PRICING IN COLUMN GENERATION FOR A ROBUST AIRLINE CREW PAIRING PROBLEM

by

DUYGU TAŞ

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
August 2008

PRICING IN COLUMN GENERATION
FOR A ROBUST AIRLINE CREW PAIRING PROBLEM

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. Dilek Tüzün Aksu	....
Assist. Prof. Dr. Hüsnü Yenigün	....

DATE OF APPROVAL: ....

To my family

Acknowledgments

It is a pleasure to thank the people who made this thesis possible. This study has been supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 106M472. I thank the anonymous referee for the invaluable suggestions and comments that helped me improve my thesis. I want to express my gratitude to Assoc. Prof. Dr. Ş. İlker Birbil for his enthusiastic supervision, patience and kindness besides including me in this project, believing in me, inspiring me and providing encouragement and advices throughout this thesis. Those he taught me cannot be expressed in words. I must acknowledge the invaluable contributions of Assist. Prof. Dr. Kerem Bülbül, his helpfulness, endless patience in answering my questions and providing constructive comments throughout my thesis. I am indepted to Assist. Prof. Dr. Hüsnü Yenigün for making the coding process much easier and for always being there when I needed help. I am grateful to thank Assist. Prof. Dr. Güvenç Şahin who has been actively interested in my work and has always been available to advise me/gave helpful advices, and Assist. Prof. Dr. Dilek Tüzün Aksu from Yeditepe University for her helpfulness, advices and providing us the flight schedules used by the airline companies.

I want to thank the project members İbrahim Muter for being a perfect colleague and instructor and Elvin Çoban for her friendship and coordination through this study. I also want to thank Barış Tümer for his endless IT support. I give my sincere thanks to L. Taner Tunç, Ayfer Başar, Alp E. Akçay, Birol Yüceoğlu, Nihan Özşamlı, Serkan Çiftlikli, Burak Aksu, Figen Öztoprak, Tamer Doyuran and Nurşen Aydın, the creators of the cordial atmosphere at the office.

I am very thankful to my family for their support throughout my life. Finally, I wish to express my deepest gratitude to U. Mahir Yıldırım for his support and continuous understanding, from the start to the accomplishment of this thesis and for being my source of strength and happiness in the hardest/most stressful times.

PRICING IN COLUMN GENERATION FOR A ROBUST AIRLINE CREW PAIRING PROBLEM

Duygu Taş

Industrial Engineering, Master of Science Thesis, 2008

Thesis Supervisor: Assoc. Prof. Dr. Ş. İlker Birbil

Thesis Co-supervisor : Assist. Prof. Dr. Kerem Bülbül

Keywords: scheduling, robustness, extra flights, crew pairing, column generation, multi-label shortest path problem, domination rules

Abstract

The crew pairing problem is to find the least costly set of pairings so that each flight given in the flight schedule is covered. In this study, the robust crew pairing problem is considered. That is, the selected pairings cover the regular flights and also provide solutions to cover some extra flights which may be introduced into the flight schedule during the operation at a later point in time. The crew pairing problem is usually solved by column generation in which the pricing subproblem becomes a multi-label shortest path problem. For the robust crew pairing problem the multi-label shortest path problem requires some modifications to solve two column generation approaches proposed by Çoban [10]. These modifications of the pricing problem with associated labels and the domination rules are presented.

The complexity of the multi-label shortest path problem grows exponentially as the number of flights (nodes) in the flight schedule increases. This curse of dimensionality is solved by using approximate and exact pruning rules. Also, a buffer column pool is formed as an intermediate step in order to find a negative reduced cost pairing without solving the multi-label shortest path problem at every iteration of the column generation algorithm. In the multi-label shortest path problem, the approximate rules based on the score-calculation are used for early pruning of the paths on the processed nodes. The optimal solution may be missed because of the coarse structure of the approximate rules. When a pairing that improves the objective function cannot be found by applying the approximate rules, we switch to the exact pruning. Another method is using a hybrid approach that applies both approximate and exact rules in the same iteration to find the optimal solution. The performance of our solution approach is demonstrated through a computational study by using actual data from a local airline.

DAYANIKLI EKİP EŞLEME PROBLEMİNDE KOLON TÜRETME YÖNTEMİNİN ÜCRETLENDİRİLMESİ

Duygu Taş

Endüstri Mühendisliği, Yüksek Lisans Tezi, 2008

Tez Danışmanı: Doç. Dr. Ş. İlker Birbil

Yardımcı Tez Danışmanı: Yard. Doç. Dr. Kerem Bülbül

Anahtar Kelimeler: çizelgeleme, dayanıklılık, ekstra uçuşlar, ekip eşleme, kolon türetme, çok takılı en kısa yol problemi, baskı kuralları

Özet

Ekip eşleme problemi uçuş çizelgesindeki her bir uçuşun kapsanmasını sağlayacak şekilde en az maliyetli eşleme kümesinin bulunması problemidir. Bu çalışmada, dayanıklı ekip eşleme problemi ele alınmıştır. Bu problemde, seçilen eşlemeler olağan uçuşları kapsamakta ve operasyon sırasında tanıtılabilecek bazı ekstra uçuşların kapsanmasını da sağlamaktadır. Ekip eşleme problemi genellikle kolon türetme yöntemiyle çözülmektedir ve bu yöntemin alt problemi çok takılı en kısa yol problemi olmaktadır. Dayanıklı ekip eşleme problemi için Çoban [10] tarafından önerilmiş olan iki model bulunmaktadır ve çok takılı en kısa yol probleminde bu modellerin çözümü için bazı değişiklikler gerekmektedir. Ücretlendirme problemindeki bu değişiklikler, ilişkilendirilmiş takılar ve baskı yöntemleriyle beraber aktarılmıştır.

Çok takılı en kısa yol probleminin karmaşıklığı, çizelgedeki uçuş (düğüm) sayısı arttıkça üssel bir şekilde artmaktadır. Bu durum iki yaklaşık ve bir pekin kural kullanılarak çözülmektedir. Ayrıca, çok takılı en kısa yol problemini çözmeden uygun bir eşleme bulabilmek için ara kolon havuzu oluşturulmuştur. Çok takılı en kısa yol problemini çözerken, işlenen düğüm üzerindeki yolları ilk olarak temizlemek için puan hesaplamaya dayalı olan yaklaşık kurallar kullanılmaktadır. En iyi çözüm yaklaşık kuralların kaba yapısından dolayı kaçırılabilir. Eğer yaklaşık kurallar kullanılarak amaç fonksiyonunu geliştirecek bir eşleme bulunamazsa, uygulanan kural pekin kural olarak değiştirilmektedir. Diğer bir yaklaşım, hem pekin hem de yaklaşık kuralların aynı iterasyonda kullanıldığı melez yöntemdir. Bu yöntemde de eniyi sonuç bulunabilmektedir. Yerel bir havayolu şirketinden alınan veriler doğrultusunda bu çözüm yaklaşımlarının sayısal sonuçları sunulmuştur.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Crew scheduling is one of the planning phases that is put into practice by airline companies. This phase is more challenging in the airline industry than in other transportation sectors, because the rules that are applied at the operational level are much more complex and the cost paid to the crew is very high in the airline industry. The input of the crew scheduling problem is the set of flights. Flights are taken into consideration as two separate segments: domestic flights and international flights. Also, the crew is branched off into two parts as cockpit crew and cabin crew. This separation is compulsory because each crew personnel can serve for certain types of fleet. The crew scheduling problem consists of crew pairing and crew assignment problems. The crew pairing problem is to find the least cost set of feasible pairings that start and end at the same crew base and cover all flights given in the flight schedule. The solution of the crew pairing problem is used by the crew assignment problem. In this problem, the schedules of each crew members are constructed by assigning the crew members to the pairings that are generated in the crew pairing problem.

In this study, we deal with the crew pairing problem. The objective function of this problem is the minimization of total cost. The crew pairing problem can be modeled as a set partitioning or a set covering problem. The set covering problem is given by

$$\begin{aligned} \min \quad & \textstyle\sum_{j\in\mathcal{P}} c_j x_j \\ \text{s.t.} \quad & \textstyle\sum_{j\in\mathcal{P}} a_{ij} x_j \geq 1, \quad i \in \mathcal{F}, \\ & x_j \in \{0,1\}, \end{aligned} \tag{1.1}$$

where P is the set of all feasible pairings, F is the set of all flights, c_j is the cost of the pairing j, a_{ij}=1 if flight i is covered by pairing j and 0, otherwise. If pairing j is selected by the solution, its corresponding variable x_j=1 and 0, otherwise. The crew pairing problem can also be modeled as a set partitioning model where the first constraint is changed as $\sum_{j \in \mathcal{P}} a_{ij}x_j = 1$. Flights are covered exactly once by the set partitioning problem, whereas they are allowed to be covered more than once by the set covering problem. This second situation is referred to as deadheading. In a deadhead flight, one crew is charged to cover this flight and other crews fly as passengers to reach the arrival destination of that flight and to continue their pairings without any disruption. In the LP relaxation of these models, the last constraint $x_j \in \{0, 1\}$ is changed to $x_j \geq 0$. Our study focuses on the LP relaxation of the set covering model of the crew pairing problem.

Generating all feasible pairings in the crew pairing problem can be costly even if the number of flights is small because the number of pairings (variables) of the problem (1.1) is very large. Therefore, a column generation method is usually applied to the crew pairing problem as a solution approach. In this method, a restricted master problem is solved at each iteration. In the restricted master problem, we only include a subset of all possible pairings in problem (1.1). At each iteration pricing is applied and a pairing with a negative reduced cost is searched. If such a pairing is found, it is added to the restricted master problem and the problem is reoptimized. If a pairing that improves the objective function cannot be found, the optimal solution is reached. The pricing subproblem corresponds to a multi-label shortest path problem in the crew pairing problem. Several labels are tracked to apply both feasibility and domination rules and to calculate the cost of the pairing.

1.1 Contributions of This Study

Airline companies sometimes need to add extra flights into the regular flight schedule. This requirement may appear because of seasonal changes. For example, the number of passengers that go to certain vacation places increases during summer. Also, there may be special demand to specific destinations by sportsmen, businessmen, companies and so forth. The exact information of the extra flights are not known at the planning phase. Airline companies can anticipate them from their past experiences. It can be said that the possible extra flights are known during the planning phase. The certainty of the extra flight becomes clear at the operational level of the flight schedule. Therefore, the anticipated (possible) extra flights are not inserted into the flight schedule and not considered as regular flights for the crew scheduling problem. When an extra flight is needed, airline companies try to cover this flight even by canceling their regular flights or hiring extra crew with high cost. In this thesis, the robust crew pairing problem and two previously proposed models are considered [10]. In the robust crew pairing problem, the pairings that are selected by the solution of the crew pairing problem cover all regular flights in the flight schedule and also possible extra flights can be covered by these selected pairings. The coverage of possible of extra flights is explained in Section 3.4 in detail. We try to maximize the number of pairings that can cover extra flights while the total cost is minimized. There is a possibility that these extra flights may not be added to the schedule at the operational level. Therefore, pairings should be feasible for both the flight schedule with regular flights solely and the flight schedule with regular and extra flights. Also, the changes of the pricing subproblem, which is altered for each proposed robust crew pairing model, are explained in this thesis.

In this study, the flight network is generated in order to solve the crew pairing problem. The properties of this network are explained in Chapter 3. The pricing subproblem of the crew pairing phase is a multi-label shortest path problem. There are several labels (total flying time, total elapsed time, cost and so forth) that are being tracked throughout the pairing. The labels are kept at every node to check the feasibility of each connection of that node. Also the cost of the pairing, which depends on labels such as flying time and elapsed time, is calculated by using these labels. Multi-label shortest path problem keeps several labels, so there might be several paths on every node. The complexity of the multi-label shortest path problem is exponential in the number of fligths in the flight network. Therefore, a proper pairing is searched by checking the pairings in the buffer column pool initially. The properties of the buffer pool is explained in Section 3.2. If all pairings in the buffer column pool have nonnegative reduced costs, the multi-label shortest path algorithm is started. To solve the multi-label shortest path problem in a reasonable execution time, we propose the following rules in this thesis to prune some paths on processed nodes:

- While a pairing with negative reduced cost is searched, initially approximate rules are applied to each node. There are two approximate rules which are explained in Sections 3.3.1 and 3.3.2. At first, approximate rule 1 is applied to the paths of the processed node. If the number of remaining paths is still large, then the approximate rule 2 is applied to that node.

- If a pairing with negative reduced cost cannot be found with approximate rules, the rule is switched to the exact rule. Exact rule uses upper and lower bounds on dual and cost values in order to prune paths which result in pairings with nonnegative reduced cost. The number of paths fathomed by the exact rule is smaller than the number of paths fathomed by the approximate rules, but the optimal path is never missed.

- Another way of finding a pairing with negative reduced cost is the hybrid approach of exact and approximate rules. In this method, both exact and approximate rules are used in same iteration.

1.2 Outline

The outline of this thesis is as follows. Chapter 2 gives the literature review of the crew pairing problem, most commonly used methodologies and approaches. Chapter 3 gives our problem definition, rules that are applied before domination checks and the modifications required for the models proposed for the robust crew pairing problem. The numerical study and computational results are given in Chapter 4. Finally, we present our conclusions and point future research directions in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

The airline scheduling problem consists of four main problems. These problems, as shown in Figure 2.1, are considered separately because of their complicated structures. Klabjan [19] provides the models for the planning processes of the airline companies. The schedule planning, fleet assignment and the aircraft routing models are thoroughly explained by Klabjan. Desaulniers *et al.* [11] consider the daily aircraft routing and scheduling problem. The objective of the daily aircraft routing and scheduling problem is to maximize the expected revenue while constructing daily schedules for the aircraft fleets. These fleets may be heterogeneous because they may have different types of aircraft. All flight legs given in the aircraft schedule with their durations must be covered by these fleets. In addition to the duration, the departure/arrival times and the stations of the flight legs are known. For the daily aircraft routing and scheduling problem, two models are provided by Desaulniers *et al.* which are the set partitioning and time constrained multicommodity network models.

The output of each step in Figure 2.1 is the input to the next step. The planning processes start with the *flight scheduling problem* in which the timetable of the flights are constructed. The solution of this problem is the flights flown by the airline company for the given time period. The *fleet assignment problem* is the allocation of aircraft to the flights according to the predicted demand and the size of the aircraft. The aim of the *maintenance routing problem* is to provide time for the compulsory checks of the aircraft.

The fourth step of the planning processes is the *crew scheduling problem.* In the crew scheduling problem there are differences in the types of the flights or in the qualifications of the crew members. The flights given in the schedule are divided

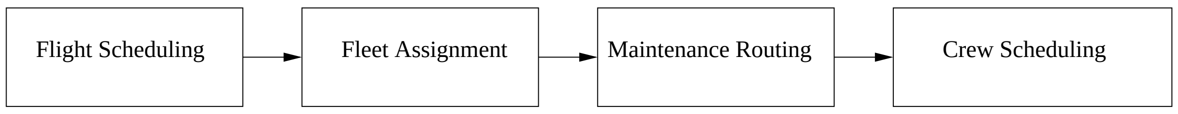

Figure 2.1: Planning processes of the airline companies.

into two categories, domestic and international flights. The main differences between these two categories are the following:

- The number of domestic flights is usually larger than the number of the international flights.
- Deadhead crews are mostly used by international flights because these flights are infrequent in the flight schedule. We refer the reader to Barnhart *et al.* [4] for the deadheading problems of the international flights.
- Domestic flights can be operated on the *daily schedule* whereas international flights are usually operated on the *weekly schedule.*

Besides the above differences, there is another categorization of the crew scheduling problem. The crew scheduling problem of the cockpit crew differs from the cabin crew. The cockpit crews are qualified to fly specific fleets, whereas the cabin crew can serve on different fleets. In this thesis, we study only the crew pairing problem for cockpit crews. Therefore, this problem is decomposed into fleets. That is the solution is provided for each fleet given in the schedule.

2.1 Definitions and Structures

Within this section, we heavily make use of the studies of Vance *et al.* [28] and Barnhart *et al.* [6] to give the definitions of the standard terms used in airline problems.

A nonstop flight is called as *flight leg* or *segment.* A flight leg can be connected to another flight leg, if the arrival station of the first flight leg is the same as the departure station of the following leg and the time between these two flights is adequate to satisfy the rules. In that way a sequence of flights, called as *duty period*, is constructed as illustrated in Figure 2.2. The flights in a duty period are flown by the single crew, and mostly the crew members do not change through a duty period. In a duty period, the time between two flight legs is called as *sit time*.

