5: The proposed static approach.

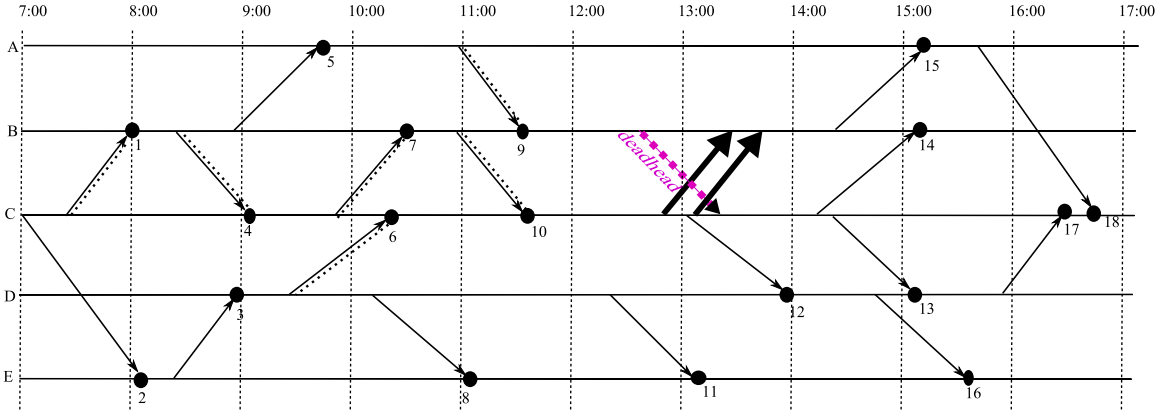


Figure 3.6: Backward tagging flight nodes - step 1.

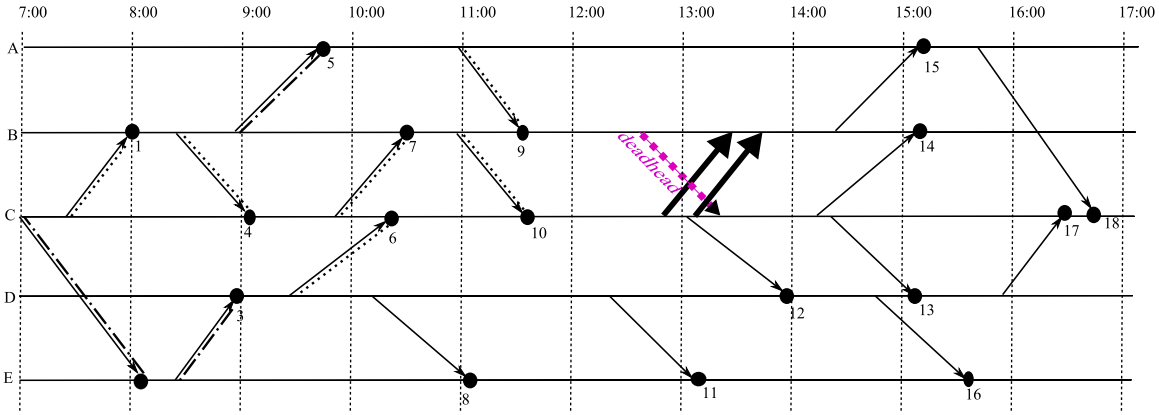


Figure 3.7: Backward tagging flight nodes - step 2.

- Pairing 2 consists of flights with topological orders 1 - 5 - 18.
- Pairing 3 consists of flights with topological orders 2 - 3 - 6 - 12 - 17.
- Pairing 4 consists of flights with topological orders 5 - 9 - 15 - 18.

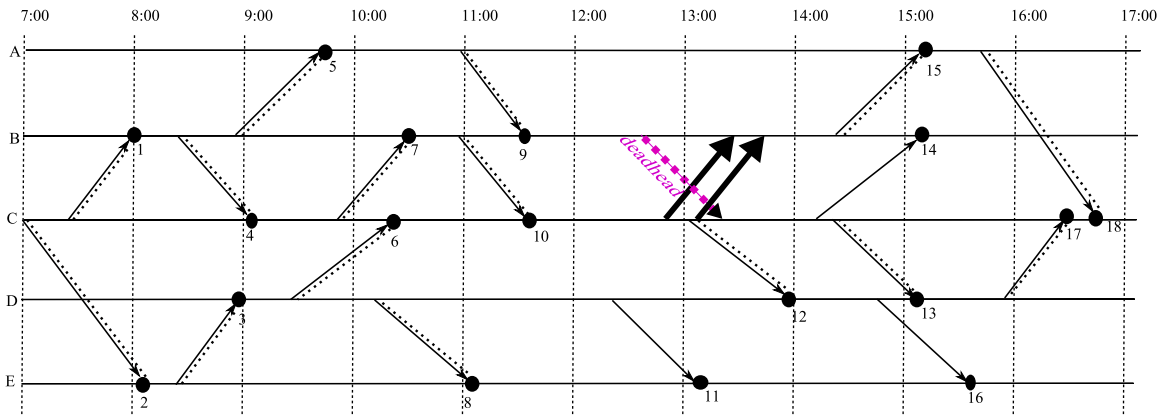


Figure 3.8: Forward search over the flight nodes - step 3.

- Pairing 5 consists of flights with topological orders 2 - 3 - 6 - 13 - 17.

As shown in Figure 3.9, from the pairings we realize at the end of multilabel shortest path that there are feasible swappable pairings for covering the extra flight and the associated deadhead. For example, pairing 4 and pairing 5 form a feasible Type A solution in Figure 3.9.

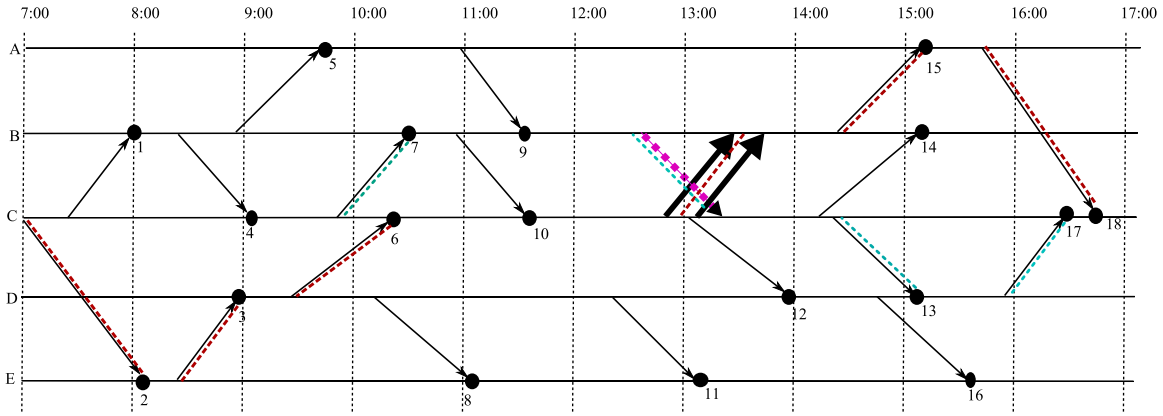


Figure 3.9: Type A solutions.

In the static approach, our regular column generation algorithm changes. As shown in Figure 3.10, after we fix feasible Type A solutions, newly coming columns do not have coefficient of one in the third constraint set of problem (3.1) which is written for feasibly swapped Type A solution. We treat Type B solutions as they are different pairings than their original pairings so we make a copy of each original pairing which is a Type B solution and update the cost of the pairing according to Type B solution. Therefore, during the column generation algorithm, we do not consider the dual variables coming from the third set of constraints. Because whatever the dual variables respective to the third constraint sets are, as their coefficients are all zero in the dual mathematical model as we fix Type a solutions constraints at the beginning of column

generation. Therefore, we can not cause dual infeasibility during column generation. Consequently, we calculate the reduced costs considering the first and second constraint set's respective dual variables.

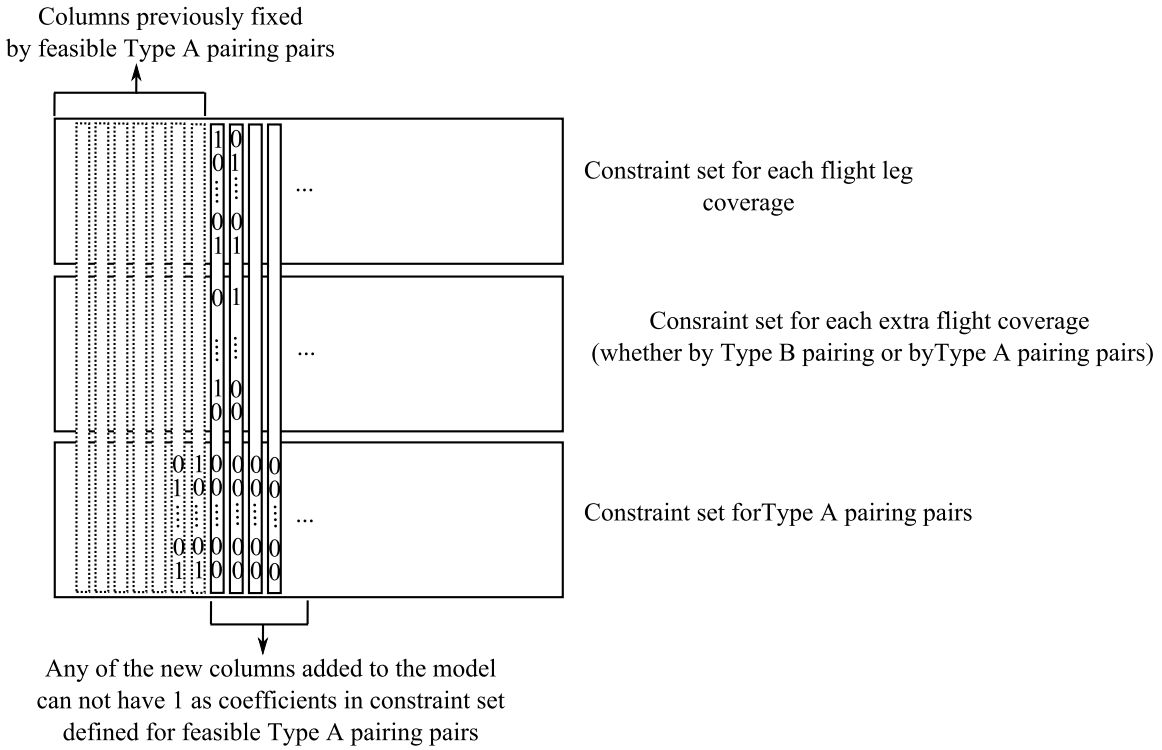


Figure 3.10: The treatment of columns in the problem (3.1) by the static approach.