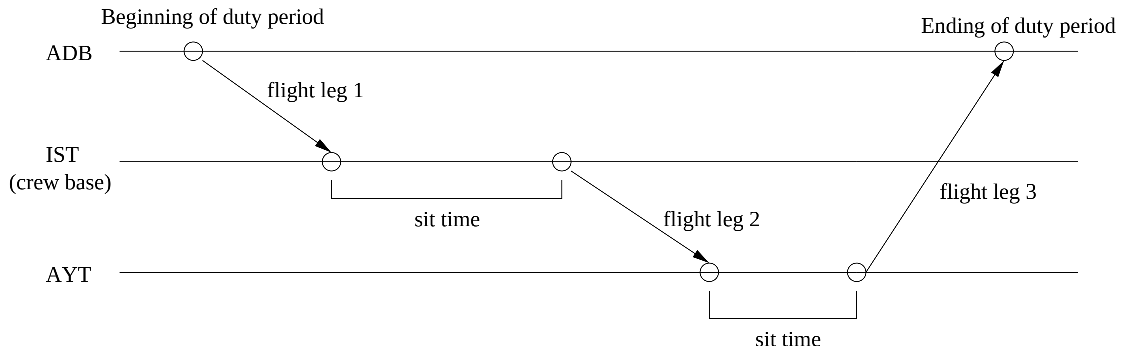

Figure 2.2: The sequence of flights in a duty period.

The sequence of several duty periods is called as *pairing*, if the sequence starts and ends at the same *crew base*, which is the city of the crews' domicile. The departure airport of each duty period must be the same as the arrival airport of the previous duty period. In a pairing, the duty periods are separated by *rest time*, or *layover*. These definitions are shown in Figure 2.3. A *schedule* is the sequence of pairings that is constructed for each individual crew member. The time between two pairings in the schedule is called as *time off*.

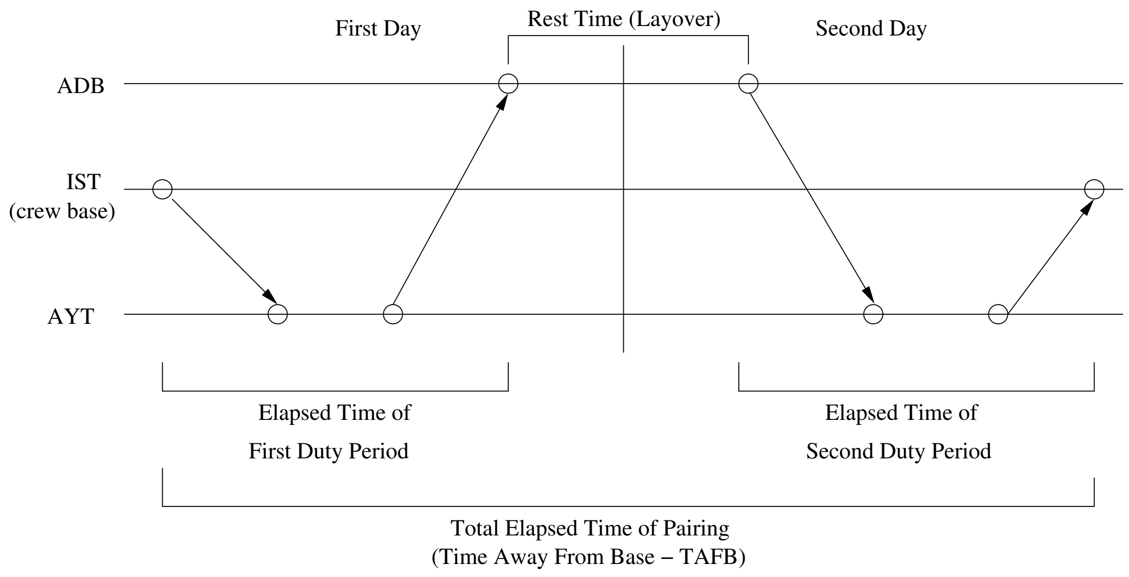

Figure 2.3: Some definitions related to a pairing.

Another widely used term is the *deadhead flight*. The deadhead flights are used for relocation of the crew members. In a deadhead flight, the considered crew members fly as passengers. Consider the illustration in Figure 2.4. The flight network has five flights. To cover all flights two pairings are formed:

- The first pairing covers flight 1, 3 and 4.
- The second pairing covers flight 2, 3 and 5.

In this scenario, flight 3 is flown by one crew and the other crew flies as passengers. The deadhead flight is required to reposition the crew to continue its pairing. Notice that the deadhead flights may be necessary to cover all flights given in a flight schedule.

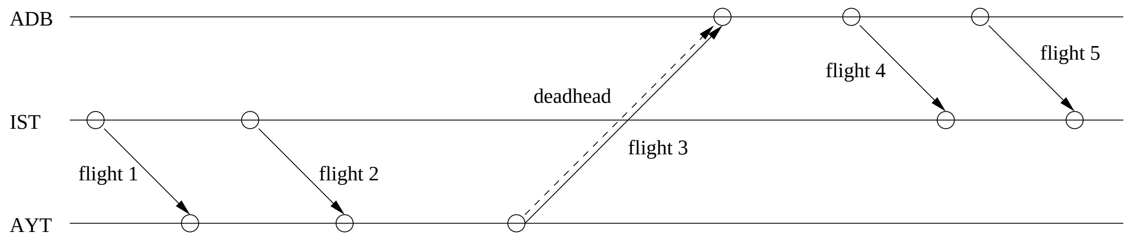

Figure 2.4: A deadhead flight.

To construct the feasible duty periods and pairings, there are several feasibility rules such as the governmental regulations and the collective agreements. The main rules associated with a duty period are as follows:

- The time between two flights in a duty period is restricted by the *minimum sit time* and the *maximum sit time*. This idle time should be greater than or equal to the minimum sit time and less than or equal to the maximum sit time.
- The total elapsed time of a duty period should be less than or equal to the *maximum elapsed time* defined for the duty.
- Similarly, the total flying time of a duty period is restricted by the *maximum flying time*.

The main rules associated with a pairing are as follows:

- The starting and ending station of a pairing should be the same and this station should be the crew base.

- The number of duty periods in a pairing is limited by the *maximum number of duty periods* defined for the pairing.
- The time between two duty periods is restricted by the *minimum rest time* and the *maximum rest time* in a pairing.
- The total elapsed time of a pairing should be less than or equal to the *maximum time away from base.*
- There is also a rather complicated rule known as the *8-in-24 rule.* This rule indicates that extra rest time should be given to a crew, if the pairing flown by this crew consists of more than 8 hour flying time in a 24 hour period.

The cost for a duty period is expressed in minutes and it is obtained by

$$c_d = max\{f_d * elapse, flying, minGuar\}, \tag{2.1}$$

where f_d is the fraction of the duty period, *elapse* is the total elapsed time of the duty period, *flying* is the total flying time in the duty period, and *minGuar* is the *minimum guaranteed* number of hours of the duty period which is the payment to the crew by the company. While the cost is calculated, *minGuar* is expressed in minutes.
The cost of a pairing is also expressed in minutes and it is given by

$$c_p = max\{f_p * TAFB, \sum_{d \in p} c_d, ndp * minGuar\}, \tag{2.2}$$

where the cost of a pairing is the maximum of three quantities:

- The first quantity is the fraction of the total time-away-from-base (TAFB) of the pairing which is calculated by multiplying the total TAFB with the fraction f_p defined for the pairing.
- The second quantity is the sum of the costs of the completed duty periods in the pairing.
- The third quantity is the *minimum guaranteed* number of minutes of the pairing, where *ndp* represents the number of duty periods in the pairing.

2.2 Crew Scheduling

It is common in crew scheduling to decompose the complex overall problem into manageable subproblems. Figure 2.5 shows a typical decomposition for the crew scheduling problem into two subproblems: the *crew pairing problem* and the *crew assignment problem.*

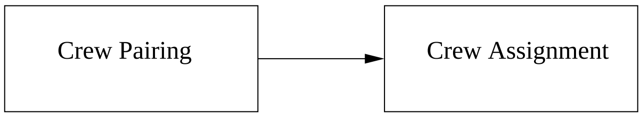

Figure 2.5: The division of the crew scheduling problem.

The crew pairing problem is to find the set of the pairings that has the minimum total cost and covers all flights given in the flight schedule. Solving the crew pairing problem is difficult because there are several feasibility rules and the cost of the pairing is nonlinear. Therefore, this problem is considered by solving three different problems: *daily*, *weekly* and *transition problems.* In the daily problem, it is assumed that each flight is flown every day. The flights which are repeated at least four days in a week are treated as though they are repeated every day. However, the flights, which are not flown on particular days of a week, may lead to infeasibilities in the pairing (*broken pairings*). At this point, the weekly problem is solved to restore the feasibility in the daily problem. The weekly problem usually requires deadhead flights to generate feasible pairings. The assumption in this problem is that the flights are repeated every week. But, there may be holidays or other changes in the schedule, such as adding extra flights, which cause the differences between weeks. Therefore, the transition problem is solved to correct the infeasibility in the weekly problem. In Andersson *et al.* [3], the details about the three stages of the crew pairing problem can be found.

As shown in Figure 2.5, the solution of the crew pairing problem is sent to the crew assignment problem. This problem assigns individual crew members to the pairings generated by the crew pairing problem. In the crew assignment problem, a set of schedules is constructed to cover all pairings. This is similar to the crew pairing problem, in which the set of the pairings is constructed to cover all of the flight legs given in the flight schedule. The crew assignment problem is usually solved in two stages. In the first stage, the specific activities such as annual leave or rest times are assigned to each crew member. Then in the second stage, the schedules are formed by assigning the crew members to pairings and determining the time-off periods of each member. The aim of the crew assignment problem is to minimize the assignment of the supplementary crew members to the pairings. If a pairing cannot be covered by the regular crew members, then supplementary crew members are assigned to that pairing and this causes an increase in the payment. We refer the reader to Gamache *et al.* [17] for the crew assignment problem, its mathematical formulation and a solution method.

The main differences between the crew pairing and the crew assignment problems are as follows:

- In the crew pairing problem, a pairing is generated for a single crew. This implies a single assignment. However, in the crew assignment problem, each crew member is assigned individually to the generated pairings.
- The objective function of the crew pairing problem is to minimize the total cost while ensuring that all flights are covered. The aim of the crew assignment problem, on the other hand, is to meet the needs of the crew members and to maximize the number of pairings that are covered by the regular crew members.

2.3 Solving LP Relaxation

We focus on the crew pairing problem by solving the LP relaxation of the set covering problem (1.1). In the literature, there are different studies for the LP relaxation of the crew pairing problem. Anbil *et al.* [1] propose a global approach called SPRINT. This approach is applied to the crew pairing problem, along with another approach referred to as TRIP. TRIP is based on subproblem optimization but it is not capable of improving the solutions. Therefore, only suboptimal solutions are provided. However, SPRINT can find the optimal solution of the LP relaxation of the crew pairing problem. This approach starts by constructing a subproblem with a subset of the columns and solving this subproblem to get optimal dual values, which are used to calculate the reduced costs of the columns. Then a new subproblem is constructed by taking the set of basic columns of the optimal solution and selecting some good columns according to their reduced costs. The large problems can then be solved optimally by solving a small number of subproblems. For example, the solution of a problem with 5.5 million columns is found by solving 25 subproblems. Bixby *et al.* [7] suggest another approach, in which both the interior point and the simplex methods are used. The main process operated in this study is called *sifting*. This method is similar to the column generation approach and it is firstly proposed by Forrest [16]. To find the optimal solution, a combination of the interior point and the simplex methods are applied to the LP relaxation of the resulting set partitioning problem.

A well-known method to solve the LP relaxation of the crew pairing problem is the column generation. In the column generation method, if the crew pairing problem (1.1) contains all feasible pairings, then it is called the *master problem*. The *restricted master problem* denotes the problem with a subset of all feasible pairings. The main steps of the column generation method are as follows:

- The problem starts with the subset of the columns to form the restricted master problem. Then, the restricted master problem is solved to find the optimal solution.

- The columns that may improve the solution of the restricted master problem are generated by solving the pricing problem. If there are no such columns, then the optimal solution of the LP relaxation is found and the algorithm is stopped.

- The generated columns are added to the restricted master problem and then the restricted master problem is reoptimized.

Crainic and Rousseau [9] use the column generation method to solve the crew pairing problem. As mentioned in Chapter 1, the crew pairing problem can be formulated as a set partitioning or as a set covering model. The difference between these models is that the deadhead flights are not allowed in the set partitioning model, whereas the set covering formulation can generate pairings with deadheads. Crainic and Rousseau focus on the set covering formulation of the crew pairing problem, where the number of variables may be large even for a small number of flights.

Therefore, the column generation method is used and an algorithm is proposed. The main procedure of the algorithm is to generate the pairings consecutively with an increasing number of duty periods.

Another study, in which the column generation method is applied to the crew pairing problem, is Anbil *et al.* [2]. They propose an algorithm, which is based on both column generation and SPRINT. Moreover, Desaulniers *et al.* [12] apply the column generation method to crew scheduling problems. The column generation method is also applied to the crew rostering problems. Gamache *et al.* [17] model the crew rostering problem as a set partitioning problem and solve this problem by column generation.

The problems with large number of variables can be solved by the *branch-and-price* algorithm. The branch-and-price is a kind of branch-and-bound method, which permits column generation algorithm to be applied at each node of the tree. Savelsbergh and Sol [22] solve the set partitioning problem by applying the branch-and-price method. As mentioned above, the column generation method is performed throughout the search tree. In the column generation method, the pricing problem is solved to find the columns that may improve the objective function. If such columns are found, they are added to the linear program and then the linear program is reoptimized. If such columns cannot be found and the solution of the problem is fractional, then the branching part of the method is started.

For the airline crew pairing problem, Vance *et al.* [28] apply the branch-and-price method. They provide near optimal solutions because the pricing problem of the column generation algorithm is solved approximately. In the literature, there are several studies which focus on the algorithms defined for the branch-and-price method. Barnhart *et al.* [5] give general models by merging these studies. They also present two examples, which are the general assignment and the crew scheduling problems, in order to explain the basic procedures of the branch-and-price method.

2.4 The Pricing Problem

When the column generation method is applied to the LP relaxation of the crew pairing problem, the pricing problem corresponds to the multi-label or constrained shortest path problem (more on this in Chapter 3). The constrained shortest path problem is solved at each iteration to generate pairings that are added to the restricted master problem to improve the solution. There are several studies which focus on the constrained shortest path problem especially for the vehicle routing problems. Desrochers and Soumis [13] propose a dynamic programming algorithm for the shortest path problem with time windows (SPPTW). This problem is to find the path with minimum cost from source node to sink node by satisfying the time windows defined for each node on the graph. Desrochers and Soumis construct their algorithm by adjusting the Ford-Bellman-Moore dynamic programming algorithm, which is appropriate for the regular shortest path problem. Desrochers *et al.* [14] focus on vehicle routing problems with time windows (VRPTW). This problem is a type of vehicle routing problem in which the time windows, denoting the allowed service times of the customers, are also considered. Desrochers *et al.* propose a new algorithm, which formulates the VRPTW as a set partitioning model and apply the column generation method to the LP relaxation of this model. In this algorithm, the columns that are added to the restricted master problem are generated by solving the shortest path problem with time windows and capacity constraints.

Nagih and Soumis [21] propose an approach for the shortest path problem with resource constraints. The objective in this problem is to find the path with least cost between the source and the sink nodes by satisfying the constraints defined for each resource. The dynamic programming algorithm can be applied to this problem but as the number of resources increases, the time required to solve the problem also increases. Therefore, Nagih and Soumis propose a heuristic method for dynamic algorithm, which reduces the size of the space constructed by the resources. This reduction is based on concatenation of the resources into a vector and is used to decrease the time consumed at the part where domination rules are applied to the paths. The projected space should be calibrated by using Lagrangian and surrogate relaxation methods to find near optimal solutions.

Desrosiers *et al.* [15] present dynamic programming algorithms for time and resource constrained shortest path problems, which are the subproblems of many scheduling problems. The objective function of the shortest path problem with time windows is to minimize the traveling cost. Moreover, the time variables that are determined by the solution of the model should satisfy the time intervals specified on each node. Each path that goes through a node is denoted by two labels, time and cost. The label correcting algorithm, which is the first algorithm in [15], depends on the node treating method. All paths of a treated node are carried to its successors through the arcs that connect these nodes. Whenever a predecessor of a node is treated, new paths are added to the set of paths on that node by applying this carrying operation. In the label setting algorithm, label treating is applied instead of the node treating method. This algorithm works even if the cost parameters on the arcs or cycles in the graph are negative. In this algorithm, the path that has the minimum time is selected and then it is carried to the successor nodes. In the shortest path problem with resource constraints, each path on the nodes are represented with $n + 1$ labels, where n is the number of resources and one label is reserved for the cost of that path. The calculation of cost is based on the consumption of resources. Desrosiers *et al.* propose the dynamic programming algorithm of pulling type which depends on determining the nondominated paths on every node. The definition and application of dominancy is explained thoroughly in Chapter 3.

In the literature there are several studies that focus on the label correcting algorithm. Skriver and Andersen [25] propose a label correcting algorithm to solve the bicriterion shortest path problem that appears as the subproblem in the transportation and the scheduling problems. The classical shortest path problem has a single objective which is the minimization of the total cost or the total traveling time. The bicriterion shortest path problem considers two objectives at the same time, such as the minimization of the total cost and the total traveling time. Therefore, solving the bicriterion shortest path problem is more difficult than solving the shortest path problem with a single objective. The label correcting algorithm suggested by Brumbaugh-Smith *et al.* [8] is improved by Skriver and Andersen. In the algorithm that is proposed by Brumbaugh-Smith *et al.* multiple labels are used for the bicriterion shortest path problem. Guerriero and Musmanno [18] focus on the multicriteria shortest path problem, in which more than two objectives are considered. They examine several label correcting methods to find the nondominated paths from source node to all other nodes. These methods are based on either the *node-selection* or the *label-selection* algorithms. Random networks are generated to test these algorithms. Their results show that for the networks with high density, the performance of the label-selection methods is better than the performance of the node-selection methods. They also suggest parallel computing to create productive algorithms.

In the crew pairing problem, pairings that are generated by the pricing problem are selected according to their reduced costs. Usually the pairing with the most negative reduced cost is sent to the restricted master problem. There are some alternative rules for selecting the pairings like the study of Bixby *et al.* [7]. They propose the *lambda pricing rule*, which is based on:

$$\lambda_p = c_p / \textstyle\sum_{i \in p} y_i,$$

where i denotes the flight leg, c_p is the cost of the pairing p and y_i is the dual value corresponding to the coverage constraint of flight i. Suppose that there are k columns that can be sent to the restricted master problem. λ_p values for all pairings that have negative reduced costs are calculated. Then, k columns with the k lowest λ_p value are sent to the restricted master problem instead of the columns with the most negative reduced costs. The number of iterations is reduced by applying this selection rule.

Makri and Klabjan [20] propose a number of pruning rules for the airline crew pairing problem. These pruning rules are applied in the pricing problem of the column generation method to fathom enumeration of columns. These rules can be classified into two groups as approximate and exact rules. Makri and Klabjan propose a new network, which is *mixed segment/duty timeline network*. In this network, the flights are represented by two nodes; one node is for the departure and the other one is for the arrival of the flight. There are also two types of arcs, duty and connection arcs. The departure node of the first flight in a duty period is connected to the arrival node of the last flight in that duty by a duty arc. The connection arcs, on the other hand, are used to connect two duty periods. If the arrival node of one duty is at the same airport as the departure node of another duty and the time between them is feasible with respect to the rest time, these duties are connected by a connection arc. The mixed segment/duty timeline network is an acyclic network which may have parallel duty arcs. Depth-first search is applied on this network to identify a negative reduced cost pairing and pruning rules are used to fathom unpromising partial pairings. Partial pairings contain duty periods within feasibility rules but do not have to end at the starting crew base. Makri and Klabjan use the score calculation proposed by Bixby *et al.* [7]. A score value of each partial pairing $s_p = c_p/\sum_{i \in P} y_i$ is calculated, where c_p is the cost of the partial pairing p and y_i is the dual value of flight i that is covered by partial pairing p. The score of a pairing is less than one, if and only if the pairing has a negative reduced cost. The approximate rules fathom partial pairings that will likely result in a pairing with a score, $s_p \geq 1$. The optimal solution may not be found just by using approximate rules because these rules might prune the pairings that will improve the objective function. Therefore, if a pairing with a negative reduced cost cannot be found by approximate rules, then they switch to the exact rules in the pricing problem. The exact rules only fathom those partial pairings, which are guaranteed to yield a nonnegative reduced cost.

2.5 Robust Airline Crew Pairing Problem

The airline crew pairing problem is usually solved under the assumption that there is no disruption at the operational level. Therefore, in most of the cases the optimal solutions found at the planning level cannot be performed exactly. In this section, we give some of the studies in which the robust models for the crew pairing problem are taken into consideration.

Schaefer *et al.* [23] propose two methods, which are used to estimate the cost of the crew schedules when disruptions occur at the operational level. Both methods are based on simulation. They assume that the disruptions are solved by delaying the regular flights. They also give a lower bound on the cost of the crew schedules by using the proposed method. Shebalov and Klabjan [24] provide a model to construct robust crew schedules at the planning phase. The problem introduced by them is a bicriterion optimization problem, in which both the total crew cost is minimized and the number of the *move-up crews* is maximized. Move-up crews are the crews that can be swapped in order to handle the disruption at the operational level. The proposed model is solved by Lagrangian relaxation and delayed column generation. Tekiner *et al.* [26, 27] also focus on the robust crew pairing problem. The type of the disruption taken into consideration by Tekiner *et al.* is adding the extra flights into the flight schedule during the operation. Two types of solutions are proposed to cover the extra flights in robust crew pairing problem, which are Type A and Type B solutions. In Type A solution, two crews are swapped to cover the extra flight. On the other hand, the extra flight is inserted into the sequence of flights of one pairing in Type B solution. Tekiner *et al.* solved only small-scale problems, where column generation is not applied. In this thesis we shall focus on the problem set forth by Tekiner *et al.* [26, 27] and analyze the pricing subproblem in case column generation is applied to solve medium-to-large problems.

CHAPTER 3

PRICING IN COLUMN GENERATION FOR A ROBUST AIRLINE CREW PAIRING PROBLEM

In our study, the subproblem of the column generation algorithm corresponds to the multi-label shortest path problem on a network. The network can be either the flight network or the duty-period network as mentioned in Vance *et al.* [28]. In this thesis, the flight network is used to denote the flights and the possible connections. A small flight network which contains flights on a given day is shown in Figure 3.1.

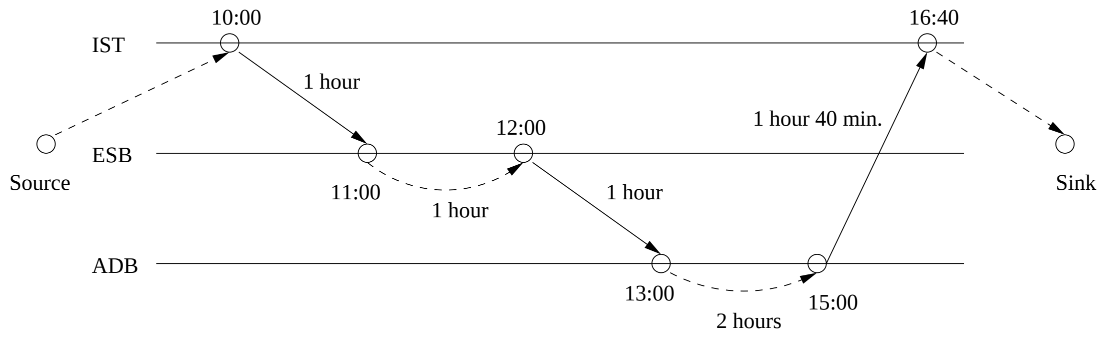

Figure 3.1: A flight network for the crew base IST.

Each flight is represented by an arc which connects two nodes, the node at the tail of the arc denotes the departure and the node at the head of the arc denotes the arrival of the flight. Such an arc is called a flight arc. The other type of arc in the flight network is the connection arc. Connection arcs are used to denote the possible connections between flights. In addition to departure and arrival nodes of the flights, the network has two more nodes, a dummy source and a dummy sink node. These nodes are used to mark the paths from the dummy source to the dummy sink. These paths correspond to the pairings in our problem. Recall that a feasible pairing starts and ends at the same crew base. If the departure airport of the flight is at the crew base, then its corresponding departure node is connected to the dummy source. There is a similar relationship between the arrival nodes and the dummy sink node. All incoming arcs to the sink emanate from the arrival nodes that are at the crew base.

Two flights can be connected if the arrival of the first flight and the departure of the second flight are at the same airport and the time between these flights is smaller than or equal to the maximum sit time. Also, this time should be greater than or equal to the minimum sit time. All flights that meet these conditions are connected in the flight network before the multi-label shortest path problem is solved. Moreover, the nodes in the flight network are topologically sorted before the multi-label shortest path problem begins. The node-treating algorithm explained in Section 3.1.2 is based on this topological order. The restrictions on the sit time are checked while constructing the flight network, and only those arcs corresponding to legal connections are inserted. As flights are treated in the multi-label shortest path problem, we shall also apply several other feasibility rules (see Section 3.1). These feasibility rules define both the feasible duty periods and the feasible pairings. The rest time between two duty periods changes dynamically depending on the elapsed time of the current duty period. Table 3.1 illustrates the rest time between duty periods that we use in our numerical examples.

Elapsed Time of Previous Duty Period (DP)	**Rest Time Between Two Duty Periods (DP)**
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
Long-range flights	Elapsed time of previous DP

Table 3.1: Rest time between two duty periods.

3.1 Multi-Label Shortest Path Algorithm

We use a column generation approach in this thesis because the number of the feasible pairings is very large. Instead of generating all feasible pairings, we start with a subset of them. The problem with the subset of all possible pairings is called the *restricted master problem*. At each iteration, the restricted master problem is optimized to get the dual variables corresponding to each flight. The dual of the LP relaxation of the primal problem (1.1) is as follows.

$$\begin{aligned} \max \quad & \textstyle\sum_{i\in\mathcal{F}} u_i & \\ \text{s.t.} \quad & \textstyle\sum_{i\in\mathcal{F}} a_{ij}u_i \le c_j, & \forall j \in \mathcal{P}, \\ & u_i \ge 0, & \forall i \in \mathcal{F}, \end{aligned} \tag{3.1}$$

where u_i is the dual variable corresponding to the i^{th} constraint (flight) in the problem (1.1). After optimizing the restricted master problem, we obtain u_i values. The pricing problem becomes a multi-label shortest path problem (MLSP) when the column generation method is applied to the crew pairing problem. The aim of the multi-label shortest path problem is to find a pairing with a negative reduced cost. The reduced cost of pairing j is calculated by

$$\bar{c_j} = c_j - \textstyle\sum_{i\in F} a_{ij}u_i. \tag{3.2}$$

If a pairing that improves the objective function is found, it is added to the restricted master problem and it is reoptimized. Then, the multi-label shortest path is resolved according to new dual values taken from the restricted master problem. If the minimum reduced cost at the end of the multi-label shortest path algorithm is nonnegative, then the optimal solution is found and the algorithm is terminated.

3.1.1 Applying Domination Rules and Determining Nondominated Paths

In the multi-label shortest path problem each path from the source node to any intermediate node (partial pairing) is denoted by a set of labels which give the state of the path. On each path the following labels are kept to calculate the cost of the pairing and to check the feasibility rules associated with the pairing:

- The total elapsed time (time-away-from-base, TAFB).
- The number of completed duty periods.
- Sum of the costs of the completed duty periods.
- Sum of the dual values of the flights covered by the path.

- Total number of flights covered by the path.

To calculate the cost of the duty period and to check the feasibility rules associated with the duty period, the following labels are kept through the duty period:

- Total elapsed time.
- Total flying time.
- Total number of flights covered.
- The cost of the current duty period.

At each node the predecessor node and the predecessor path of each path are also stored. In addition to these labels, the extra flights or the required deadheads that may be covered by the path are also kept by a label. The rules and the types of coverage of extra flights are explained in the Section 3.4. When a duty period is finished, all label values kept for the current duty period are reset for the next duty period. As it can be seen in Figure 3.2, there may be more than one path emanating from the source node to any intermediate node. In this network there are two connections from the source, which means that there are two paths at the beginning. Both paths go through the departure node of the right-most flight in the figure. Therefore, there are two paths, hence two set of labels, on the departure and the arrival nodes of this flight.

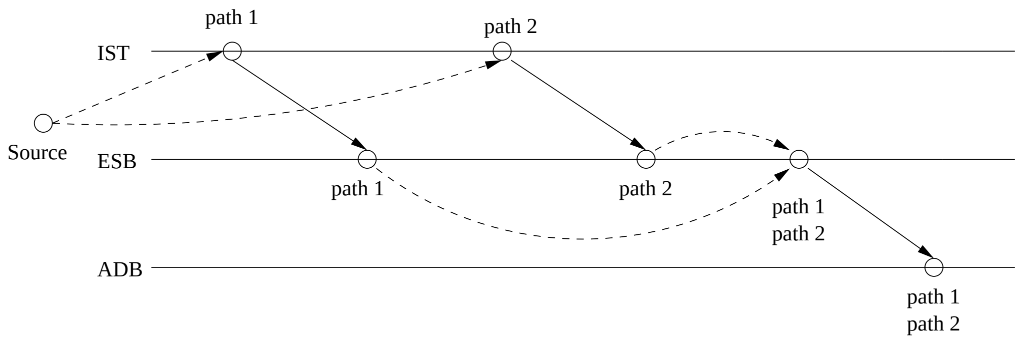

Figure 3.2: Multi-path on the flight network.

As a general representation, each path i from the source node to node j is denoted by a state (R_j^{iL}, C_j^i) where L is the set of labels kept by path i, R_j^{iL} corresponds to the set of values on each label $(R_j^{i1}, R_j^{i2}, ..., R_j^{i|L|})$ of path i and C_j^i is the cost of the path i. To optimize the objective function of problem (1.1), we want to keep the nondominated paths on each node.

Definition 3.1.1 Suppose that there are two paths emanating from the source node to node j. The corresponding states of these paths are (R_j^{1L}, C_j^1) and (R_j^{2L}, C_j^2), respectively. The first path dominates the second path if and only if $C_j^1 \leq C_j^2$ and $R_j^{1l} \leq R_j^{2l}$, $\forall l \in L$. In such a case, the first and second paths are called as nondominated and dominated paths, respectively.

An illustrative example is given in Figure 3.3. For simplicity, only two labels associated with the duty period (total elapsed time and total flying time) are considered. The value on each flight arc stores the flying time. The value on each

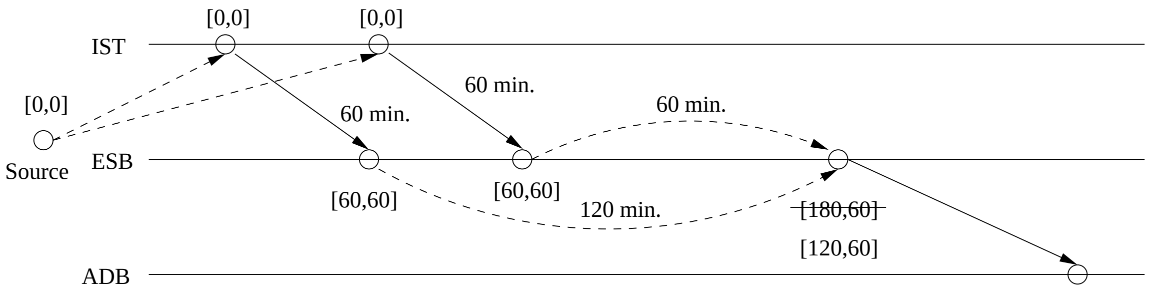

Figure 3.3: Domination of paths.

connection arc is the elapsed time for that connection. At the departure node of the third flight, there are two paths and because of that there are two states: [180,60] and [120,60]. Before carrying these paths to the arrival node of the third flight, domination rules are applied. Total elapsed time in the first path is greater than the total elapsed time in the second path. The flying time takes the same value on both paths. It can be said that first path is dominated by the second one. So, only second path will be carried through the third flight arc to the arrival node of that flight.

3.1.2 Node-Treating Algorithm

Recall that the nodes in the flight network are topologically sorted before the pricing problem is solved. This topological sorting ensures that each node in the network is treated exactly once. While a node is being treated, it is given that all its predecessors are already treated. The algorithm applied in this thesis is derived from the algorithms explained in Desrosiers *et al.* [15].

Let *S(j)* be the set of successor nodes of node *j*. Suppose that the number of nondominated paths on node j is m. The set of these paths is denoted by $P_j = \cup_{k=1}^{m}(R_j^{kL}, C_j^{k})$. In the node-treating algorithm each nondominated path on node j is carried to its each successor node by the arc that connects node j and its successor. There is also a list in this algorithm that keeps the nodes that will be treated. When a node is treated, its successors are added to that list according to their topological order. The node with the smallest order is treated first.

Algorithm 1 Node-Treating Algorithm in MLSP

```
1. Assigning labels at source node
    (a) $R^l_{source} = 0, \forall l \in L$
    (b) $C_{source} = 0$
    (c) Add successors of source node to the treating list, $T$
    (d) Sort $T$ according to increasing topological order
2. Node-Treating
    (a) Take the first node $j \in T$
    (b) Apply domination rules to paths on node $j$ and get $P_j$
    (c) For all $k \in S(j)$
        i. Carry each path of $P_j$ to node $k$ through the arc $(j, k)$
        ii. Add new paths to the set of paths on node $k$
        iii. Add node $k$ to the list, $T$ (if it is not in the list)
3. Removing nodes
    (a) Remove node $j$ from $T$
    (b) Sort $T$ according to increasing topological order
    (c) If list $T$ is empty, then stop; otherwise, go to second step
```

3.2 Buffer Column Pool

In column generation approach, there are some column selection criteria of that will be sent to the restricted master problem. Savelsbergh and Sol [22] propose different methods for the selection of the columns. One idea is to add any random column with negative reduced cost to the restricted master problem. Another idea is to send a number of columns that have negative reduced costs. The pros and cons of both criteria are explained in Savelsbergh and Sol [22]. They also suggest a column pool for column generation approach in the same study. After the pricing problem is solved, more than one column with negative reduced cost may be found. All these columns except the ones sent to the restricted master problem are kept in a column pool.