3.5 Dynamic Approach

The dynamic approach is the second approach that we propose for handling extra flights occurring in our regular flight schedule. We call this approach dynamic because as new columns are formed during column generation, the ones that are Type A solutions are formed too. We exclude all Type A solutions from our problem (3.1) and apply column generation over the relaxation of the remaining problem.

The mathematical model used in dynamic approach is given as follows;

$$\begin{aligned}
\min \quad & \sum_{p\in P} y_p c_p + \sum_{k\in K} d_k z_k + \sum_{k\in K} d_k \Big(\sum_{p\in P} (1-y_p)\bar{a}_{kp}\Big) \\
\text{s.t} \quad & \\
& \sum_{p\in P} a_{ip} y_p \geq 1, && \forall i \in F, \\
& \sum_{p\in P} \bar{a}_{kp} y_p \geq 1 - z_k && \forall k \in K, \\
& y_p \in \{0,1\}, && p \in P \\
& z_k \in \{0,1\}, && k \in K
\end{aligned} \tag{3.3}$$

We do not write any set of constraints in problem (3.1) for Type A solutions. Because if we consider the Type A solutions during iterations, the pricing problem structure should be changed. With the addition of a new column(variable), a new constraint may be added to our problem and the dual problem may also change. The dual problem also grows dynamically both in the number of constraints and variables. Therefore, the dual variables we use for finding reduced costs are not the same as considering our previous respective dual problem. Therefore, we write problem (3.1) including the Type A solutions' constraints after the column generation. As an addition, we relax the constraints of defining the decision variables as binary variables in the column generation. In this approach, column generation method is done over the model excluding Type A solutions for handling the dynamic change both in the number of constraints and variables in the problem (3.1).

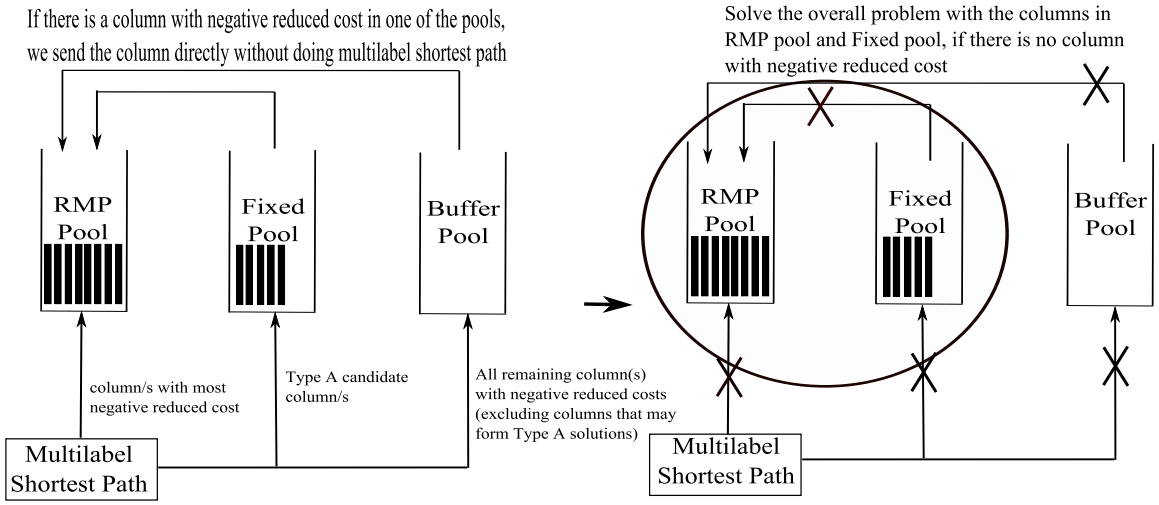


Figure 3.11: Proposed dynamic approach.

In Figure 3.11, the dynamic approach is summarized. The crosses over some arcs show that there is no favorable column remaining. Firstly, we start with the initial pairings and solve the model excluding the constraints of feasible Type A solutions. We take the dual variables' values and solve the multilabel shortest path for checking whether there exists any favorable column. If there exists, we send this column(s) to RMP. Before solving the model, we check if there are appropriate columns that can be sent to the fixed column pool or to the buffer column pool. If there are, we update the column pools with these pairings. We solve the relaxation of the model excluding variables and constraints of Type A solutions once more. Then, we check if there are column(s) with negative reduced cost in the fixed column pool or in the buffer column pool. If there are, we do not solve the multilabel shortest path problem at this iteration.

In case none of the pools include a column with negative reduced cost, then we solve the multilabel shortest path problem and update the column pools according to the introduced pairings. Our column generation stops if there will be no favorable column coming from the multilabel shortest path or from the column pools.

When the column generation algorithm ends, we check if the pairings at RMP column pool including the pairings at the fixed column pool are candidate Type A solutions. If there are any candidate Type A solutions, we swap these pairings and check their feasibility when they are swapped. If there are any feasible Type A solutions, we consider them during solving problem (3.1). At the end, we solve the problem (3.1) by taking into account all Type A and Type B solutions.

The dynamic approach is only a heuristic because during the multilabel shortest path, candidate Type A solutions may be dominated by other pairings [30]. Therefore, we may miss some of the Type A solutions.

In Figure 3.12, proposed dynamic approach is summarized.

3.6 Primal Heuristics

The crew pairing problem is a binary integer programming problem in which all variables are binary variables (0-1 variables). Therefore, all variables must have integer values at the end. Most successful algorithms for solving the integer programming problems incorporate corresponding linear programming (LP) problems, which are referred as the LP relaxations, because the integer programming problems are much more difficult to solve than the LP problems. During the column generation method, the RMP is also a LP relaxation to the mathematical model of the robust airline crew pairing problem. Due to solving the LP relaxations, the solutions of the dynamic and the static approaches are not guaranteed to be integer. Therefore, we implement a primal heuristic to result with integer solutions. As an addition, we implement another primal heuristic to obtain an initial basic feasible set of pairings.

An initial basic feasible set of pairings is necessary found for both the dynamic approach and the static approach. We need to start with a feasible coverage of all flight legs in the regular flight schedule and we search for the favorable pairings by the column generation approach. There are two alternatives to obtain the initial feasible set of pairings: constructing new set of pairings until all flight legs are covered or taking the previous period's feasible pairings and modify them according to the current flight schedule. The latter way of finding an initial feasible set of pairings will be an extension

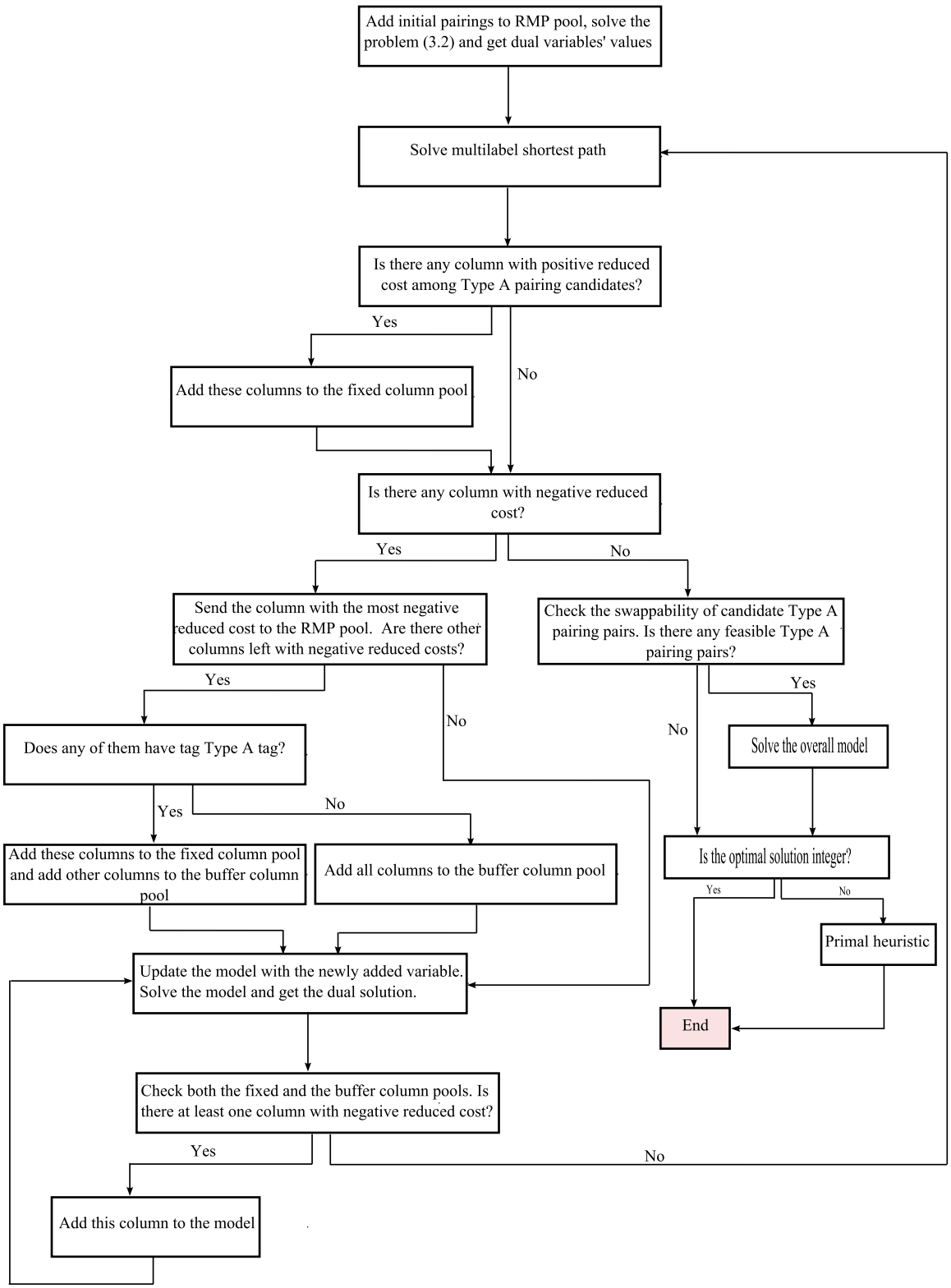


Figure 3.12: The flowchart of the dynamic approach.

of our study.