In this thesis, a buffer column pool is used like the column pool explained in Savelsbergh and Sol [22]. After the multi-label shortest path problem is solved, the paths on the sink node correspond to the feasible pairings that are generated by the node-treating algorithm described in Section 3.1.2. As it can be seen in Figure 3.4, the columns with negative reduced costs are stored in a list. One pairing, which has the minimum reduced cost, is sent to the restricted master problem from that list. Other ones are added to the buffer column pool.

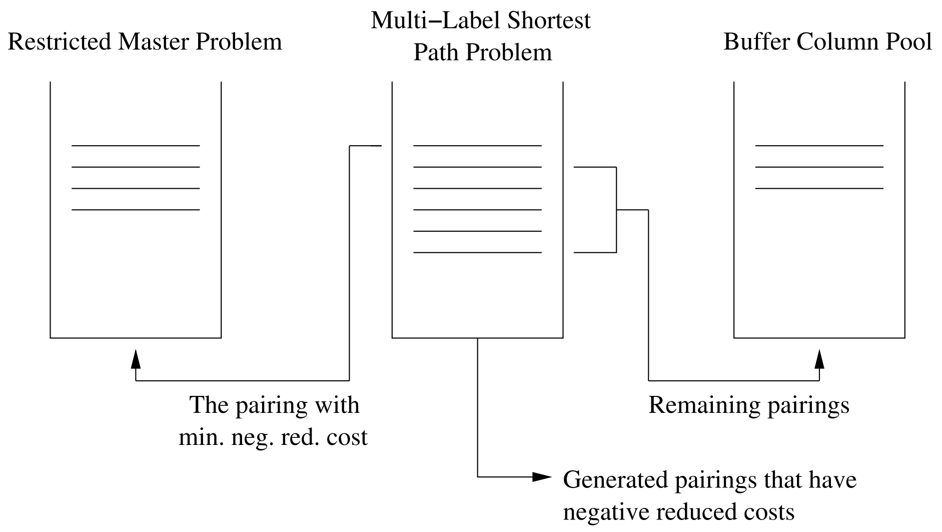
Restricted Master Problem
Multi–Label Shortest Path Problem
Buffer Column Pool
The pairing with min. neg. red. cost
Remaining pairings
Generated pairings that have negative reduced costs

Figure 3.4: The buffer column pool.

Initially the buffer column pool is checked after the restricted master problem is optimized. If a pairing with a negative reduced cost is found, it is sent directly to the restricted master problem and removed from the buffer pool. If such a pairing cannot be found, which means all pairings in the buffer column pool have nonnegative reduced costs, some of the pairings are removed from the buffer column pool. This is based on calculating the maximum reduced cost ($\bar{c}_{max}$) of the pairings in the buffer column pool. Then the pairings that have reduced costs greater than or equal to a fraction of $\bar{c}_{max}$ are discarded from the pool. This is followed by solving the multi-label shortest path problem to find a pairing that improves the objective function. The algorithm for the buffer column pool can be seen in the following part:

Algorithm 2 Buffer Column Pool Algorithm

```
1. Solve the restricted master problem for initial feasible solution
2. Solve pricing problem (MLSP)
3. Add columns with negative reduced costs into the list. If all columns have
   nonnegative reduced costs, then stop (optimal solution)
4. Send the column with the minimum reduced cost to the restricted master
   problem. Add all the remaining ones to the buffer column pool
5. Solve the restricted master problem
6. If buffer column pool contains a column with negative reduced cost, then re-
   move it from the buffer column pool and send to the restricted master problem.
   Then go to 5
7. If all columns in the buffer column pool have nonnegative reduced cost, remove
   the ones that have a reduced cost greater than or equal to a fraction of $\bar{c}_{max}$.
   Then, go to 2
```

From the computational point of view, the most costly part of the crew pairing problem is the pricing problem. On the other hand taking the columns from the buffer column pool saves a significant amount of time. The buffer column pool provides pairings with negative reduced costs without solving the multi-label shortest path problem. We only need to check the reduced costs of the pairings that are in the buffer column pool. When the restricted master problem is optimized, the dual values of the flight coverage constraints change. Therefore, at each iteration the sum of dual values of the flights and hence the reduced costs of the pairings in the buffer column pool may alter. Therefore, we need to update the reduced costs of all pairings in the buffer column pool at each iteration.

3.3 Pruning Methods

In this thesis, three rules are developed and applied to partial pairings. When domination rules are applied to the paths (partial pairings) on a processed node, each partial pairing is compared to every other partial pairings. This comparison part is very time consuming even if the number of paths is not large. Therefore, we use approximate and exact rules to fathom some partial pairings before domination rules are applied. In this way, the time spent on the application of the domination rules is reduced. These rules have some similarities with the rules mentioned in Makri and Klabjan [20]. One important difference is that we apply the pruning rules on the flight network. The rules defined by Makri and Klabjan are applied to the partial pairings on the *mixed segment/duty timeline network*. In this network, the partial pairings are composed of the completed duty periods. Moreover, they model the crew pairing problem as a set partitioning problem whereas we solve the LP relaxation of the set covering problem. The properties of the rules developed and applied in this thesis are explained in the following subsections.

3.3.1 Approximate Rule 1

In the multi-label shortest path problem, we keep several labels which means there may be several paths on each node of the network. An example is shown in Figure 3.5. In this figure there are four paths from the source node to the arrival node of the sixth flight (Node 12). A closer view to these paths is given in Figure 3.6.

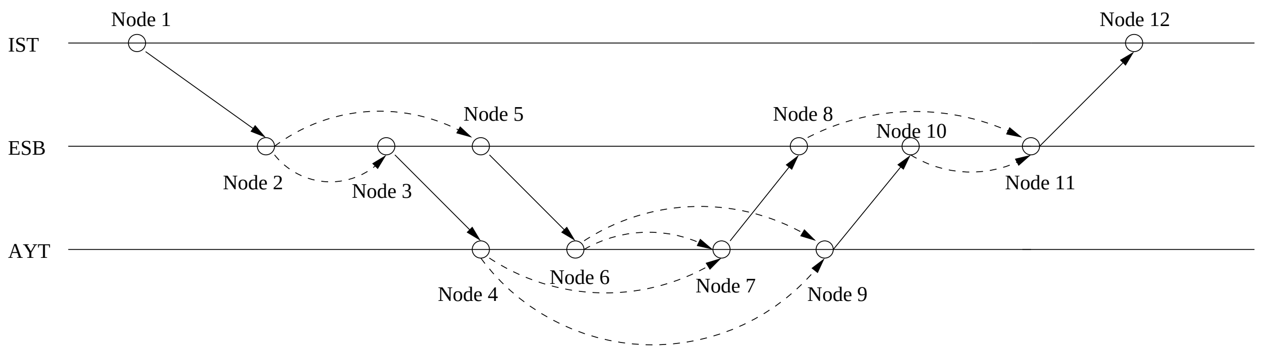

Figure 3.5: Flight network for crew base IST.

The first approximate rule calculates a *score value* of the paths on the processed node by using the values of the labels on the paths. We have the following values

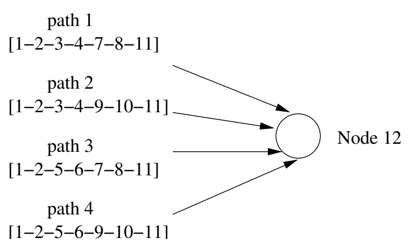

Figure 3.6: The paths on Node 12 in Figure 3.5

on each path:

- Sum of the costs of the completed duty periods of path j ($c_{j,total}$).
- Sum of the dual values of the flights covered by path j ($u_{j,total}$).
- The cost of the current duty period of path j ($c_{j,duty}$).

A score value of each path on the processed node i is calculated by

$$s_j = \frac{(c_{j,total}+c_{j,duty})}{u_{j,total}}, \quad j \in P_i, \tag{3.3}$$

where P_i is the set of all paths on node i. This score does not provide the definite information about the reduced cost of the completed pairing. The reduced cost of the completed pairing is estimated according to the current values of the partial pairing. If a path (partial pairing) of the processed node has a score value which is smaller than 1, then we can just say that the completed pairing of that path is likely to have a negative reduced cost. Similarly, if the score of the path is greater than or equal to 1, then the completed pairing is likely to have a nonnegative reduced cost. Nonetheless, it is still possible that this partial pairing may have a negative reduced cost when it is completed. Therefore, we keep partial paths with both negative and nonnegative reduced costs within the limits determined according to the total number of paths on the processed node i. The limit associated with the paths with $s_j < 1$ is n and the limit associated with the paths with $s_j \geq 1$ is m. The calculations of these numbers (n and m) are explained in Chapter 4.

After the proposed approximate rule is applied, the number of paths on the processed node is reduced. For example, there are four paths on Node 12 as shown in Figure 3.6. Suppose that we set n=2 and m=1. That is, after the approximate

Algorithm 3 Approximate Rule 1

```
1. Set $n$ and $m$ ($n \geq m$)
2. Set $n_0 = 0$ and $m_0 = 0$
3. For all $j \in P_i$
    (a) Calculate the score value, $s_j$
    (b) If $s_j < 1$ and $n_0 < n$ (or $s_j \geq 1$ and $m_0 < m$):
        i. Keep path $j$
        ii. $n_0$++ (or $m_0$++)
    (c) Otherwise:
        i. Fathom path $j$
```

rule is applied, there should be at most two paths with $s_p < 1$ and at most one path with $s_p \geq 1$. Suppose that the first three paths in Figure 3.6 have a score value less than 1. Then path 3 is erased by the approximate rule 1.

3.3.2 Approximate Rule 2

This second approximate rule is applied to the processed node after the first approximate rule fathoms some paths on the node. The average values of the pairings that are generated so far are used in this rule. Our aim is to anticipate the score of the completed pairing according to the values of current partial pairing and the values from already generated pairings. The average cost of all generated pairings over the flights covered by these pairings is given by

$$\bar{ac} = \frac{\sum_j c_j}{\sum_j \sum_i a_{ij}}, \quad j \in \bar{P}, i \in F, \tag{3.4}$$

where $\bar{P}$ is the set of pairings generated until this iteration. The average dual value of the flights covered by these pairings is obtained by

$$\bar{u} = \frac{\sum_j \sum_i a_{ij} u_i}{\sum_j \sum_i a_{ij}}, \quad j \in \bar{P}, i \in F. \tag{3.5}$$

To apply this rule we need to find the maximum number of flights (L) that can be covered by only one pairing that is generated so far. This implies that we search all generated pairings. To calculate the value of L, both the pairings in the pool of the restricted master problem and the pairings in the buffer column pool are considered. When approximate rule 2 is applied, we have the following values on each path:

- Sum of the costs of the completed duty periods of path j ($c_{j,total}$).
- Sum of the dual values of the flights covered by path j ($u_{j,total}$).
- The cost of the current duty period of path j ($c_{j,duty}$).
- The number of flights that are covered by path j (l_j).

Then, the score value of each path on processed node i is calculated by

$$\bar{s}_j = \frac{(c_{j,total}+c_{j,duty})+(L-l_j)\bar{a}c}{u_{j,total}+(L-l_j)\bar{u}}, \quad j \in P_i. \tag{3.6}$$

The score of each path on the processed node is calculated. And then some paths with a score $\bar{s}_j \geq 1$ are pruned (see Chapter 4). The score value calculated by the approximate rule 2 is more definite than the score value calculated by the approximate rule 1. We assume that the completed pairing of a partial pairing, which has a score value smaller than 1, will have a negative reduced cost. Therefore, all the paths with $\bar{s}_j < 1$ are kept. Consider again Node 12 given in Figure 3.6 after applying the approximate rule 1. Suppose that the score values of the remaining three paths are $s_1 = 0.8$, $s_2 = 1.1$ and $s_4 = 0.7$ (recall that path 3 is removed by approximate rule 1). The second approximate rule can further erase the second path.

3.3.3 Exact Rule

The multi-label shortest path problem searches a pairing with a negative reduced cost to improve the objective function. Approximate rules fathom several paths and provide fast column generation. However, pairings that have a negative reduced cost may also be pruned by these rules. Thus, we cannot say that the solution is optimal if only approximate rules are used. We next develop an exact rule to apply when the approximate rules cannot find a pairing with negative reduced cost.

When applying the exact rule, a score value of each path on the processed node i is calculated by

$$e_j = \frac{(c_{j,total}+c_{j,duty}+sp_i)}{(u_{j,total}+lp_i)}, \quad j \in P_i, \tag{3.7}$$

where sp_i and lp_i are the distances from node i to the sink node found by solving the shortest path and longest path problems on the flight network, respectively. The arc costs are flying times in the shortest path problem and dual values in the longest path problem. We know that the total flying time of a pairing is a lower bound on the cost of that pairing. We minimize the cost of the completed pairing while maximizing its sum of dual values. Therefore, if the score of the partial pairing is found as $e_p \geq 1$ by the exact rule, then the completed pairing of that path can never have a negative reduced cost (see Proposition 3.3.1 below). The number of paths that are pruned by the exact rule is not large, but the optimal solution is guaranteed.

To calculate the score of the partial pairing with the exact rule, a shortest path problem and a longest path problem are solved on the reverse flight graph; i.e., the distances are found from the sink node to the source node. In the shortest path problem each flight arc has its own flying time as its cost and each connection arc has zero cost. This problem is solved only once because the cost figures (flight time) are the same in all iterations. The longest path problem is solved according to the dual values. Each flight arc has its corresponding dual value as its cost and each connection arc has zero cost. The longest path problem is solved at every iteration that applies the exact rule, because the dual values may be different from one iteration to another. Under these circumstances, we have the longest distance in dual values and the shortest distance in flying times from any node to the sink node.

Proposition 3.3.1 *If the score value of partial pairing j on node i is calculated by the exact rule and*

$$e_j = \frac{(c_{j,total} + c_{j,duty} + sp_i)}{(u_{j,total} + lp_i)} \geq 1,$$

then any pairing that results from this partial pairing j will has a nonnegative reduced cost.

Proof: Suppose that the partial pairing j has m flights and it is completed to a pairing p by adding the flights $m+1, m+2, ..., k$. The flying times of these flights are denoted by $f_{m+1}, f_{m+2}, ..., f_k$. Then, using (2.1) and (2.2), we have the total flying time as a lower bound on the cost of the pairing. Therefore,

$$c_p \geq c_{j,total} + c_{j,duty} + \sum_{t=m+1}^{k} f_t \geq c_{j,total} + c_{j,duty} + sp_i,$$

since sp_i is the shortest path distance over flying times from node i to the sink node. We also have,

$$\sum_{t=1}^{k} u_t = u_{j,total} + \sum_{t=m+1}^{k} u_t \leq u_{j,total} + lp_i,$$

since lp_i is the longest path distance over dual values from node i to the sink node. Hence, we obtain a lower bound on the ratio of the cost of the pairing p to the sum of the dual values of the flights covered by that pairing:

$$\frac{c_p}{\sum_{t=1}^{k} u_t} \geq \frac{c_{j,total}+c_{j,duty}+\sum_{t=m+1}^{k} f_t}{\sum_{t=1}^{k} u_t} \geq \frac{c_{j,total}+c_{j,duty}+sp_i}{u_{j,total}+lp_i}.$$

Therefore, if a partial pairing j has a score value $e_j \geq 1$, then its completed pairing has a nonnegative reduced cost since $\frac{c_p}{\sum_{t=1}^{k} u_t} \geq 1$ implies $c_p - \sum_{t=1}^{k} u_t \geq 0$.
□

3.3.4 Exact and Hybrid Approaches

We generate two methods for the application of the exact rule in the multi-label shortest path problem. In the first approach, if the approximate rules do not find a pairing with a negative reduced cost, then we switch to the exact rule. The exact approach applies the exact rule to all nodes of the flight network throughout the pricing problem. The paths on each node are fathomed by using solely the exact rule in this approach.

The other approach, which we call as hybrid approach, is to apply both the approximate and the exact rules in the pricing problem. We explain this approach in Figure 3.7. The numbers on each node are given according to the topological order. The exact rule is applied to the node with the smallest order (Node 1) and then the approximate rules are applied to all other nodes. If a pairing with a negative reduced cost cannot be found at the sink, we go back to Node 2 to apply the exact rule. The paths on the remaining nodes are pruned by the approximate rules. Suppose that a pairing with negative reduced cost is found at the sink node. At this point the hybrid algorithm is stopped and the pairing with negative reduced cost is added to the restricted master problem. If a pairing with the negative reduced cost cannot be found, the hybrid approach continues that is the exact rule is applied to the next node (node 3) and the approximate rules are applied to the subsequent nodes.

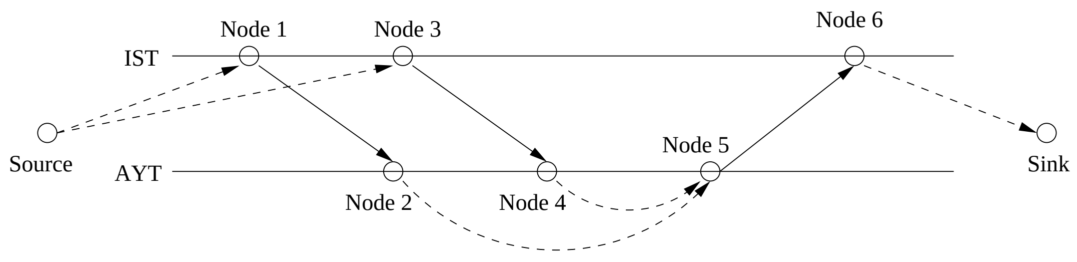

Figure 3.7: Hybrid approach on the flight network.

3.4 Managing Extra Flights

The crew pairing problem is solved after the timetable of the flights is constructed and the assignment of the fleets is completed. Therefore, it is assumed that the flight scheduling and fleet assignment problems are solved before we start to solve the crew pairing problem. After the solution of the crew pairing problem is found, it is sent to the crew assignment problem to complete the planning process in the airlines.

The airline companies need to add extra flights into the flight schedule at the operational level [26, 27]. This may occur because of seasonal changes or specific customer requests. In this thesis, we analyze the robust crew pairing problem in which the extra flights are taken into consideration at the planning phase. The exact information of the extra flights are not known during the crew scheduling process but they can be predicted with the help of the extra flights flown in previous seasons. These possible extra flights may or may not be inserted into the flight schedule during the operation. Therefore, the pairings that can cover any extra flights should also be feasible in case the extra flights are not flown.

Each extra flight brings about at least one deadhead to reposition the crew at the required station. The deadhead associated with the extra flight is different from the regular deadhead definition. A flight without any passengers can also be called as deadhead in the robust crew pairing problem. Tekiner *et al.* [26, 27] suggest nine possible conditions for covering the extra flights and their required deadhead flights. These conditions are developed for the case that all feasible pairings are generated. We apply the column generation method instead of generating all feasible pairings. Therefore, two Type B and one Type A solutions which are composed of one extra flight and one deadhead flight are considered primarily in this thesis (see Sections 3.4.1 and 3.4.2). The proposed solution method applied in this study can be extended for other Type A and B solutions. As previously mentioned, airline companies can predict the extra flights. This is based on anticipating the departure time interval of the extra flights as shown in Figure 3.8. In addition to the time interval, the departure and arrival stations and the duration of the extra flights are also known.

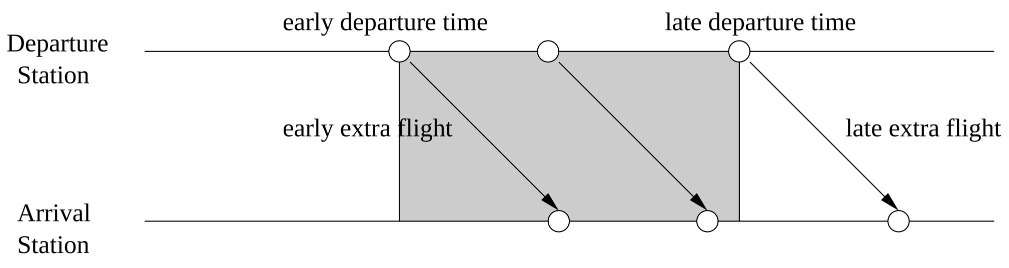

Figure 3.8: Time window for the departure time of the extra flight.

In [10] two approaches are proposed to solve the robust crew pairing problem. The first one is called the dynamic approach. This approach requires pricing and hence it is directly related to the multi-label shortest path problem. The multi-label shortest path problem is altered to find the Type B solutions and the pairings that may potentially form Type A solutions. In the second approach, which is referred to as static approach, all pairings that form feasible Type A solutions are generated before the column generation method is started. We refer the reader to [10] for the details of the static approach of the robust crew pairing problem. In the following subsections, we describe the changes required in the pricing problem for applying the dynamic approach to the robust crew pairing problem.

3.4.1 Type A Solution

Tekiner *et al.* [26, 27] propose six Type A solutions. In [26] and [27] all possible pairings are generated. However, we apply column generation to the crew pairing problem. Therefore, we do not have all feasible pairings and we consider the Type A solution which consists of one extra flight and one deadhead flight (Type A.1).

Figure 3.9 shows that two pairings can construct the first Type A solution only if they are still feasible after swapping to cover the extra flight and its associated deadhead. In this figure the first pairing (p_1) covers flights 1 and 2, whereas the second pairing (p_2) covers flights 3 and 4. After swapping, the first pairing covers flight 1, the extra flight and flight 4. The sequence composed of flight 3, deadhead associated with the extra flight and flight 2 is covered by the second pairing.

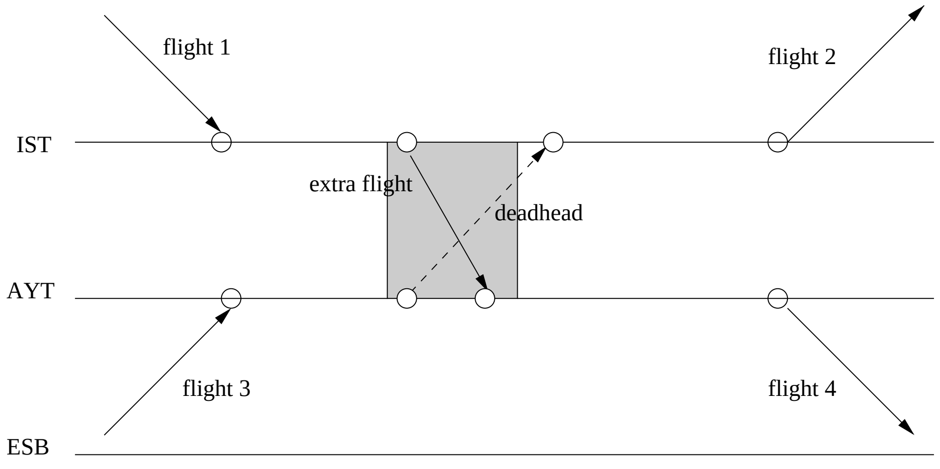

Figure 3.9: Type A solution.

The solution above is most suitable when all feasible pairings are generated offline before solving the crew pairing problem. However, recall that when we use column generation method in the crew pairing problem, we do not have all feasible pairings. While a pairing is being generated in one iteration, the pairing that may take part in a Type A solution with that pairing, possibly has not been generated yet. Hence, the extra flight and its associated deadhead flight are considered separately to specify the candidate partial pairings that may provide a Type A solution. Figure 3.10 and Figure 3.11 demonstrate the required conditions for an extra flight and its associated deadhead, respectively.

A partial pairing (path from source node to any node) is a candidate pairing for Type A solution, if it meets the following feasibility rules. We illustrate these rules in Figure 3.10:

- The time between the arrival node of flight 1 (Node 1) and the departure time of flight 2 (Node 4) should include the time window defined for the departure time of the extra flight.

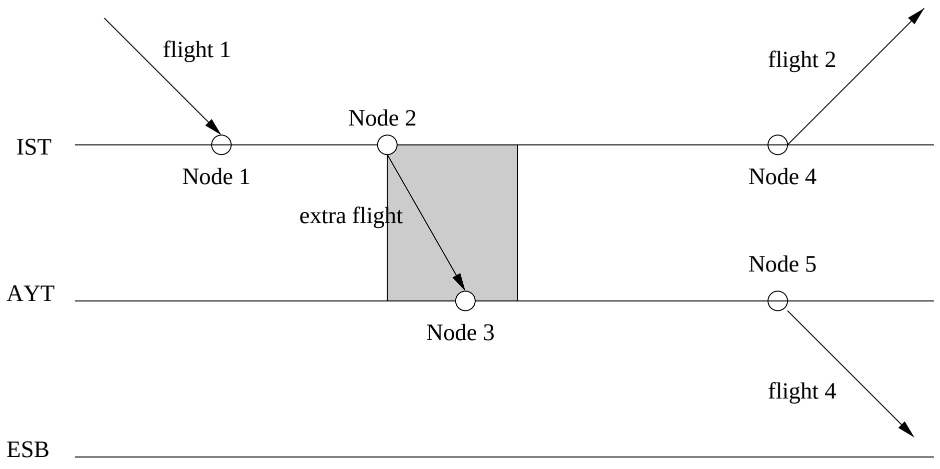

Figure 3.10: Extra flight condition for Type A.1 solution.

- The time between Node 1 and the early departure node of the extra flight (Node 2) should be greater than or equal to the minimum sit time defined for the regular flights. When the minimum sit time is checked, the early departure time is considered.

- The time between the arrival node of late extra flight and Node 5 should be considered. This check cannot be controlled during the pricing problem, because while Node 1 is being processed there is no connection between Node 1 and Node 5. Therefore, this check should be completed during column management (see [10] for details).

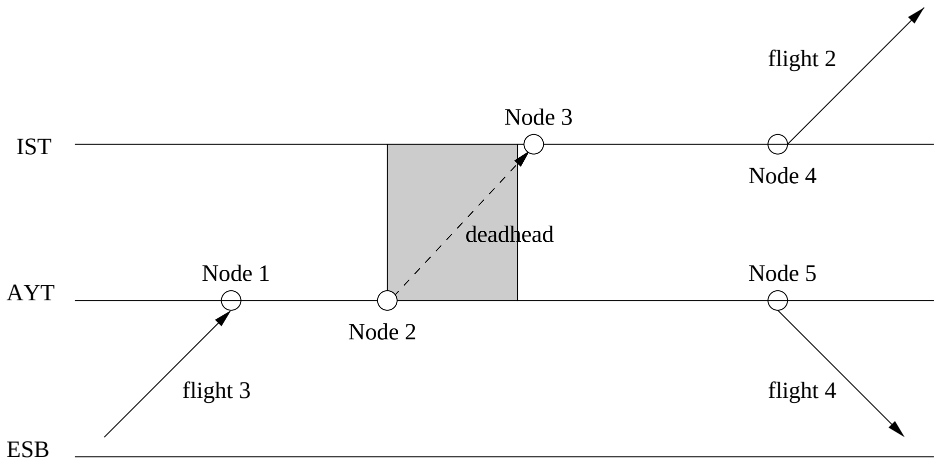

Figure 3.11: Deadhead condition for Type A.1 solution.

The feasibility checks for the associated deadhead flight are slightly different from the checks for the extra flight. Using now Figure 3.11, we list these rules:

- The time between the arrival node of flight 3 (Node 1) and the departure time of flight 4 (Node 5) should include the time window defined for the departure time of the extra flight.
- The time between Node 1 and the early departure node of the deadhead flight (Node 2) should be greater than or equal to the minimum sit time defined for the deadhead flights.
- The time between the arrival of deadhead flight and Node 4 should be considered. This check cannot be controlled during the pricing problem, because while Node 1 is being processed there is no connection between Node 1 and Node 4. Therefore, this check should be completed during column management (see [10] for details).

In the node-treating algorithm, all connections emanating from the processed node are examined for the extra flight. Suppose that the processed node is Node 1 in Figure 3.11. If the connection between Node 1 and Node 4 satisfies the feasibility rules defined for the deadhead flight, then the partial path from source node to Node 4 is tagged as a candidate pairing for Type A solution by using an additional label. The candidate pairings are then divided into two categories, candidates that can cover the extra flight, and the candidates that can cover the associated deadhead flight.

3.4.2 Type B Solution

The mathematical model of the robust crew pairing problem is different from the model of the classical crew pairing problem. The robust crew pairing model associated with both Type A and Type B solutions is given by

$$\begin{array}{lll}
\min & \sum_{j\in\mathcal{P}} c_j x_j + \sum_{k\in\mathcal{K}} d_k z_k + \sum_{k\in\mathcal{K}} d_k (\sum_{j\in\mathcal{P}} -x_j \bar{a}_{kj} + \sum_{j,m\in\mathcal{P}} -y^k_{(j,m)} \bar{a}_{jmk}) & \\
\text{s.t.} & \sum_{j\in\mathcal{P}} a_{ij} x_j \geq 1, & \forall i \in \mathcal{F}, \\
& \sum_{j\in\mathcal{P}} \bar{a}_{kj} x_j + \sum_{j,m\in\mathcal{P}} \bar{a}_{jmk} y^k_{(j,m)} + z_k \geq 1, & \forall k \in \mathcal{K}, \\
& 2\bar{a}_{jmk} y^k_{(j,m)} \leq x_j + x_m, & \forall (j,m) \in \mathcal{P}, \forall k \in \mathcal{K}, \\
& x_j \in \{0,1\}, & \forall j \in \mathcal{P}, \\
& z_k \in \{0,1\}, & \forall k \in \mathcal{K},
\end{array} \tag{3.8}$$

where K is the set of all possible extra flights, d_k represents the cost that is incurred when extra flight k is not covered by any selected pairing; $\bar{a}_{kj}$ is equal to 1, if extra flight k is covered by pairing j in Type B solution and it is equal to 0, otherwise. The variable z_k becomes 1, if extra flight k is not covered by any selected pairing and becomes 0, otherwise. $\bar{a}_{jmk}$ is equal to 1, if extra flight k is covered by (j,m) pairing tuple in Type A solution and becomes 0, otherwise.

The robust crew pairing model associated with the Type B solutions is given by

$$\begin{array}{lll}
\min & \sum_{j\in\mathcal{P}} c_j x_j + \sum_{k\in\mathcal{K}} d_k z_k + \sum_{k\in\mathcal{K}} d_k (\sum_{j\in\mathcal{P}} -x_j \bar{a}_{kj}) & \\
\text{s.t.} & \sum_{j\in\mathcal{P}} a_{ij} x_j \geq 1, & \forall i \in \mathcal{F}, \\
& \sum_{j\in\mathcal{P}} \bar{a}_{kj} x_j + z_k \geq 1, & \forall k \in \mathcal{K}, \\
& x_j \in \{0,1\}, & \forall j \in \mathcal{P}, \\
& z_k \in \{0,1\}, & \forall k \in \mathcal{K}.
\end{array} \tag{3.9}$$

The third component, $\sum_{k\in\mathcal{K}} d_k(\sum_{j\in\mathcal{P}} -x_j \bar{a}_{kj})$, is added to the objective function, because if an extra flight can be covered by any pairing in Type B solution, then we encourage this pairing to enter the solution. While Type B solutions are being searched, the dual values of each extra flight can be taken from the second constraint. The reduced cost of pairing j is calculated by

$$\bar{c_j} = (c_j - \sum_{k\in\mathcal{K}} d_k \bar{a}_{kj}) - \sum_{i\in F} a_{ij} u_i - \sum_{k\in K} \bar{a}_{kj} v_k, \tag{3.10}$$

where v_k is the dual variable corresponding to the coverage constraint of k^{th} extra flight. Moreover, the feasibility rules for both the departure node of the extra flight and the arrival node of the deadhead can be checked because we try to insert them into one pairing. Therefore, the pairings, which are Type B solutions, are provided at the end of the multi-label shortest path problem. This is different from the pairings that can form Type A pairings. Type A solutions cannot be provided by the pricing problem, we only label the candidate ones (see [10] for managing Type A solutions during column management).

In this thesis, two Type B solutions can be provided. The first solution involves the deadhead flight after the extra flight as shown in Figure 3.12. The feasibility rules for this solution are explained through the flights given in Figure 3.12.

- The connection between Node 2 and Node 3 should be feasible and the time window defined for the departure of the extra flight should fit into this connection time.
- Time between Node 2 and the early departure node of the extra flight (e_1) should be greater than or equal to the minimum sit or rest time. The rest time is checked if flight 1 is the last flight in a duty period.
- Time between the arrival node of deadhead (d_2) and Node 3 should be greater than or equal to the minimum sit time.

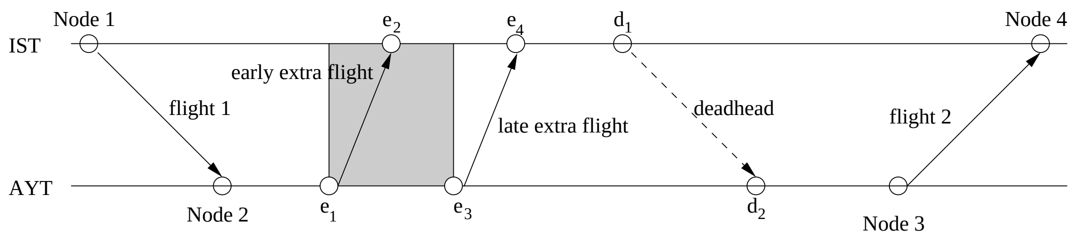

Figure 3.12: First Type B solution.

As it can be seen, there is no time window for the deadhead of the extra flight. However, there is a feasibility rule to check the deadhead flight. The time between the arrival node of the late extra flight (e_4) and the departure node of the regular flight (Node 3) should be at least as long as the sum of the following components:

- The duration of the deadhead flight.