Currently, we search the flight network for all nodes having a zero indegree or a zero outdegree. If we find such nodes, we need to connect the nodes to either sink or source node. The source and the sink nodes represent the same crew bases. If the indegree of a node is zero, we connect the node to the source node. As shown in Figure 3.13 the

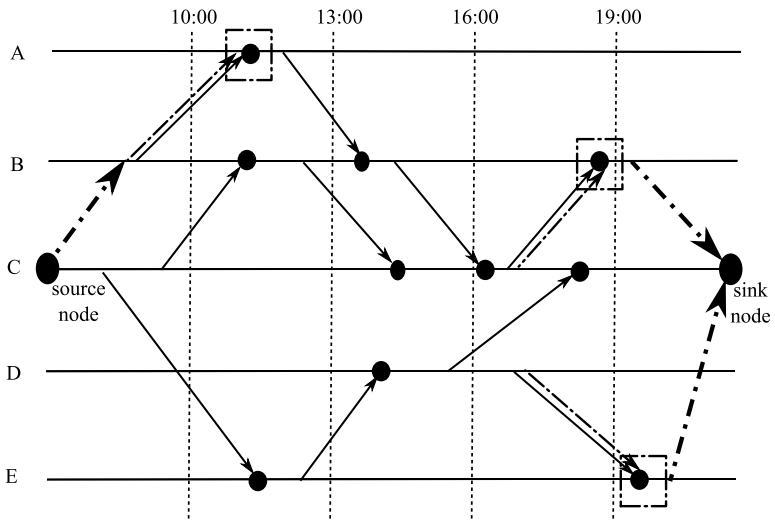


Figure 3.13: Connecting the flight nodes having zero indegree and zero outdegree to source and sink node, respectively.

nodes shown with a dashed lined rectangle around, have either zero indegree or zero outdegree. In this figure, the crew base for pairings is City C. Therefore, each legal pairing should start and end at City C. If a node has an outdegree of zero, we connect the node to sink node located at City C. In Figure 3.13 these connections shown with dashed arcs and the flight legs after these connections are shown with two parallel arcs. The dashed arcs connecting the flight legs to source or sink node correspond to deadheading. After satisfying the connectivity of these nodes to source or sink nodes, we solve the multilabel shortest path problem once. With all generated pairings by the multilabel shortest path problem, we solve the conventional mathematical model as a set covering model. The optimal solution found forms our initial feasible set of pairings. We replace some of the pairings in this set to obtain a worse initial feasible set of pairings. While replacing the pairings in the set of initial feasible pairings with the other pairings generated by the multilabel shortest path problem, property of coverage of all flight nodes' at least once is not violated.

The second primal heuristic is for obtaining integer solutions for the decision variables. We implement the heuristic proposed by Lan *et al.* However, as the proposed mathematical model is slightly different than the conventional mathematical model of the crew pairing problem, we develop an additional procedure about covering the extra flights. The heuristics in [23] is for solving the set covering problem by applying the Meta Heuristics for Randomized Priority Search (Meta-RaPS). A feasible solution is generated by introducing some random factors. Then, an improvement heuristics is applied but the selection of priority rules, the penalization of worst columns are

With Meta-RaPS, if the decision variables for our regular pairings are not integers, we can end up with integer solutions for the decision variables. However, this heuristic is not enough for making the decision variables of Type A solutions, x_{pq}^k, integer. Therefore, we propose a new procedure consisting of three steps: checking if all x_{pq}^k is integer or not, fixing some variables, removing the redundant pairings.

Firstly, we check if there are any noninteger x_{pq}^k variable in the solution. If there are any, then we fix the missing pairing pair of the variable, whether y_p or y_q. Because the only way of the decision variable x_{pq}^k can be noninteger,if its pairing pair is missing.

$$2\bar{a}_{pqk}x_{p,q}^k \leq y_p + y_q, \forall p, q \in P, \forall k \in K \tag{3.4}$$

The constraint 3.4 is not violated when $x_{(p,q)}^k = 0.5$ and one of the y_p and y_q is equal to 1 and the remaining Type A solution pair is equal to 0. Therefore, we fix the missing pairing value as one and then do this fixing for all missing pairing pairs of the Type A solutions (for all $x_{(p,q)}^k = 0.5$). After that, we search if there are redundant pairings, which are not feasible Type A solutions, caused by the addition of the fixed pairings which are part of feasible Type A solutions. After removing the redundant pairings, we end the primal heuristic. To sum up, both of the primal heuristics are guaranteed to arrive at feasible integer solutions.

CHAPTER 4

COMPUTATIONAL RESULTS

4.1 Feasibility Check for the Swapped Pairings

Feasibility rules are vital during construction of the pairings. Each pairing consists of duty periods which are day-long consecutive flights assigned to crews and each duty period should end with rest period. Each rest period is decided according to the elapsed time of the duty period. Considering the duty periods, we have rules for maximum sit/connect time, minimum sit/connect time and maximum elapsed time. We construct the duty periods from the flight legs that are sequential in space and time. We also make the controls if the crew has enough time for resting.

According to the rules of the FAA, Table 4.1 shows the maximum elapsed time for each duty period. Besides, the minimum rest periods after each duty period according to the ending duty period's elapsed time are shown in Table 4.2. Minimum sit times between flights can not be smaller than 60 minutes, which is the preparation time for the next flight leg. At the beginning of each duty period there should be 30 minutes difference other than 60 minutes according to the rules.

	Number of Flight Legs		
Starting Time	**1 - 3**	**4 - 5**	**6 or more**
05:00-14:00	14 hours	13 hours	12 hours
14:01-17:00	13 hours	12 hours	11 hours
17:01-04:59	12 hours	11 hours	10 hours

Table 4.1: Maximum elapsed time for duty periods according to duty period's starting time.

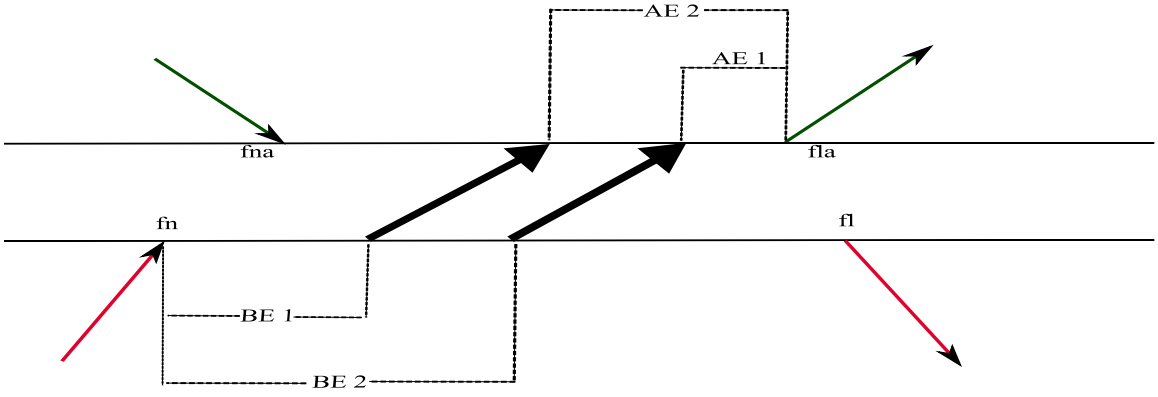
Previous Duty Period's Elapsed Time	Minimum Rest Periods
4 hours or less	8 hours
4-11 hours	10 hours
11-12 hours	12 hours
12-14 hours*	14 hours
18 hours or more	20 hours

(*: if time zone difference is more than 3 hours)

Table 4.2: Minimum rest periods after each duty period according to previous duty period's elapsed time.

Besides these duty period rules, we always have location consistency between consecutive flight legs, if there are no deadheads in the schedule. Each arrival of a flight leg will be next flight leg's departure location. Considering pairings, we have rules for duty periods in a pairing. In a pairing the first duty period starts and the last duty period ends at the same location ,which is the crew base. All duties in a pairing, like flight legs in a duty period, are sequential in space and time. There are minimum required rests after the duty periods as illustrated in Table 4.2.

For checking whether the pairings can form Type A solutions, we also consider these feasibility rules during the construction of the duty periods and the pairings. In Figure 4.1, we see a zoomed section of a candidate Type A solution pair. One of the pairings includes the flight nodes *fn* and *fl*. This pairing is called pairing X and the other one including flight nodes *fna* and *fla* is called pairing Y. Suppose the first pieces of the pairing (a pairing piece is the partial sequence of flights until the extra flight or the associated deadhead flight) is called X' and Y' where the second pieces of these pairings are called X'' and Y'', respectively. For Type A pairing solutions we need to check whether newly formed pairings (X' - extra flight - Y'' and Y' - deadhead - X'') are feasible according to the rules mentioned above.



In Figure 4.1, BE shows the time difference between extra flight and the previous node of the pairing that should cover the extra flight (which is the last node of X' partial pairing). AE shows the time after the extra flight, but this time we consider the swapped partial pairing's (Y'') first node denoted by *fla* in this figure. We also show BE1 and BE2 (also AE1 and AE2) because our extra flight is scheduled to a time window. Therefore, during feasibility controls we need to consider this time window of the extra flight.

There are two conditions during the coverage of the extra flight. In the first one, the extra flight does not violate the feasible duty period connection and in the second one, current duty period rules are violated either by exceeding the maximum elapsed time or by the length of the connection. Therefore, we need to finish this duty period and start a new duty period.

Considering the deadhead flight associated with the extra flight, we go backwards from flight nodes *fl* to *fna* for 60 minutes, which is the preparation time for the next flight leg. As we do not have exact information about the place and the time of the deadhead, we assume that it will be in the reverse direction of the extra flight and we place the deadhead 60 minutes before the flight node *fna*. Another difference between feasibility controls of the extra flight and the deadhead is the preparation time, which is 60 minutes for the extra flight and 30 minutes for the deadhead flight. The insertion of the deadhead is illustrated in Figure 4.1. Then, we check whether the remaining time is appropriate for the feasibility of the duty period. If the swapped pairing is still feasible, we conclude that the pairings X and Y form a feasible Type A solution.