- The minimum sit time between the arrival of the late extra flight and the departure of the deadhead flight.

- The minimum sit time between the arrival of the deadhead flight and the departure of the regular flight.

The second Type B solution involves the deadhead flight before the extra flight. The feasibility rules checked for this solution are explained by using Figure 3.13.

- The connection between Node 2 and Node 3 should be feasible and the time window defined for the departure of the extra flight should be involved by this connection time.

- Time between the arrival node of the late extra flight (e_4) and Node 3 should be greater than or equal to the minimum sit time.

- Time between Node 2 and the departure node of the deadhead flight (d_1) should be greater than or equal to the minimum sit (or rest) time. The rest time is controlled if flight 1 is the last flight in a duty period.

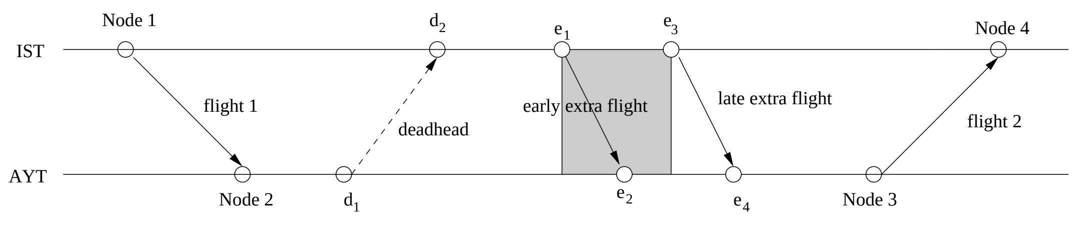

Figure 3.13: Second Type B solution.

In this solution again there is no time window for the deadhead flight. However, there is a feasibility rule to check the deadhead flight. The time between the arrival node of the regular flight (Node 2) and the departure node of the early extra flight (e_1) should be at least as long as the sum of the following components:

- The duration of the deadhead flight.

- The minimum sit time between the arrival of the regular flight and the departure of the deadhead flight.

- The minimum sit time between the arrival of the deadhead flight and the departure of the early extra flight.

In the node-treating algorithm the above feasibility rules are checked for all connections of the processed node. The extra flight and its deadhead flight are inserted into the path (partial pairing), if any connection between two regular flights of that path is feasible for them. This insertion is represented by another path which means one path is branched into two paths if it can cover any extra flight as a Type B solution. This is illustrated in Figure 3.14.

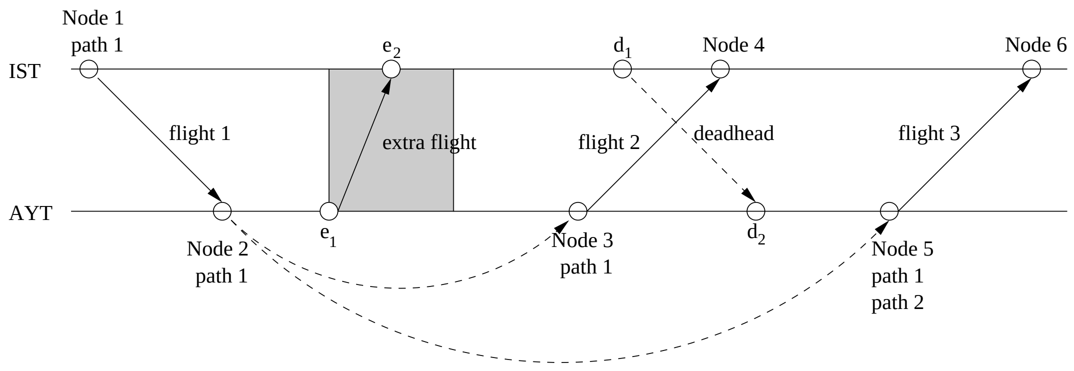

Figure 3.14: Node-treating algorithm for Type B solution.

There are two connections emanating from Node 2 to Node 3 and Node 5. Moreover, there is one path from source node to Node 2. The connection between Node 2 and Node 3 is feasible. Therefore, the path on Node 2 is carried to Node 3 through the connection that connects them. This connection cannot cover the extra flight as a Type B solution because the deadhead of the extra flight cannot be inserted within the connection between Node 2 and Node 3. However, the connection between Node 2 and Node 5 can include the extra flight and its deadhead. This connection is also feasible. The path on Node 2 is carried through the connection arc to Node 5 and it is branched into two paths. One path (path 1) is the actual connection in crew pairing problem and the second one (path 2) is for the case that the extra flight is flown. The differences between these two paths are as follows:

- The flying time of path 2 involves the extra flight whereas path 1 does not.
- The dual value of the extra flight is added to the sum of the dual values on path 2.

The path that can cover the extra flight is considered as a different path from path 1. Even if the extra flight is not inserted to the flight schedule during the operation, path 2 remains feasible.

As mentioned before, we use two approaches to cover the extra flights in the crew pairing problem. The dynamic approach is related to the pricing problem of the column generation technique. In this method the multi-label shortest path problem is altered to find the Type B solutions and the pairings that are labeled as the candidate pairings for Type A solutions. In addition to the buffer column pool, the dynamic approach involves another pool, which is called as *fixed pool*, to keep the candidate pairings to form Type A solutions. As we explained before, after the pricing problem is solved, one pairing with the most negative reduced cost is sent to the restricted master problem. Now, in addition to that column, we send all candidate pairings that can form Type A solutions regardless of the negativity of their reduced costs, to the fixed pool. The remaining pairings with negative reduced costs are kept in the buffer column pool. The optimal solution is found for problem (3.9) and then feasibility checks for the swapping the candidate Type A pairings are completed.

3.5 Finding A Lower Bound

The optimal solution of the restricted master problem is an upper bound for the problem (1.1) with all possible pairings. At each iteration that we apply the exact rule, we can also calculate a lower bound for the master problem. By this way, we can stop the algorithm if the upper bound is close to the lower bound.

To find a lower bound, we consider the nodes that are not processed yet. A γ_j^i value is calculated for each path j on the unprocessed node i:

$$\gamma_j^i = (c_{j,total} + c_{j,duty} + sp_i)/(u_{j,total} + lp_i), \quad i \in N, \quad j \in P_i, \tag{3.11}$$

where N is the set of unprocessed nodes whose predecessors have been processed and $(u_{j,total} + lp_i) > 0$. Then, we evaluate

$$\bar{\gamma} = \min(\gamma_j^i), \quad \forall i \in N, \quad \forall j \in P_i. \tag{3.12}$$

When node i is treated, γ_j^i values of its partial pairings are removed from the calculation of $\bar{\gamma}$. Then, the lower bound is found by multiplying $\bar{\gamma}$ with the optimal solution of the restricted master problem.

Proposition 3.5.1 *Let $\bar{x}$ be the optimal solution of a restricted master problem of* (1.1) *and $\bar{u}$ be the corresponding dual optimal solution. Then,*

$$\bar{\gamma}\textstyle\sum_{p\in P} c_p\bar{x}_p = \bar{\gamma}\sum_{t\in F} u_t$$

is a lower bound on the optimal objective function of the LP relaxation of the problem (1.1).

Proof: Suppose that the partial pairing j is completed into a pairing p which has m flights. We know from the proof of Proposition 3.3.1 that

$$\tfrac{c_p}{\sum_{t=1}^{m} u_t} \geq \tfrac{c_{j,total}+c_{j,duty}+sp_i}{u_{j,total}+lp_i},\ \forall j \in P_i.$$

Therefore,

$$\underbrace{\min_{p\in P}\left\{\frac{c_p}{\sum_{t=1}^{m} u_t}\right\}}_{\gamma} \geq \underbrace{\min_{i\in N, j\in P_i}\left\{\frac{c_{j,total}+c_{j,duty}+sp_i}{u_{j,total}+lp_i}\right\}}_{\bar{\gamma}}.$$

If $\bar{\gamma}\bar{u}$ is a feasible solution to problem (3.1), then $\bar{\gamma}\sum_{t\in F}\bar{u}_t$ is a lower bound on the objective function of problem (1.1). Since $\bar{\gamma} \leq \gamma$, then we can show the feasibility of $\bar{\gamma}\bar{u}$ by using $\gamma\bar{u}$. First of all, the constraint $\gamma\sum_{t\in p}\bar{u}_t \leq c_p$, $\forall p \in P$ should be satisfied. If $\sum_{t\in p}\bar{u}_t = 0$, then $0 \leq c_p$ which is given. If $\sum_{t\in p}\bar{u}_t > 0$, then $\gamma \leq \frac{c_p}{\sum_{t\in p}\bar{u}_t}$ which is met by the definition of γ. Since $\bar{u} \geq 0$ and $\gamma > 0$, then $\bar{u}\gamma \geq 0$. This satisfies the second constraint.

□

3.6 Flow Chart

The following figure gives the flow chart of all algorithms described in this chapter for the crew pairing problem. We start to solve this problem by finding an initial feasible solution. The pairings of the initial feasible solution are sent to the restricted master problem (RMP). At the first step, after the RMP is optimized, the multi-label shortest path problem is solved. The pairing that has the minimum negative reduced cost is sent to the RMP and remaining ones are sent to the buffer column pool. The RMP is again optimized. However, this time the buffer column pool is checked to find a pairing with a negative reduced cost before the multi-label shortest path problem is solved. If such a pairing is found, it is sent to the RMP. If all pairings in the buffer column pool have nonnegative reduced costs, then some of them are removed and the multi-label shortest path problem is solved. The optimal solution of problem (1.1) is found, if there is not a pairing with negative reduced cost at the end of the multi-label shortest path problem.

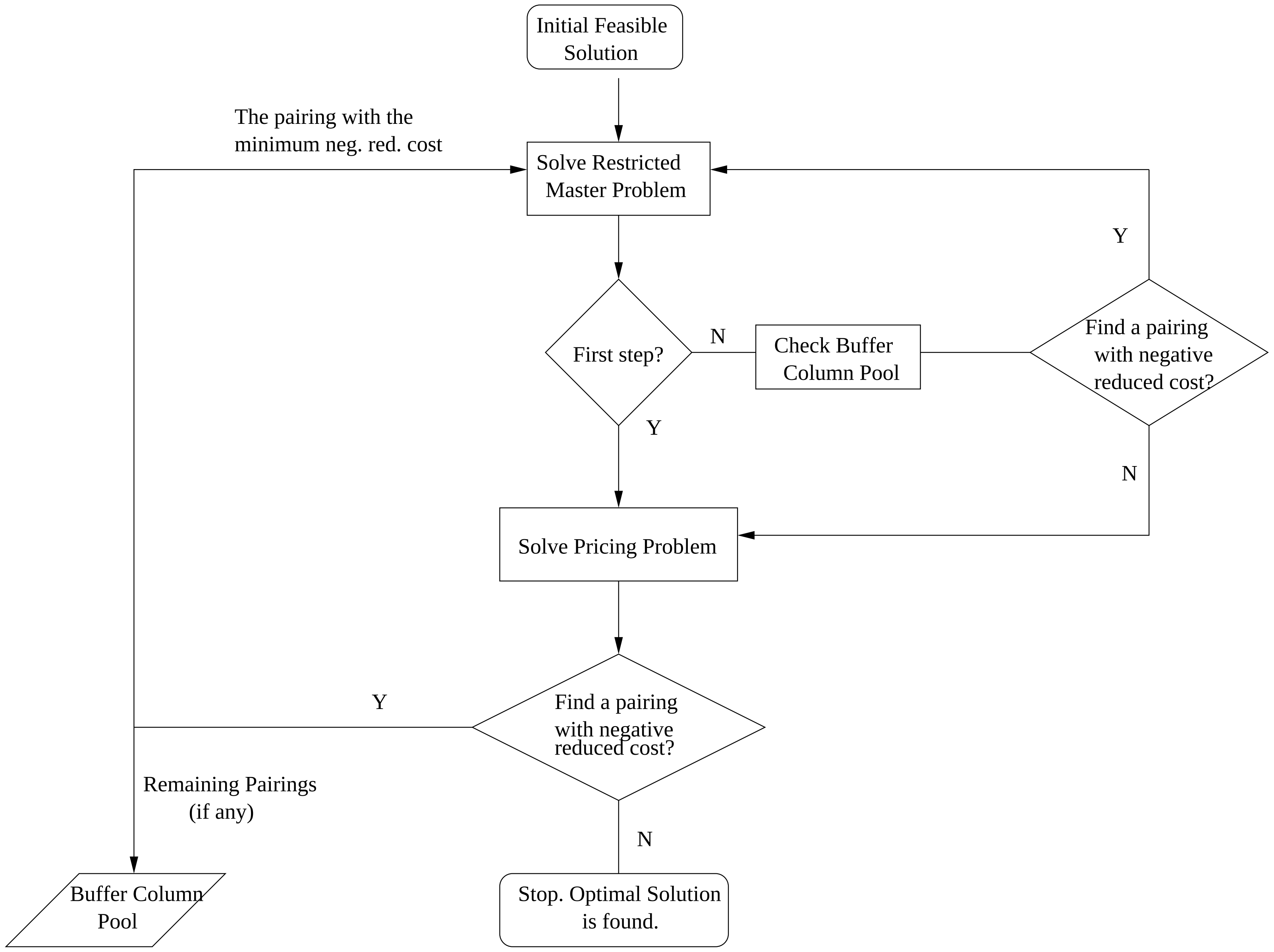
Initial Feasible Solution
The pairing with the minimum neg. red. cost
Solve Restricted Master Problem
Y
First step?
N
Check Buffer Column Pool
Find a pairing with negative reduced cost?
Y
N
Solve Pricing Problem
Y
Find a pairing with negative reduced cost?
Remaining Pairings (if any)
N
Buffer Column Pool
Stop. Optimal Solution is found.

Figure 3.15: Flow chart of the algorithms described in Chapter 3.

CHAPTER 4

COMPUTATIONAL RESULTS

In this thesis, we solve three problem instances (see Appendices A, B and C). Two problems are daily and one problem is weekly. The number of all feasible pairings is very large especially for the weekly problem. Hence, we apply column generation to these three problems instead of generating all possible pairings. We wrote a code in Visual C++ to get the solution of the LP relaxation of the problem (1.1). The pricing problem of the column generation algorithm is solved on the flight network. For these three problems, the feasible pairings are generated for one crew base. The restricted master problem of the LP relaxation of problem (1.1) is optimized by ILOG CPLEX 11.0 [29].

Possible extra flights are taken as input in addition to the regular flight schedule. However, extra flights differ from regular flights. They may or may not be added to the flight schedule at the operational level. For each daily problem, we consider one possible extra flight. We assume that two possible extra flights are taken into consideration for the weekly problem. We know that each extra flight requires at least one deadhead flight to reposition the crew members. Type B solution is the pairing that can cover the extra flight and its required deadhead flight between its two regular flights. For these three problems we can find Type B solutions in multi-label shortest path problem.

We apply three pruning rules in the multi-label shortest path problem. Approximate rule 1 keeps both the partial pairings (paths) that have score values smaller than 1 and the partial pairings that have score values greater than or equal to 1 within the limits. These limits are calculated according to the number of paths on the processed node. First of all, we want to keep more paths whose completed pairings are likely to have negative reduced costs. Therefore, the limit (n) determined for the paths with $s_j < 1$ is greater than the limit (m) determined for the paths with $s_j \geq 1$. The limits n and m are calculated by multiplying the number of paths with certain parameters. The parameter used for computing the limit n is called $fneg$. This parameter takes values 0.2, 0.4 and 0.8. The calculation of the limit m is based on another parameter that is called $fpos$. It takes values 0.1 and 0.2. If the number of the paths on the processed node is large, the limits n and m are also large. Then, the time required by the approximate rule 1 to prune the paths is increased. Therefore, we need another parameter that is called $fraction$ which depends on the number of paths on the processed node. The calculation of the parameter $fraction$ is given by

$$fraction = exp^{(-\alpha)*nop}, \tag{4.1}$$

where nop is the number of paths on the processed node. We determine α values in such a way that the value of the $fraction$ does not decrease suddenly as the number of paths increases. Therefore, we choose the values 10^{-3} and 8×10^{-4} values for α. Figure 4.1 and Figure 4.2 illustrate the $fraction$ for these two α values on the following page. The calculations of the limits n and m are given by

$$n = fneg * nop * fraction, \tag{4.2}$$

$$m = fpos * nop * fraction. \tag{4.3}$$

where $fneg$ and $fpos$ are determined percentages, nop is the number of the paths and $fraction$ is the parameter that depends on nop.