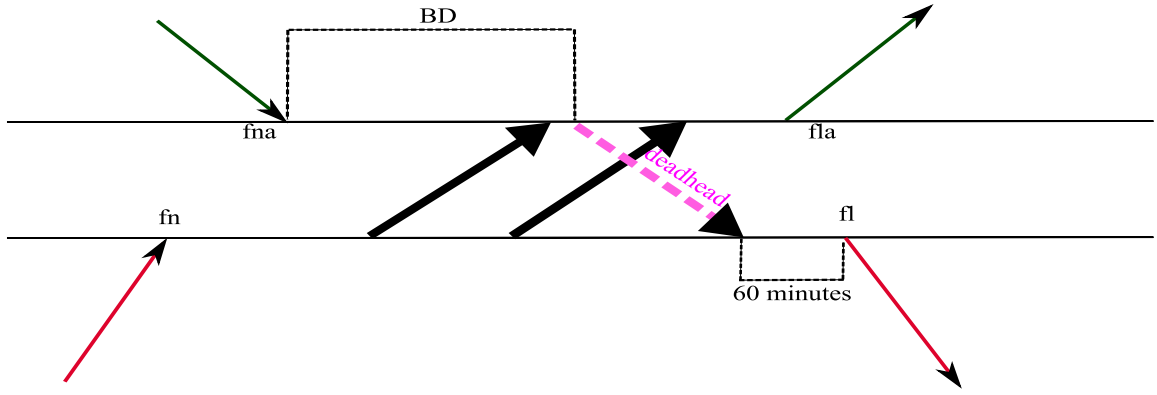


Figure 4.2: Time window controls for feasibility - deadhead.

4.2 Experimental Results

We conduct numerical study on a set of actual data acquired from a local airline company. There are two data sets with 42 and 96 flight legs, respectively. The maximum number of extra flights goes up to three in all data sets. 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 number of feasible Type A solutions found by the static and the dynamic approaches are presented in Table 4.3 and Table 4.4, respectively. The second column in Table 4.3 and Table 4.4 shows the number of extra flights in each data set.

Number of flight legs	**Number of extra flights**	**Number of feasible Type A solutions**
	1	18
42	2	118
	3	128
	1	64
96	2	128
	3	141

Table 4.3: The number of Type A solutions found by the static approach.

Number of flight legs	**Number of extra flights**	**Number of feasible Type A solutions**				
		$N = 0$	$N = 10$	$N = 50$	$N = 100$	$N = 500$
	1	0	10	14	18	18
42	2	0	10	86	98	118
	3	0	10	93	106	128
	1	10	40	64	64	64
96	2	20	80	128	128	128
	3	38	60	136	141	141

Table 4.4: The number of feasible Type A solutions found by the dynamic approach for varying N.

In Table 4.4, we compare the results within each data set for different number of extra flights and for varying N values, where N is the number of candidate Type A solution paths that is promoted during the multilabel shortest path calls. From the computational results, the importance of N can be observed. In the dynamic approach, each path may be dominated by other paths. However, by using parameter N, at least a number of candidate Type A solutions are fixed, and these solutions

candidate Type A solutions. As N increases, the chance that the candidate pairings to form Type A solution are dominated is less. The details about the parameter N can be found in [30]. After solving the multilabel shortest path, the candidate Type A solutions are moved to the fixed pool. At the end of the multilabel shortest path, we obtain many Type A candidate solutions. However, we still can not be sure about whether these pairings shall form feasible Type A solutions. Therefore, we swap these pairing pairs and check whether they are feasible.

As mentioned in Section 3, the static approach is an exact method. Therefore, the number of feasible Type A solutions are at their upper bounds. As the number of extra flights is increased, the number of feasible Type A solutions also increases. This also depends on the time ranges of the extra flights. For certain instances, the extra flights may not be covered by any type of solution.

When we compare Table 4.4 and Table 4.3, as expected, the number of feasible Type A solutions found by the static approach is the same as the number of solutions found by the dynamic approach, when N is large (for example, for the data set with 42 flight legs, when $N = 500$ the number of feasible Type A solutions found by the dynamic approach is same as the number of feasible Type A solutions found by the static approach). We also note that the individual pairings forming feasible Type A solutions obtained by both the static and the dynamic approaches are the same.

In the dynamic approach, the effect of N can be observed in detail from Table 4.5, in which the columns shown with **d** and **e** denote the number of pairings covering the deadhead and the extra flights, respectively. When we consider only one extra flight, we write 1 in the second column in the tables. If we have two or three extra flights in our schedule, we first write the total number of extra flights in the schedule. Then, we write for which extra flight the results are presented. For example, 3-2 means we have three extra flights in our schedule and we present the results for the second extra flight in that row.

When the number of extra flights increases, the decrease in the number of candidate Type A solutions for a specific extra flight can be observed in Table 4.5. For instance, consider the results for the data set with 42 flight legs and 1 extra flight; when N is equal to 10, there are 42 candidate Type A solutions covering the deadhead, but with two extra flights, there are 40 candidate Type A solutions covering the deadhead for the first extra flight. This decrease is caused by the number of total candidate Type A solutions remaining the same but some candidate Type A solutions are held also for

the second extra flight. Therefore, if N paths will be held on a flight node, when we have two extra flights, we hold some paths also for the second extra flight. Therefore, some of the candidate Type A solutions for the first extra flight cannot be held as we start holding some candidate Type A solutions for the second extra flight. Such decrease in the candidate Type A solutions can be also observed from Table 4.5 when the number of extra flights increases.

Number of flight legs	**Number of extra flights**	**Number of feasible Type A solutions**									
		$N = 0$		$N = 10$		$N = 50$		$N = 100$		$N = 500$	
		d	**e**	**d**	**e**	**d**	**e**	**d**	**e**	**d**	**e**
42	1	8	4	42	7	93	13	102	15	110	15
42	2-1	8	4	40	7	93	13	102	15	110	15
42	2-2	3	0	7	0	18	7	19	8	21	10
42	3-1	8	4	39	7	88	13	102	15	109	15
42	3-2	3	0	7	0	18	7	119	8	21	10
42	3-3	1	0	2	0	31	7	36	8	40	10
96	1	6	10	82	21	176	33	196	33	233	33
96	2-1	6	10	82	21	176	33	196	33	233	33
96	2-2	6	10	82	21	176	33	196	33	233	33
96	3-1	8	12	82	15	175	32	187	33	223	33
96	3-2	8	12	82	15	175	32	187	33	223	33
96	3-3	18	2	26	5	67	7	69	7	72	11

Table 4.5: The number of candidate Type A solutions found by the dynamic approach for varying N values.

Moreover, the optimal objective function of the conventional mathematical model (1.1) is presented in Table 4.6. These results are obtained when there is no extra flight in our schedule. Table 4.7 presents the objective function found by the static approach. Table 4.8 gives the objective function values obtained by the dynamic approach. The results in Table 4.7 and Table 4.8 are not guaranteed to be optimal. Recall that if the solution is not integer, we implement the primal heuristic mentioned in Section 3.

Number of flight legs	**Optimal objective function value**
42	4926.75
96	13660.50

Table 4.6: The objective function value of the conventional mathematical model.

Number of flight legs	Number of extra flights	Objective function value
42	1	4629.75
42	2	-154.25
42	3	-9785.25
96	1	9737.75
96	2	-3533.00
96	3	-16008.30

Table 4.7: The objective function value of the static approach.

Number of flight legs	Number of extra flights		Objective function value				
			$N = 0$	$N = 10$	$N = 50$	$N = 100$	$N = 500$
42	1	UB	6904.25	7564.75	7607.75	8070.75	7891.25
		LB	6904.25	5746.00	2942.75	2611.75	2536.75
42	2	UB	7144.25	7804.75	-4362.50	-6296.25	-11449.80
		LB	7144.25	5986.00	-8585.50	-11313.30	-16481.30
42	3	UB	7384.25	8044.75	-6112.25	-8459.50	-14170.00
		LB	7384.25	6226.00	-10335.30	-13581.60	-19291.50
96	1	UB	11778.80	9231.75	8102.00	7907.75	7907.75
		LB	11296.80	5776.25	686.50	569.25	569.25
96	2	UB	7269.25	-7249.25	-18560.50	-18754.80	-18754.80
		LB	7269.25	-7249.25	-19714.50	-19908.80	-19908.80
96	3	UB	5885.25	-2619.75	-21348.50	-22818.50	-23012.80
		LB	5885.25	-2619.75	-22502.50	-23972.50	-24166.80

Table 4.8: The number of feasible Type A solutions found by the dynamic approach for varying N.

In Table 4.8, there are two objective function values for each specific N values for each extra flight. They are the upper bound (UB), and the lower bound (LB). UB values of objective function are obtained from the primal heuristics and LB values are obtained from the linear programming relaxation. From now on, we use UB values as our objective function value, as the linear programming relaxation is not sufficient to satisfy the integrality constraints.

The decrease in the objective function values of the dynamic approach is caused by the increase in the number of feasible Type A solutions as N grows. When the swapping of the candidate pairs is feasible, they lead to a decrease in the objective function value. Besides, there is a difference between the objective function values of the static approach and the dynamic approach, when all feasible Type A solutions are found in both approaches. When solving the multilabel shortest path, some pruning rules are

static and the dynamic approaches result in different objective function values. For instance, in Table 4.9 the solutions obtained with the static approach and the dynamic approach are presented. Each row illustrates a pairing with the corresponding flight nodes. We check whether the pairings in the solution of the dynamic (static) approach are also generated by the static (dynamic) approach. We observe that during the multilabel shortest path, in the dynamic approach pairing P186 is not generated due to the pruning rules. The pruning rules used in the multilabel shortest path algorithm are detailed in [30].

Pairings chosen by the static approach					**Pairings chosen by the dynamic approach**				
P6:	4	7	17	34	P2:	40	29	3	33
P15:	13	37			P11:	26	41		
P21:	13	35	21	22	P21:	2	6		
P47:	10	16	23	36	P23:	1	32		
P130:	40	29	3	33	P24:	9	12		
P139:	26	41			P35:	4	7	15	19
P162:	2	6			P41:	8	11		
P166:	9	12			P44:	39	31		
P172:	14	18			P48:	10	16	21	22
P174:	25	24			P51:	17	34	14	18
P175:	8	11			P59:	25	24		
P177:	27	42			P75:	20	30	5	38
P180:	39	31			P83:	27	42		
P185:	20	30	5	38	P84:	28	42		
P186:	1	32	15	19	P92:	13	37		
P187:	28	42			P211:	13	35	23	36

Table 4.9: The pairings and corresponding flight legs chosen by two approaches for the data set with 42 flight legs and one extra flight.

Table 4.10 shows the costs of the solutions obtained with the static and the dynamic approaches, when these solutions are plugged into the objective function of the conventional model (1.1).Percentage gap is calculated as the difference between the objective function value of the static (or the dynamic approach) and the objective function of the conventional mathematical model, divided by the the objective function of the conventional mathematical model. As illustrated in Table 4.10, the gap increases when the number of the extra flights increases. In Table 4.3 and Table 4.4, we observe that the number of feasible Type A solutions increases as the number of extra flights increases. Therefore, both static and dynamic approaches include more pairings in their solutions. As the objective function value of the problem (3.1) in Section 3 decreases,

the total cost of the conventional mathematical model increases with the pairing pairs forming feasible Type A solutions.

Number of flight legs	Number of extra flights	Objective function value		Percentage gap	
		stat. app.	dyn. app.	stat. app.	dyn. app.
42	1	5311.25	13845.80	7.80	181.03
42	2	12951.80	27199.75	162.88	452.08
42	3	19900.50	27713.50	303.92	462.51
96	1	15261.80	34570.30	11.72	153.06
96	2	17662.50	34570.30	29.29	153.06
96	3	22218.80	34705.00	62.65	154.05

Table 4.10: The conventional mathematical model's objective function value of the solutions found by the proposed approaches.

To compare the dynamic and static approaches, the total computation time in both approaches are measured. First of all, the total computation time according to the number of extra flights are measured. Figure 4.3 and Figure 4.5 present the total computation time required for the static approach according to the number of extra flights. As the number of extra flights increases, the total time required to solve the robust airline crew pairing problem increases. For instance, for the data set for 42 flight legs the total computation time is 0.500 seconds. when there is only one extra flight. When we have three extra flights, the total time elapsed is 1.344 seconds. Figure 4.4 and Figure 4.6 represent the total computation time required for the dynamic approach according to the number of extra flights. In x-axis we have the parameter N. As the number of extra flights increases in Figure 4.4 and Figure 4.6, the total time required to solve the robust airline crew pairing problem increases. When there is no extra flight, which is the same as solving the conventional mathematical model, the total time required to solve the crew pairing problem is same as N increases.

Figure 4.7 and Figure 4.9 illustrate the relation between the total computation time required for the dynamic and the static approaches. In the figures, the total computation time required for the dynamic approach is calculated by taking the averages of the measurements of the total computation time for different values of N. Therefore, the dynamic approach is not guaranteed to end with the same number of feasible Type A solutions. If we want to compare the two approaches when they have the same number of feasible Type A solutions, we need to consider $N = 500$ (see Table 4.3 and Table 4.4). Figure 4.8 and Figure 4.10 show the result when $N = 500$.

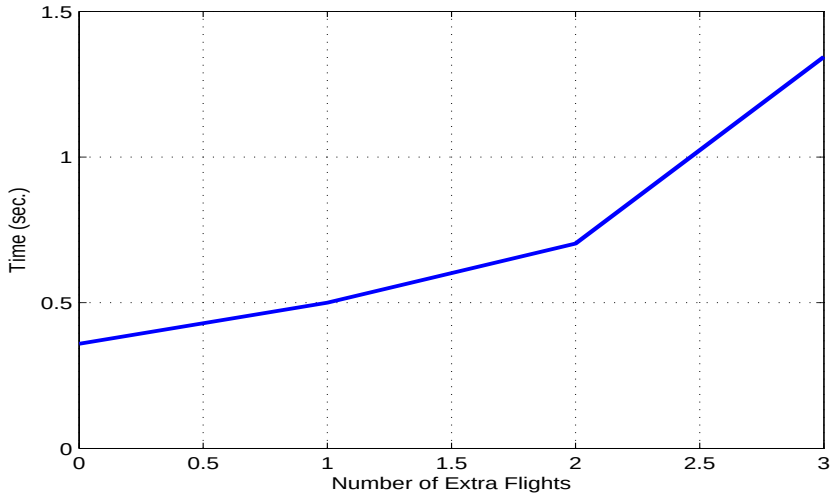


Figure 4.3: Total computation time measured for the data set with 42 flight legs vs. the number of extra flights in the static approach.

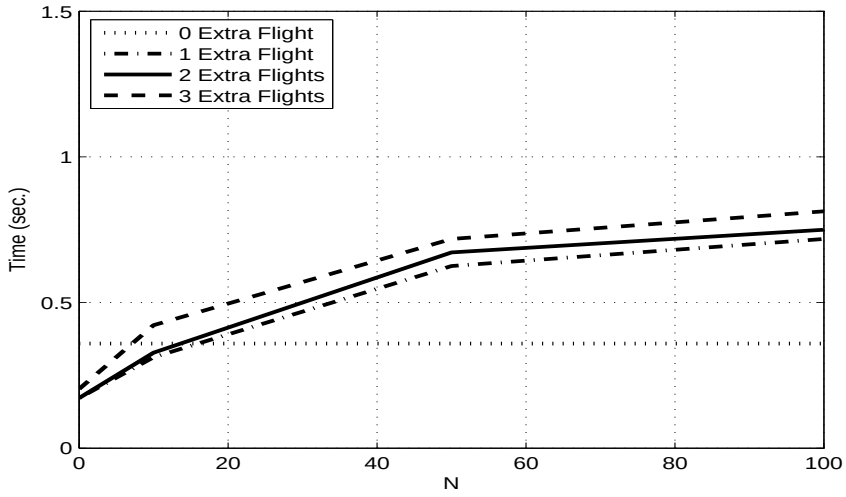


Figure 4.4: Total computation time measured for the data set with 42 flight legs vs. the number of extra flights in the dynamic approach.

In Figure 4.7 and Figure 4.9, the static approach seems to result in longer total computation times than the dynamic approach. However, this is the result of considering the averages of the measurements of the total computation time for different values of N. When a sufficiently large N value is considered for reaching the same number of feasible Type A solutions, the static approach may result with shorter total computation time (see Figure 4.8 and Figure 4.10). Only when the number of extra flights is two or three for the data set of 96 flight legs, there is a difference between the static approach and the dynamic approach. The dynamic approach seems to result in a shorter time, when the extra flight is near to the end of the schedule. However, a disadvantage of the dynamic approach is to determine the appropriate value of N that

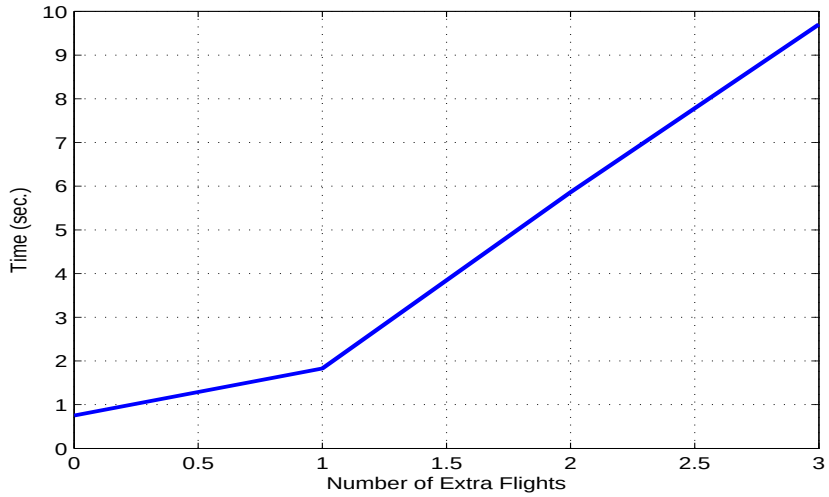


Figure 4.5: Total computation time measured for the data set with 96 flight legs vs. the number of extra flights in the static approach.

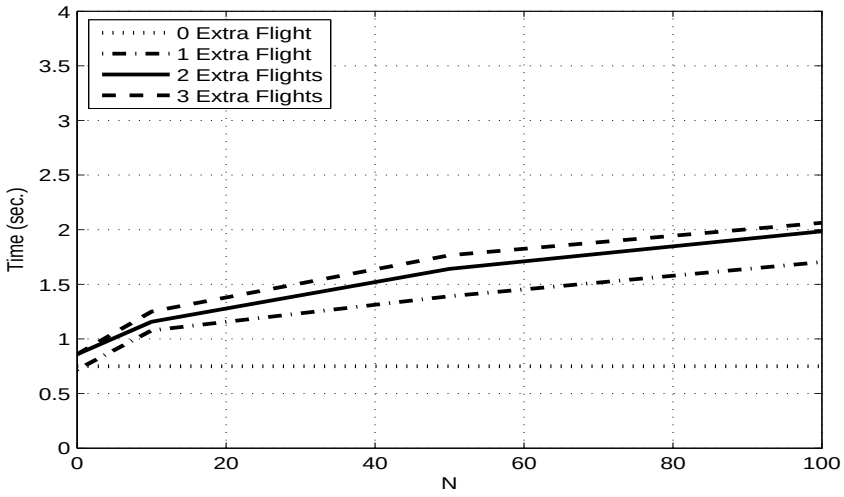


Figure 4.6: Total computation time measured for the data set with 96 flight legs vs. the number of extra flights in the dynamic approach.

would be large enough to find all feasible Type A solutions. In the static approach, we guarantee to find all feasible Type A solutions.