Approximate rule 2 keeps all paths that have score values smaller than 1. If there are several paths that have score values greater than or equal to 1, it keeps some of them; otherwise, it prunes all of them. We know that the completed pairing may have a negative reduced cost even if its partial pairing has a score value $\bar{s}_j \geq 1$ that is calculated by approximate rule 2. Moreover, we do not want to prune too many partial pairings. Therefore, we keep some paths that have score values greater than or equal 1. To do this, we need a parameter which is called $flim$. There are two situations for pruning paths by applying approximate rule 2:

1. If the number of paths, which have score values $\bar{s}_j \geq 1$, is smaller than or

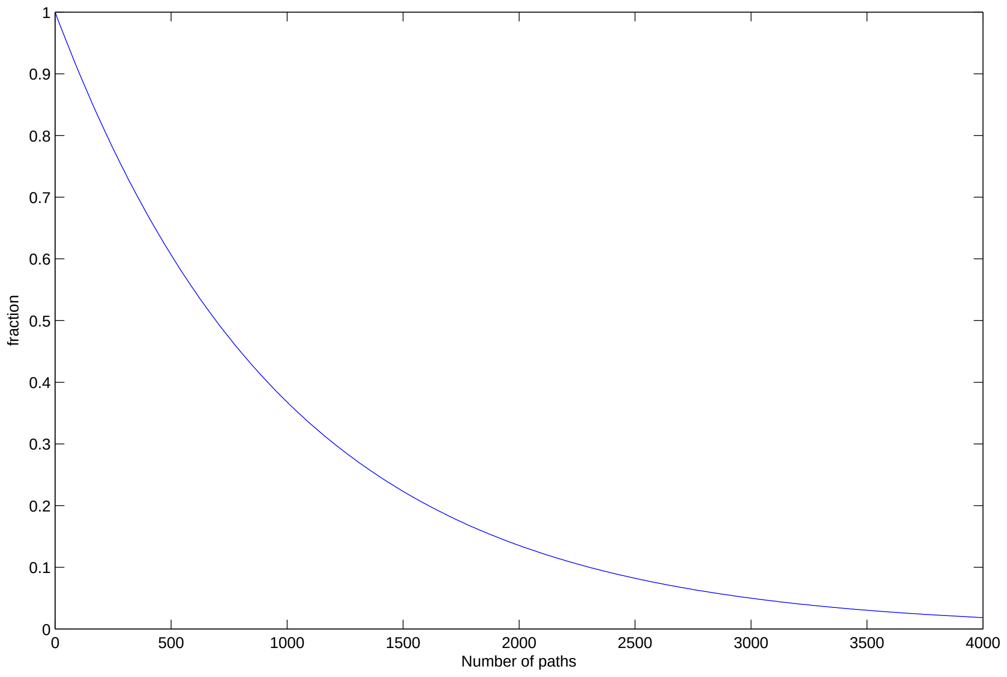

Figure 4.1: $fraction$ versus *number of paths* for $\alpha = 10^{-3}$.

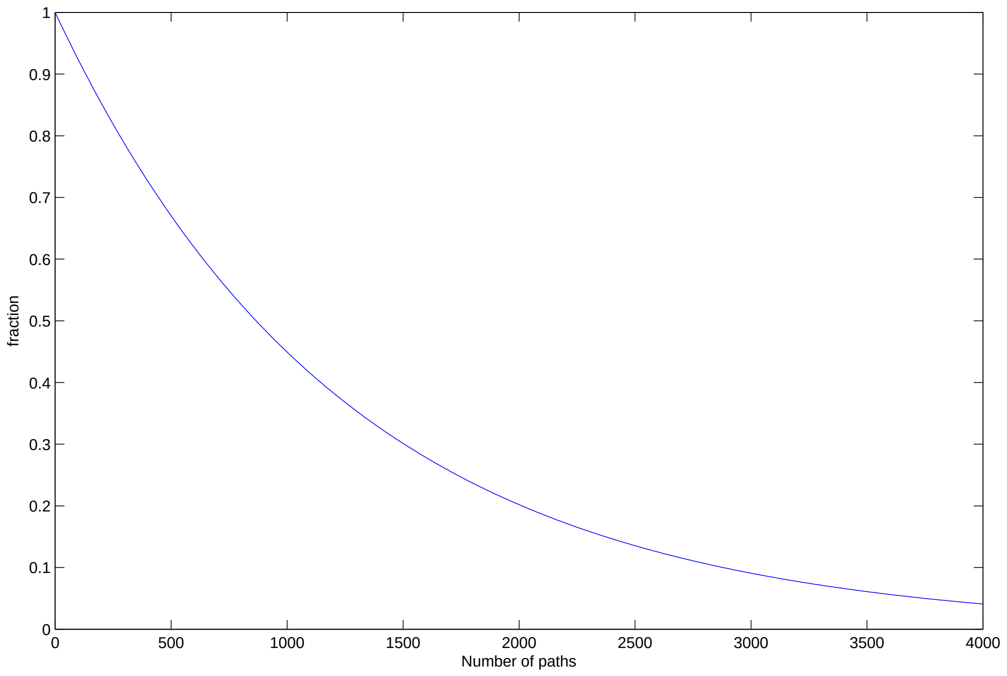

Figure 4.2: $fraction$ versus *number of paths* for $\alpha = 8 \times 10^{-4}$.

equal to ($flim$*nop), then all of these paths are fathomed.

2. If the number of paths, which have score values $\bar{s}_j \geq 1$, is greater than ($flim$*nop), then the number of paths that will be pruned is equal to the value of ($flim$*nop).

Another parameter (N) is needed when the domination rules are applied to the partial pairings of the processed node. Recall that, in the multi-label shortest path problem, the candidate partial pairings that may construct Type A solutions are labeled (see Section 3.4.1). The objective of the robust crew pairing problem (3.9) is to maximize the number of pairings that can cover extra flights and their required deadhead flights. However, we recognize that there are several partial pairings labeled as candidate Type A pairings. If we prevent all of these candidate Type A partial pairings from being pruned during domination, we cannot find the solution in a reasonable computation time. On the other hand, domination rules may prune too many candidate type A pairings. Therefore, we use the parameter N that specifies how many additional paths will be kept as candidate Type A pairings on the node currently processed.

The values of the parameters that are used by three test instances are given by

Parameter	**Values**	**Problem**
N	0, 25, 750, ∞	1
N	0, 50, 1000,∞	2
N	0, 25, 50, 75	3
$fneg$	0.2, 0.4, 0.8	1, 2 and 3
$fpos$	0.1, 0.2	1, 2 and 3
α	10^{-3}, 8×10^{-4}	1, 2 and 3
$flim$	0.3, 0.6	1, 2 and 3

problem 1 :	Problem with 42 flights
problem 2 :	Problem with 96 flights
problem 3 :	Problem with 135 flights

Table 4.1: Values of the parameters.

We solved three problems with 42, 96 and 135 flights which are referred to as problem 1, 2 and 3, respectively. We have 96 sets of parameters for each problem and a set of parameters corresponds to a experiment in our study. Each experiment is applied with the exact approach and it is repeated with the hybrid approach. The experiments and the associated average CPU times for an iteration of the multi-label shortest path problem with the exact and hybrid approaches are given in Appendices A, B and C for problems 1, 2 and 3, respectively. The changes in parameters do not affect the computational time in problem 1 and problem 2 because these problems are daily and smaller than the weekly problems. This is valid in case the experiment is applied both with the exact approach and with the hybrid approach.

Problem 3 is a weekly problem and the number of candidate Type A partial pairings is large. Therefore, the average duration increases as the parameter N is increased. If experiments are applied with the hybrid approach, parameter $fneg$ is very effective on the average durations. $fneg$ is the parameter which is related to the partial pairings that have negative reduced costs. As $fneg$ increases, the number of paths that are kept by approximate rule 1 and whose completed pairings are likely to have negative reduced costs is increased. A pairing with a negative reduced cost can be found by the approximate rules without applying the exact rule and hence the hybrid approach. Therefore, the computational time decreases as $fneg$ increases in problem 3 when the hybrid approach is applied.

Also, when the problems are solved by the hybrid approach, the topological order of the last node to which the exact rule is applied is usually large which means this node is close to the sink node. Therefore, the average durations in the hybrid approach are larger than the average durations in the exact approach for all experiments of all problems. Moreover, the effectiveness of the approximate rules and the exact rule is compared. At one iteration approximate rules can prune 68 % of all generated paths on the average whereas the exact rule can prune approximately 10 % of all generated paths.

The schedule and the detailed solutions for the problem with 42 flights are given in Appendix A. The main results are summarized below:

- The value of the initial feasible solution is 7858.
- Optimal solution of the problem is found (4822.25).
- There is one Type B pairing in the solution.

The schedule and the detailed solutions for the problem with 96 flights are given in Appendix B. The main results are summarized below:

- The value of the initial feasible solution is 18024.
- Optimal solution of the problem is found (13922.3).
- There is one Type B pairing in the solution.

The schedule and the detailed solutions for the problem with 135 flights are given in Appendix C. The main results are summarized below:

- The value of the initial feasible solution is 48565.
- Optimal solution of the problem is found (46730.3).
- There are two Type B pairings in the solution.

CHAPTER 5

CONCLUSION AND FUTURE RESEARCH

In this study, we focus on the pricing subproblem of the crew pairing and robust crew pairing problems when a column generation method is applied. We consider the extra flights at the planning level. We can find Type B solutions that can cover the extra flights and their required deadhead flights. We apply three pruning rules to reduce the computational time of the problem. Moreover, we use a buffer column pool that provides pairings with negative reduced costs without solving the multi-label shortest path problem.

We solved medium-to-large problem instances in very short times. However, there is a possibility that the large problems may not be solved by our pruning rules and buffer column pool method. Therefore, we develop a method to find a lower bound for large problems. Also, the robust crew pairing problem can be solved by applying the row-and-column generation algorithm. As a future research, we intend to study this algorithm to solve the robust crew pairing problem.

Bibliography

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

[2] Anbil, R., Forrest J.J., and Pulleyblank, W.R., Column generation and the crew pairing problem, Documenta Mathematica, Extra Volume ICM (3), 677-686, 1998.

[3] Andersson, E., Housos, E., Kohl, N. and Wedelin, D., Crew pairing optimization, In G. Yu, editor, Operations Research in the Airline Industry, Kluwer Academic Publishers, 228-258, 1998.

[4] Barnhart, C., Hatay, L. and Jonhson, E.L., Deadhead selection for the long-haul crew pairing problem, Operations Research, 43(3), 491-499, 1995.

[5] Barnhart, C., Jonhson, 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, 46(3), 316-329, 1998.

[6] Barnhart, C., Cohn, A.M., Johnson, E.L., Klabjan, D., Nemhauser, G.L., and Vance, P.H., Airline crew scheduling, In R. W. Hall, editor, Handbook of Transportation Science, Kluwer Scientific Publishers, 517-560, 2003.

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

[8] Brumbaugh-Smith, J., Shier, D., An empirical investigation of some bicriterion-shortest path algorithms, European Journal of Operational Research, 43, 216-224, 1989.

[9] Crainic, G., and Rousseau, J., The column generation principle and the airline crew scheduling problem, Informs, 25(2), 136-151, 1987.

[10] Çoban, E., Column generation approaches to a robust airline crew pairing model for managing extra flights, Master of Science thesis, Industrial Engineering, Sabanci University, Turkey, 2008.

[11] Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M. M. and Soumis, F., Daily aircraft routing and scheduling, Management Science, 43(6), 841-855, 1997.

[12] Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M. M. and Soumis, F., A unified framework for deterministic time constrained vehicle routing and crew scheduling problems, In T. Crainic and G. Laporte, editors, Fleet Management and Logistics, Kluwer Publishing Company, 57-93, 1998.

[13] Desrochers, M. and Soumis, F., A generalized permanent labeling algorithm for the shortest path problem with time windows, Infor, 26(3), 191-212, 1988.

[14] Desrochers, M., Desrosiers, J. and Solomon, M. M., A new optimization algorithm for the vehicle routing problem with time windows, Operations Research, 40(2), 342-354, 1992.

[15] Desrosiers, J., Dumas, Y., Solomon, M. M. and Soumis, F., Time constrained routing and scheduling, In M. Ball, editor, Handbooks in Operations Research and Management Science, Elsevier, 35-140, 1995.

[16] Forrest, J.J., Mathematical programming with a library of optimization subroutines, presented at the ORSA/TIMS Joint National Meeting, New York, 1989.

[17] Gamache, M., Soumis, F., Morquis, G. and Desrosiers, J., A column generation approach for large-scale aircrew rostering problems, Operations Research, 47(2), 247-263, 1999.

[18] Guerriero, F. and Musmanno, R., Label correcting methods to solve multicriteria shortest path problems, Journal of Optimization Theory and Applications, 111(3), 589-613, 2001.

[19] Klabjan, Diego, Large-scale models in the airline industry, In G. Desaulniers, J. Desrosiers, M.M. Solomon, editors, Column Generation, Kluwer Academic Publishers, 163-195, 2003.

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

[21] Nagih, A. and Soumis, F., Nodal aggregation of resource constraints in a shortest path problem, European Journal of Operational Research, 172, 500-514, 2006.

[22] Savelsbergh, M. and Sol, M., DRIVE: Dynamic routing of independent vehicles, Operations Research, 46(4), 474-490, 1998.

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

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

[25] Skriver, A.J.V. and Andersen, K.A., A label correcting approach for solving bicriterion shortest-path problems, Computers and Operations Research, 27, 507-524, 2000.

[26] Tekiner, H., Robust crew pairing for managing extra flights, Master of Science thesis, Industrial Engineering, Sabanci University, Turkey, 2006.

[27] Tekiner, H., Birbil, Ş. İ. and Bülbül, K., Robust crew pairing for managing extra flights, Technical Report, Sabanci University, Turkey, 2008.

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

[29] ILOG software, http://www.ilog.com, June 2008.

Appendix A

Solution of the problem with 42 Flights

ID	Dep-Arr	Dep.Time-Arr.Time	ID	Dep-Arr	Dep.Time-Arr.Time
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	23:05-00:05
21	IST-ADB	17:00-18:00	42	ESB-IST	00:00-00:55

Table A.1: Flight data with 42 flights.

Ex.	$fneg$	$fpos$	$fraction$	$flim$	N	**Ex.**	$fneg$	$fpos$	$fraction$	$flim$	N
1	0.2	0.1	8×10^{-4}	0.3	0	25	0.2	0.2	10^{-3}	0.3	0
2	0.2	0.1	8×10^{-4}	0.3	25	26	0.2	0.2	10^{-3}	0.3	25
3	0.2	0.1	8×10^{-4}	0.3	750	27	0.2	0.2	10^{-3}	0.3	750
4	0.2	0.1	8×10^{-4}	0.3	∞	28	0.2	0.2	10^{-3}	0.3	∞
5	0.2	0.1	8×10^{-4}	0.6	0	29	0.2	0.2	10^{-3}	0.6	0
6	0.2	0.1	8×10^{-4}	0.6	25	30	0.2	0.2	10^{-3}	0.6	25
7	0.2	0.1	8×10^{-4}	0.6	750	31	0.2	0.2	10^{-3}	0.6	750
8	0.2	0.1	8×10^{-4}	0.6	∞	32	0.2	0.2	10^{-3}	0.6	∞
9	0.2	0.1	10^{-3}	0.3	0	33	0.4	0.1	8×10^{-4}	0.3	0
10	0.2	0.1	10^{-3}	0.3	25	34	0.4	0.1	8×10^{-4}	0.3	25
11	0.2	0.1	10^{-3}	0.3	750	35	0.4	0.1	8×10^{-4}	0.3	750
12	0.2	0.1	10^{-3}	0.3	∞	36	0.4	0.1	8×10^{-4}	0.3	∞
13	0.2	0.1	10^{-3}	0.6	0	37	0.4	0.1	8×10^{-4}	0.6	0
14	0.2	0.1	10^{-3}	0.6	25	38	0.4	0.1	8×10^{-4}	0.6	25
15	0.2	0.1	10^{-3}	0.6	750	39	0.4	0.1	8×10^{-4}	0.6	750
16	0.2	0.1	10^{-3}	0.6	∞	40	0.4	0.1	8×10^{-4}	0.6	∞
17	0.2	0.2	8×10^{-4}	0.3	0	41	0.4	0.1	10^{-3}	0.3	0
18	0.2	0.2	8×10^{-4}	0.3	25	42	0.4	0.1	10^{-3}	0.3	25
19	0.2	0.2	8×10^{-4}	0.3	750	43	0.4	0.1	10^{-3}	0.3	750
20	0.2	0.2	8×10^{-4}	0.3	∞	44	0.4	0.1	10^{-3}	0.3	∞
21	0.2	0.2	8×10^{-4}	0.6	0	45	0.4	0.1	10^{-3}	0.6	0
22	0.2	0.2	8×10^{-4}	0.6	25	46	0.4	0.1	10^{-3}	0.6	25
23	0.2	0.2	8×10^{-4}	0.6	750	47	0.4	0.1	10^{-3}	0.6	750
24	0.2	0.2	8×10^{-4}	0.6	∞	48	0.4	0.1	10^{-3}	0.6	∞

Table A.2: Parameters of the first 48 experiments of the problem with 42 flights.