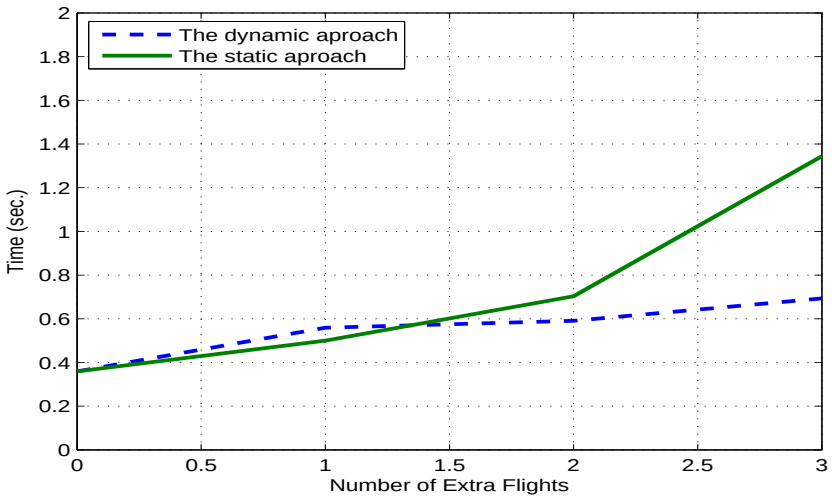


Figure 4.7: Total computation time measured for the static and dynamic approach for the data set with 42 flight legs.

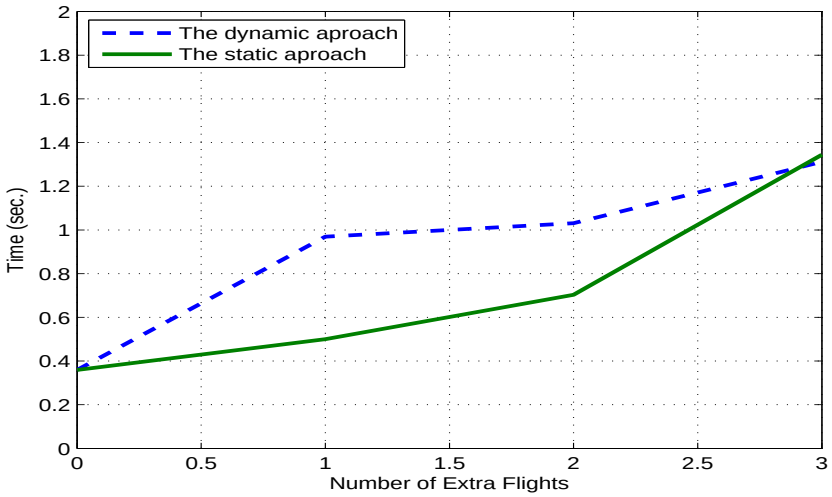


Figure 4.8: Total computation time measured for the static and dynamic approach (N=500) for the data set with 42 flight legs.

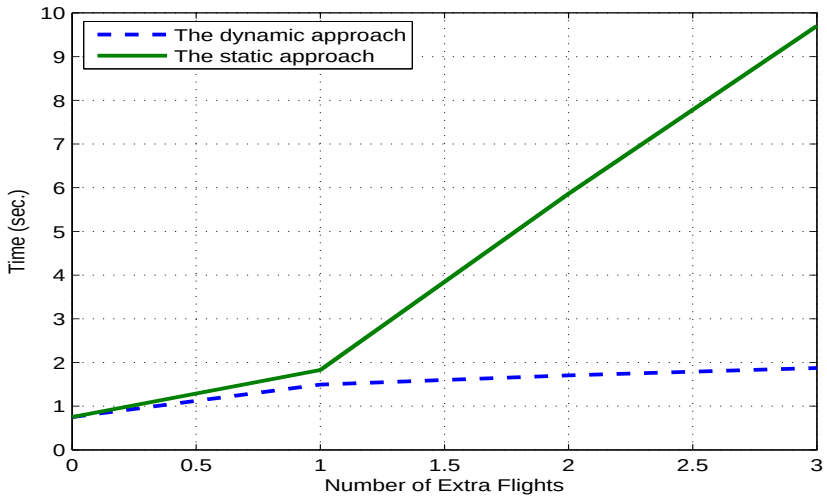


Figure 4.9: Total computation time measured for the static and dynamic approach for the data set with 96 flight legs.

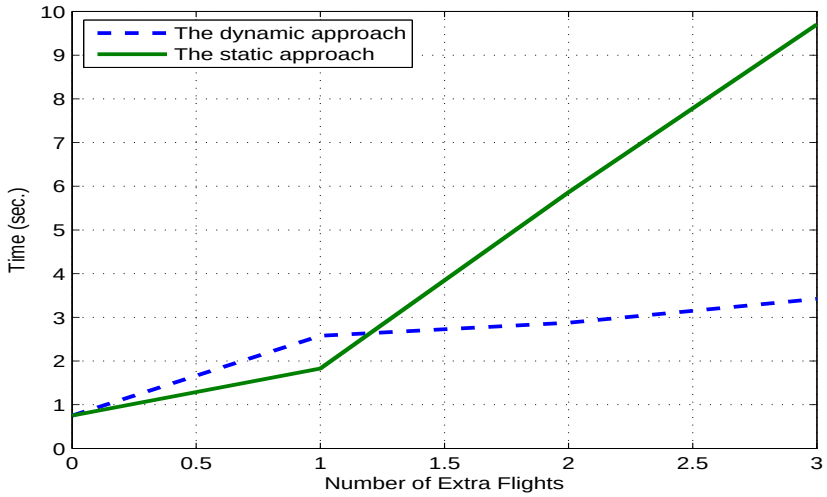


Figure 4.10: Total computation time measured for the static and dynamic approach (N=500) for the data set with 96 flight legs.

CHAPTER 5

CONCLUSION

In this study, we handle the robust airline crew pairing problem at the planning stage for managing the potential extra flights added to our regular flight schedule. We conclude that when an extra flight is added, the solution of the robust problem provides the alternative solutions for covering the extra flights without delaying or canceling any of the flights in our regular schedule.

A new robust airline crew pairing mathematical model is proposed, which is different than the conventional model in terms of the constraints and the objective function. As the number of variables is very large as in the conventional mathematical model, the column generation technique is used for solving the proposed robust model. However, we have extra column-dependent constraints in the robust model. Therefore, we also introduce two different approaches for handling the dynamic growth of the model both in the number of the constraints and the variables, the static approach and the dynamic approach, respectively.

We conducted experiments on relatively small and medium sized networks and reported our results. The dynamic approach is only a heuristic but the static approach is an exact method. Therefore, for promoting the dynamic approach to conclude with the all feasible Type A solutions, we need to introduce an auxiliary parameter. As the value of the parameter increases, the number of feasible Type A solutions may increase. However, in the static approach all feasible Type A solutions are concluded without introducing any auxiliary parameter. According to the computational results, the total time elapsed required for the dynamic approach is affected by the value of the auxiliary parameter.

Even the auxiliary parameter is large enough to conclude with all feasible Type A solutions for the small sized network, the static approach results in shorter total computation times than the dynamic approach. The dynamic approach results in better total computation times when the time range of the extra flight is far away from the

source node. As the static approach promotes all candidate Type A solutions, even for the extra flights having time ranges late in the flight schedule, we may generate more candidate pairings at the beginning of column generation method in the static approach. This holds even if these candidates do not form feasible Type A solutions. We also observe that both approaches may conclude with the same number of feasible Type A solutions, if appropriate auxiliary parameter value is chosen in the dynamic approach. For medium sized network, the dynamic approach results in shorter total computation times than the static approach. Therefore, there is a trade-off between the total computation times and the number of feasible Type A solutions found. The dynamic approach seems to result better than the static approach in both the computational running time and the number of feasible Type A solutions found, if the value of the auxiliary parameter is carefully assigned. If it is not, then extra trials may become necessary for checking if all feasible Type A solutions are found.

Besides, in this thesis we do not prevent multiple coverage (double counting) of the extra flights as one of our aim is providing as much as flexible pairings to the user. Our feasible Type A solutions may include double counting for the extra flights or the swapped pairings. However, we had some other feasible Type A solutions that are previously found but lost when we prevent multiple coverage of the extra flights. In addition, a pairing may cover more than one extra flight as a Type A solution. For instance, at the operational level the first extra flight is not realized but the second extra flight. There is a pairing which is a pairing pair of the feasible Type A solutions for both of the extra flights. However, as we prevent double counting, the pairing is assigned for covering the first extra flight. Therefore, during the operational level, the company cannot benefit from the solutions provided by the robust airline crew pairing model. Therefore, we do not prevent double counting due to provide flexibility.

In addition, there is only one crew base in our data sets. If multiple crew bases are considered, as each legal pairing should start and end at the same crew base, the starting and ending crew bases of each path during the multilabel shortest path should be tracked. One way of handling the multiple bases is generating copies of the source node and the sink node for each different crew base. We can solve the multilabel shortest path for each crew base or we can connect all source and sink nodes to another artificial source node and sink node. Another way is tracking the starting crew base of each path (candidate pairing) during the multilabel shortest path.

We can extend this research in several directions. Combining small and medium

sized networks to handle the problem of large networks is a further study. Considering the multiple bases is another extension. We also intend to study other solution techniques such as combining Lagrangian relaxation with the column generation for handling the column-dependent constraints.

Bibliography

[1] Anbil, R., Gelman, E., Patty, B. and Tanga, R., Recent advances in crew-pairing optimization at American Airlines, Interfaces, 21, 62-74, 1991.

[2] Anbil, R., Tanga, R. and Johnson, E.L., A global approach to crew pairing optimization, IBM Systems Journal, 31(1), 71-78, 1992.

[3] Anbil, R., Forrest, J.J. and Pulleybank, W.R., Column generation and the airline crew pairing problem, Documenta Mathematica, 677-686, 1998.

[4] Andersson, E., Kohl, N. and Wedelin, D., Crew pairing optimization, Operations Research in the Airline Industry, Kluwer Academic Publishers, 228-258, 1997.

[5] Barahona, F. and Anbil, R., The volume algorithm: producing primal solutions with a subgradient method, Research Report RC 21103, IBM T.J. Watson Research Center, 1997.

[6] Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P. and Vance, P.H, Branch-and-price: column generation for solving huge integer programs, Operations Research, 48(3), 316-329, 1998.

[7] Barnhart, C., Cohn, A.M., Johnson, E.L., Klabjan, D., Nemhauser, G.L. and Vance, P.H, Airline crew scheduling, Handbook of Transportation Science(2nd edition), Kluwer Academic Publishers, MA, 517-560, 2003.

[8] Bixby, R.E., Gregory, J.W., Lustig, I.J., Marstem, R.E. and Shanno, D.F., Very large-scale linear programming: a case study in combining interior point and simplex methods , Operations Research, 40(5), 885-898, 1992.

[9] Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B, Solomon, M.M. and Soumis, F., Crew pairing at Air France, European Journal of Operational Research, 245-259, 1997.

[10] Desrosiers, J. and Lübbecke, M.E., A primer in column generation, Column Generation, Springer, 1-32, 2005.

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

[12] Fisher, M.L., An applications oriented guide to Lagrangian relaxation, Interfaces, 15, 10-21, 1985.

[13] Forrest, J.J., Mathematical programming with a library of subroutines, ORSA/TIMS Joint National Meeting, 1989.

[14] Garey, M.R. and Johnson, D.S., Computers and intractability: a guide to the theory of NP-completeness, Series of books in the mathematical sciences, 1979.

[15] Gustaffson, T., A heuristic approach to column generation for airline crew scheduling, Chalmers University of Technology and Göteborg University, Department of Mathematics, Master Thesis, 1999.

[16] Held, M. and Karp, R.M., The traveling-salesman problem and minimum spanning trees, Operations Research, 18, 1138-1162, 1970.

[17] Held, M. and Karp, R.M., The traveling-salesman problem and minimum spanning trees: part II, Mathematical Programming, 1, 6-25, 1971.

[18] Held, M., Wolfe, P. and Crowder, H.P., Validation of subgradient optimization, Mathematical Programming, 6, 62-88, 1974.

[19] Klabjan, D and Schwan, K., Airline crew pairing generation in parallel, Technical Report TLI/LEC-99-02, Georgia Institute of Technology, 1999.

[20] Klabjan, D, Johnson, E.L. and Nemhauser, G.L., Solving large airline crew scheduling problems: random pairing generation and strong branching, Computational Optimization and Applications, 20, 73-91, 2001.

[21] Klabjan, D., Large-scale models in the airline industry, Column Generation, Springer, 163-195, 2005.

[22] Kohl, N., The use of linear and integer programming in airline crew scheduling, Carmen Systems, Technical Report SE-411 03, Carmen Systems, 1999.

[23] Lan, G., DePuy, G.W. and Whitehouse, G.E., An effective and simple heuristic for set covering problem, European Journal of Operational Research, 176, 1387-1403, 2007.

[24] Lavoie, S., Minoux, M. and Odier, E., A new approach for crew pairing problems by column generation with an application to air transportation, European Journal of Operational Research, 35, 45-58, 1988.

[25] Lettovský, L., Johnson, E.L. and Nemhauser, G.L., Airline crew recovery, Transportation Science, 34(4), 337-348, 2000.

[26] Lubbecke, M. and Desrosiers, J., Selected topics in column generation, Operations Research, 53(6), 1007-1023, 2005.

[27] Rosenberger, J.M., Schaefer, A.J., Goldsman, D., Johnson, E.L., Kleygwegt, A.J. and Nemhauser, G.L., SIMAIR: A stochastic model of airline operations, Proceedings of the 2000 Winter Simulation Conference, 2000.

[28] Schaefer, A.J., Johnson, E., Klegwegt, A and Nemhauser, G., Airline crew scheduling under uncertainty, Transportation Science, 39(3), 340-348, 2005.

[29] Shebalov, S. and Klabjan, D., Robust airline crew pairing: move-up crews, Transportation Science, 40, 300-312, 2006.

[30] Taş, D., Pricing in column generation for a robust airline crew pairing problem, Master Thesis, Industrial Engineering, Sabanci University, 2008.

[31] Tekiner, H., Birbil, Ş.İ. and Bülbül, K., Robust Crew Pairing for Managing Extra Flights, Computers and Operations Research, DOI:10.1016/j.cor.2008.07.005, 2008.

[32] Vance, P.H., Barnhart, C., Johnson, E.L. and Nemhauser, G.L.,Airline crew scheduling: a new formulation and decomposition algorithm, Operations Research, 45(2), 1997.

[33] Vance, P.H., Atamtürk, A., Barnhart, C., Gelman, E., Johnson, E., Krishna, A., Mahidhara, D., Nemhauser, G. and Rebello, R., A heuristic branch-and price approach for the airline crew pairing problem, 1997, http://citeseer.ist.psu.edu/vance97heuristic.html.

[34] Yen, J. and Birge, J., A stochastic programming approach to the airline crew scheduling problem, Technical Report, University of Washington, 2000.

[35] Carmen Systems, http://www.carmen.se/airlines/, July 2008.

Appendix A

First Problem Flight Data

Flight ID	Or-Des	DT-AT	Flight ID	Or-Des	DT-AT
1	IST - ESB	07:00 - 08:00	22	ADB - IST	19:20 - 20:20
2	IST - ADB	06:00 - 07:00	23	IST - ESB	17:00 - 18:00
3	ADB - ESB	10:05 - 11:20	24	ADB - IST	22:00 - 23:00
4	IST - ADA	08:25 - 09:40	25	IST - ADB	20:00 - 21:00
5	ADB - ESB	19:20 - 20:40	26	IST - ESB	19:00 - 20:00
6	ADB - IST	09:00 - 10:00	27	IST - ESB	22:00 - 23:00
7	ADA - IST	11:00 - 12:00	28	IST - ESB	22:00 - 23:00
8	IST - ADA	14:25 - 15:50	29	ESB - ADB	07:45 - 09:05
9	IST - ADB	09:00 - 10:00	30	ESB - ADB	17:00 - 18:20
10	IST - ADB	11:00 - 12:00	31	ESB - IST	08:00 - 09:00
11	ADA - IST	16:50 - 18:05	32	ESB - IST	11:00 - 12:00
12	ADB - IST	11:00 - 12:00	33	ESB - IST	14:00 - 15:00
13	IST - ESB	11:00 - 12:00	34	ESB - IST	17:00 - 18:00
14	IST - ADA	19: 00 - 20:00	35	ESB - IST	13:00 - 14:00
15	IST - ADB	13:00 - 14:00	36	ESB - IST	21:00 - 22:00
16	ADB - IST	13:00 - 14:00	37	ESB - IST	20:00 - 21:00
17	IST - ESB	13:00 - 14:00	38	ESB - IST	22:00 - 23:00
18	ADA - IST	21:15 - 22:30	39	IST - ESB	05:00 - 06:00
19	ADB - IST	15:00 - 16:00	40	IST - ESB	05:30 - 06:30
20	IST - ESB	15:00 - 16:00	41	ESB - IST	23:05 - 24:05
21	IST - ADB	17:00 - 18:00	42	ESB - IST	24:00 - 00:55

Table A.1: Flight data for the first problem.

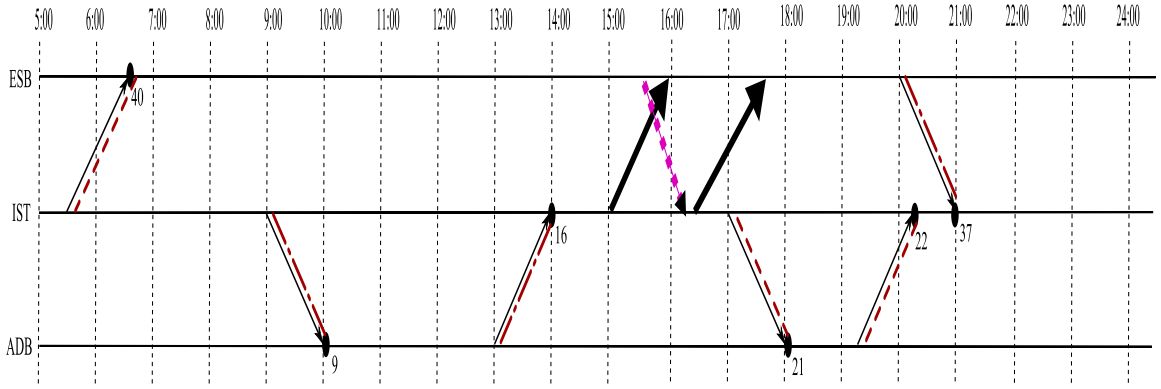


Figure A.1: One of the feasible Type A solution for the first problem.

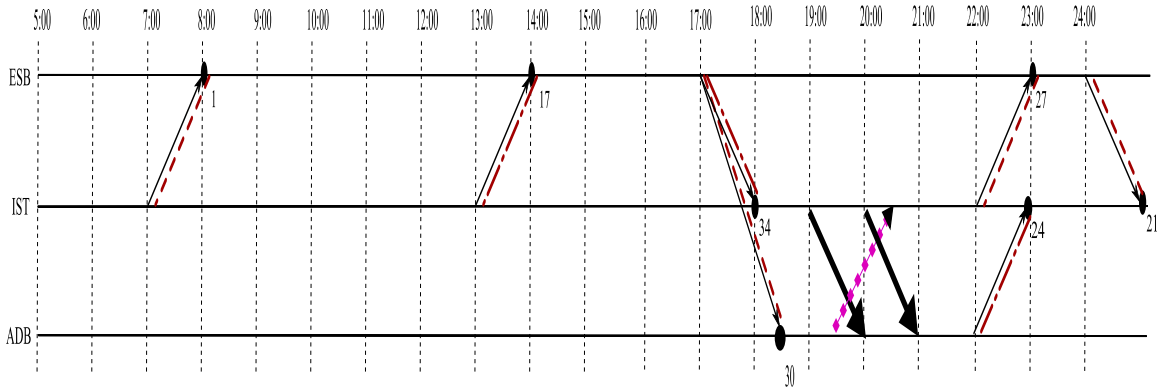


Figure A.2: One of the feasible Type A solution for the first problem.

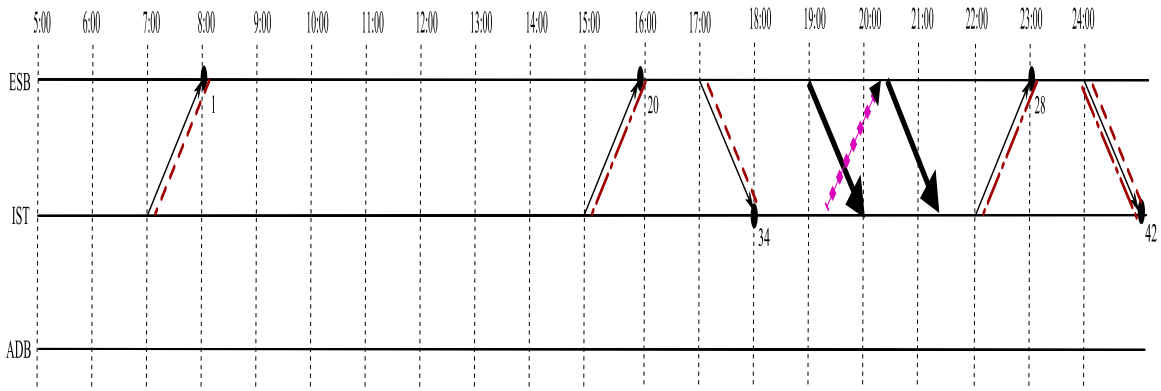


Figure A.3: One of the feasible Type A solution for the first problem.

Appendix B

Second Problem Flight Data

Flight ID	Or-Des	DT-AT	Flight ID	Or-Des	DT-AT
1	IST - ESB	04:00 - 05:05	49	IST - ADB	16:00 - 17:05
2	IST - ESB	05:10 - 06:15	50	ADB - IST	18:05 - 19:10
3	ESB - IST	04:15 - 05:20	51	IST - ADB	17:00 - 18:05
4	ESB - IST	05:30 - 06:35	52	ADB - IST	19:05 - 20:10
5	IST - ESB	06:40 - 07:45	53	IST - ADB	21:45 - 22:50
6	ESB - IST	06:00 - 07:05	54	ADB - IST	23:50 - 00:55
7	ESB - IST	07:00 - 08:05	55	ESB - ADB	05:45 - 07:00
8	ESB - IST	07:30 - 08:35	56	ADB - ESB	08:00 - 09:15
9	IST - ESB	07:00 - 08:05	57	ESB - ADB	15:00 - 16:15
10	ESB - IST	08:00 - 09:05	58	ADB - ESB	17:15 - 18:30
11	IST - ESB	09:00 - 10:05	59	ESB - ADB	20:50 - 22:05
12	IST - ESB	10:00 - 11:05	60	ADB - ESB	23:10 - 00:25
13	ESB - IST	09:00 - 10:05	61	ESB - AYT	04:15 - 05:15
14	IST - ESB	11:00 - 12:05	62	AYT - ESB	06:15 - 07:15
15	ESB - IST	11:00 - 12:05	63	ESB - AYT	19:00 - 20:00
16	IST - ESB	13:00 - 14:05	64	AYT - ESB	21:00 - 22:00
17	ESB - IST	12:00 - 13:05	65	IST - AYT	06:25 - 07:40
18	IST - ESB	14:00 - 15:05	66	AYT - IST	08:40 - 09:55
19	ESB - IST	13:00 - 14:05	67	IST - AYT	09:30 - 10:45
20	ESB - IST	14:00 - 15:05	68	AYT - IST	11:45 - 13:00
21	ESB - IST	15:00 - 16:05	69	IST - AYT	12:45 - 14:00
22	IST - ESB	15:00 - 16:05	70	AYT - IST	15:00 - 16:15
23	ESB - IST	16:00 - 17:05	71	IST - AYT	15:30 - 16:45
24	IST - ESB	16:00 - 17:05	72	AYT - IST	17:55 - 19:10
25	IST - ESB	16:15 - 17:20	73	IST - AYT	17:00 - 18:15
26	ESB - IST	17:00 - 18:05	74	AYT - IST	19:15 - 20:30
27	IST - ESB	17:00 - 18:05	75	IST - AYT	18:30 - 19:45
28	IST - ESB	18:00 - 19:05	76	AYT - IST	20:45 - 22:00
29	IST - ESB	19:00 - 20:05	77	IST - AYT	21:55 - 23:10
30	ESB - IST	19:00 - 20:05	78	AYT - IST	00:15 - 01:30
31	ESB - IST	20:00 - 21:05	79	IST - ADA	06:20 - 07:50
32	IST - ESB	20:00 - 21:05	80	ADA - IST	08:50 - 10:20
33	IST - ADB	05:00 - 06:05	81	IST - ADA	15:00 - 16:30
34	ADB - IST	07:05 - 08:10	82	ADA - IST	17:30 - 19:00
35	IST - ADB	06:00 - 07:05	83	IST - ADA	17:20 - 18:50
36	ADB - IST	08:05 - 09:10	84	ADA - IST	19:55 - 21:35
37	IST - ADB	06:40 - 07:45	85	IST - ADA	12:15 - 13:45
38	ADB - IST	08:45 - 09:50	86	ADA - IST	14:45 - 16:25
39	IST - ADB	07:00 - 08:05	87	IST - ADA	14:00 - 15:30
40	ADB - IST	09:05 - 10:10	88	ADA - IST	16:30 - 18:00
41	IST - ADB	09:00 - 10:05	89	IST - ADA	19:30 - 21:00
42	ADB - IST	11:05 - 12:10	90	ADA - IST	22:00 - 23:30
43	IST - ADB	11:00 - 12:05	91	IST - ADA	09:15 - 10:45
44	ADB - IST	13:10 - 14:15	92	ADA - IST	11:45 - 13:15
45	IST - ADB	13:00 - 14:05	93	IST - ADA	21:35 - 23:05
46	ADB - IST	15:05 - 16:10	94	ADA - IST	01:30 - 03:00
47	IST - ADB	14:00 - 15:05	95	ESB - ADA	17:30 - 18:30
48	ADB - IST	16:10 - 17:15	96	ADA - ESB	19:30 - 20:30

Table B.1: Flight data for the second problem.

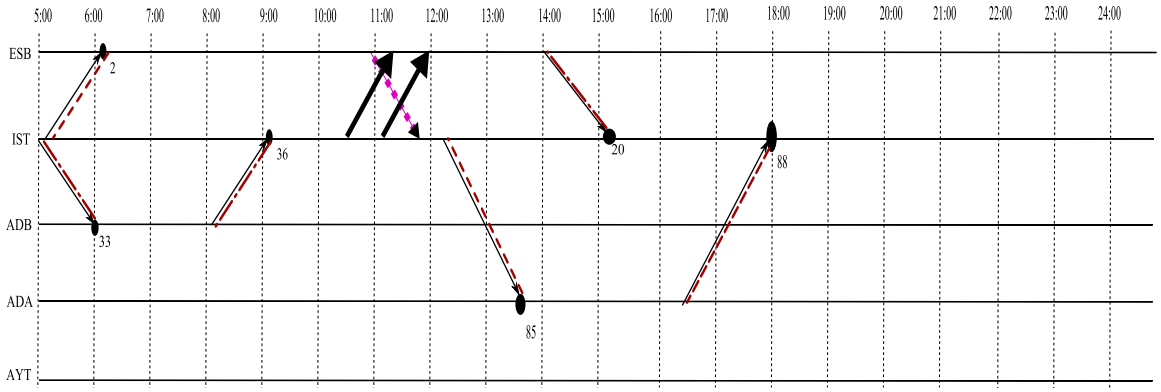


Figure B.1: One of the feasible Type A solution for the second problem.

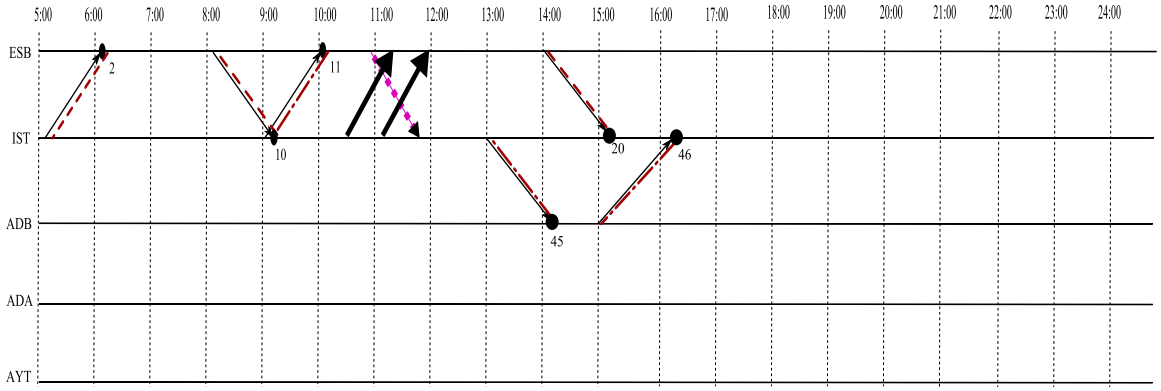


Figure B.2: One of the feasible Type A solution for the second problem.

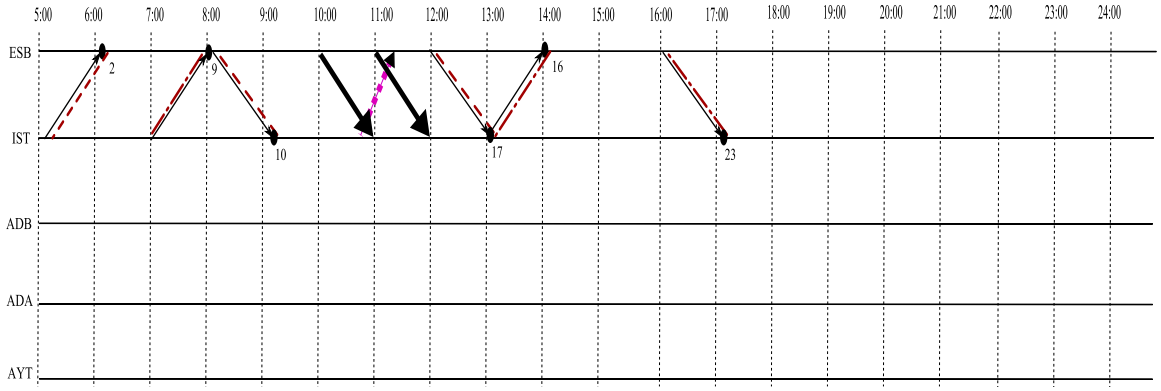


Figure B.3: One of the feasible Type A solution for the second problem.