Ex.	$fneg$	$fpos$	$fraction$	$flim$	N	**Ex.**	$fneg$	$fpos$	$fraction$	$flim$	N
49	0.4	0.2	8×10^{-4}	0.3	0	73	0.8	0.1	10^{-3}	0.3	0
50	0.4	0.2	8×10^{-4}	0.3	25	74	0.8	0.1	10^{-3}	0.3	25
51	0.4	0.2	8×10^{-4}	0.3	750	75	0.8	0.1	10^{-3}	0.3	750
52	0.4	0.2	8×10^{-4}	0.3	∞	76	0.8	0.1	10^{-3}	0.3	∞
53	0.4	0.2	8×10^{-4}	0.6	0	77	0.8	0.1	10^{-3}	0.6	0
54	0.4	0.2	8×10^{-4}	0.6	25	78	0.8	0.1	10^{-3}	0.6	25
55	0.4	0.2	8×10^{-4}	0.6	750	79	0.8	0.1	10^{-3}	0.6	750
56	0.4	0.2	8×10^{-4}	0.6	∞	80	0.8	0.1	10^{-3}	0.6	∞
57	0.4	0.2	10^{-3}	0.3	0	81	0.8	0.2	8×10^{-4}	0.3	0
58	0.4	0.2	10^{-3}	0.3	25	82	0.8	0.2	8×10^{-4}	0.3	25
59	0.4	0.2	10^{-3}	0.3	750	83	0.8	0.2	8×10^{-4}	0.3	750
60	0.4	0.2	10^{-3}	0.3	∞	84	0.8	0.2	8×10^{-4}	0.3	∞
61	0.4	0.2	10^{-3}	0.6	0	85	0.8	0.2	8×10^{-4}	0.6	0
62	0.4	0.2	10^{-3}	0.6	25	86	0.8	0.2	8×10^{-4}	0.6	25
63	0.4	0.2	10^{-3}	0.6	750	87	0.8	0.2	8×10^{-4}	0.6	750
64	0.4	0.2	10^{-3}	0.6	∞	88	0.8	0.2	8×10^{-4}	0.6	∞
65	0.8	0.1	8×10^{-4}	0.3	0	89	0.8	0.2	10^{-3}	0.3	0
66	0.8	0.1	8×10^{-4}	0.3	25	90	0.8	0.2	10^{-3}	0.3	25
67	0.8	0.1	8×10^{-4}	0.3	750	91	0.8	0.2	10^{-3}	0.3	750
68	0.8	0.1	8×10^{-4}	0.3	∞	92	0.8	0.2	10^{-3}	0.3	∞
69	0.8	0.1	8×10^{-4}	0.6	0	93	0.8	0.2	10^{-3}	0.6	0
70	0.8	0.1	8×10^{-4}	0.6	25	94	0.8	0.2	10^{-3}	0.6	25
71	0.8	0.1	8×10^{-4}	0.6	750	95	0.8	0.2	10^{-3}	0.6	750
72	0.8	0.1	8×10^{-4}	0.6	∞	96	0.8	0.2	10^{-3}	0.6	∞

Table A.3: Parameters of the last 48 experiments of the problem with 42 flights.

Expr.	Dur.(sec.)	Expr.	Dur.(sec.)	Expr.	Dur.(sec.)
1	0.0157	33	0.016	65	0.0160
2	0.0158	34	0.0160	66	0.0160
3	0.0157	35	0.0160	67	0.0154
4	0.0158	36	0.0158	68	0.0160
5	0.0160	37	0.0160	69	0.0158
6	0.0157	38	0.0160	70	0.0158
7	0.0157	39	0.0156	71	0.0158
8	0.0158	40	0.0158	72	0.0156
9	0.0158	41	0.0158	73	0.0160
10	0.0157	42	0.0156	74	0.0156
11	0.0157	43	0.0156	75	0.0156
12	0.0157	44	0.0156	76	0.0158
13	0.0160	45	0.0160	77	0.0157
14	0.0158	46	0.0156	78	0.0156
15	0.0157	47	0.0158	79	0.0158
16	0.0160	48	0.0156	80	0.0158
17	0.0158	49	0.0158	81	0.0160
18	0.0158	50	0.0158	82	0.0158
19	0.0160	51	0.0160	83	0.0156
20	0.0157	52	0.0160	84	0.0156
21	0.0157	53	0.0160	85	0.0160
22	0.0158	54	0.0160	86	0.0156
23	0.0157	55	0.0156	87	0.0158
24	0.0155	56	0.0158	88	0.0158
25	0.0158	57	0.0158	89	0.0160
26	0.0158	58	0.0154	90	0.0158
27	0.0158	59	0.0158	91	0.0158
28	0.0160	60	0.0154	92	0.0160
29	0.0158	61	0.0160	93	0.0158
30	0.0157	62	0.0158	94	0.0158
31	0.0160	63	0.0158	95	0.0154
32	0.0158	64	0.0160	96	0.0158

Table A.4: Average CPU times for an iteration of the MLSP of the problem with 42 flights with exact approach.

Expr.	Dur.(sec.)	Expr.	Dur.(sec.)	Expr.	Dur.(sec.)
1	0.0418	33	0.0418	65	0.0417
2	0.0417	34	0.0530	66	0.0560
3	0.0442	35	0.0592	67	0.0658
4	0.0522	36	0.0594	68	0.0626
5	0.0420	37	0.0418	69	0.0368
6	0.0418	38	0.0530	70	0.0500
7	0.0522	39	0.0596	71	0.0660
8	0.0468	40	0.0532	72	0.0624
9	0.0338	41	0.0365	73	0.0367
10	0.0418	42	0.0532	74	0.0504
11	0.0445	43	0.0532	75	0.0624
12	0.0442	44	0.0594	76	0.0628
13	0.0395	45	0.0420	77	0.0418
14	0.0443	46	0.0498	78	0.0564
15	0.0468	47	0.0532	79	0.0658
16	0.0390	48	0.0562	80	0.0594
17	0.0368	49	0.0420	81	0.0420
18	0.0417	50	0.0436	82	0.0566
19	0.0520	51	0.0562	83	0.0598
20	0.0445	52	0.0498	84	0.0596
21	0.0392	53	0.0420	85	0.0417
22	0.0445	54	0.0406	86	0.0566
23	0.0498	55	0.0532	87	0.0628
24	0.0470	56	0.0502	88	0.0624
25	0.0420	57	0.0390	89	0.0392
26	0.0417	58	0.0438	90	0.0596
27	0.0467	59	0.0562	91	0.0594
28	0.0492	60	0.0658	92	0.0656
29	0.0418	61	0.0415	93	0.0418
30	0.0443	62	0.0500	94	0.0566
31	0.0495	63	0.0532	95	0.0628
32	0.0522	64	0.0562	96	0.0658

Table A.5: Average CPU times for an iteration of the MLSP of the problem with 42 flights with hybrid approach.

Iteration	**Dur. (sec.)**	**Method**
1	0.016	Approximate
2-27	-	Buffer
28	0.016	Approximate
29-42	-	Buffer
43	0.016	Approximate
44-50	-	Buffer
51	0.016	Approximate
52	-	Buffer
53	0.031	Approximate + Exact

Table A.6: The duration of each iteration of the problem with 42 flights in Experiment 26 (exact method).

Iteration	**Dur. (sec.)**	**Method**	**Last Node ***
1	0.016	Approximate	-
2-27	-	Buffer	-
28	0.015	Approximate	-
29-42	-	Buffer	-
43	0.015	Approximate	-
44-50	-	Buffer	-
51	0.016	Approximate	-
52	-	Buffer	-
53	0.188	Approximate + Hybrid	86 (sink)

* : The topological order of the last node to which the exact rule is applied.

Table A.7: The duration of each iteration of the problem with 42 flights in Experiment 26 (hybrid method).

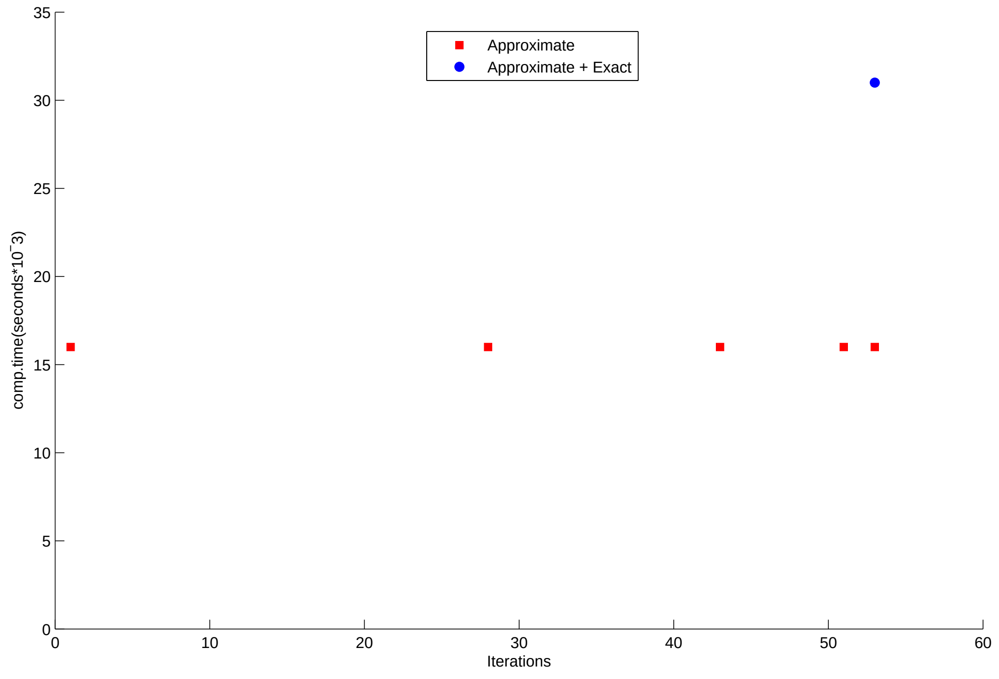

Figure A.1: Results associated with Table A.6.

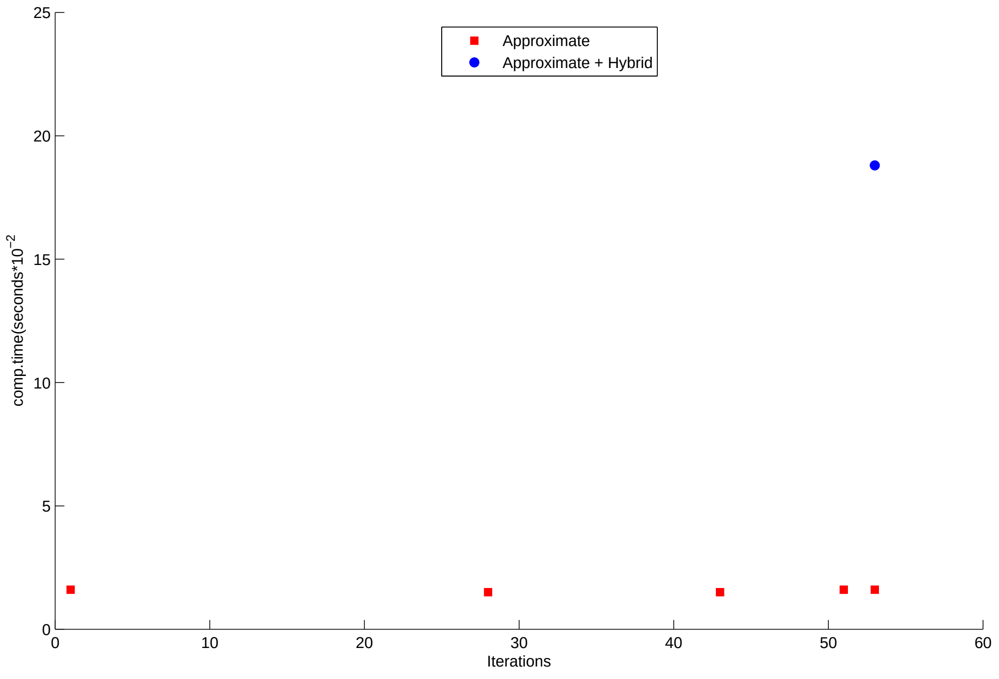

Figure A.2: Results associated with Table A.7.

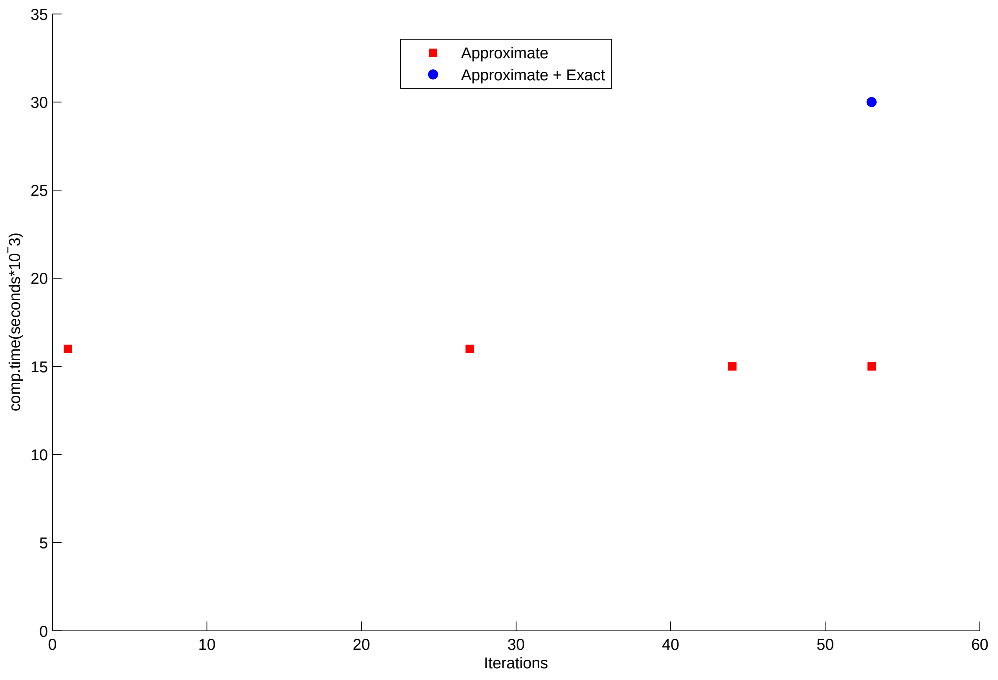

Figure A.3: Results associated with Experiment 46 (exact method).

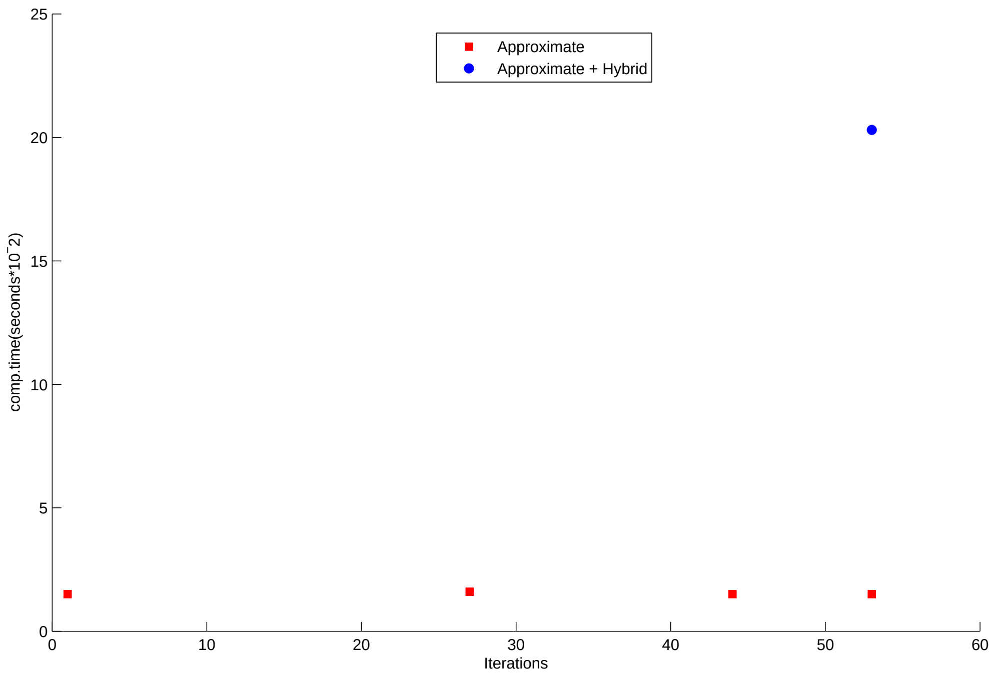

Figure A.4: Results associated with Experiment 46 (hybrid method).

Appendix B

Solution of the problem with 96 Flights

ID	Dep-Arr	Dep.Time-Arr.Time	ID	Dep-Arr	Dep.Time-Arr.Time
1	IST-ESB	04:00-05:05	25	IST-ESB	16:15-17:20
2	IST-ESB	05:10-06:15	26	ESB-IST	17:00-18:05
3	ESB-IST	04:15-05:20	27	IST-ESB	17:00-18:05
4	ESB-IST	05:30-06:35	28	IST-ESB	18:00-19:05
5	IST-ESB	06:40-07:45	29	IST-ESB	19:00-20:05
6	ESB-IST	06:00-07:05	30	ESB-IST	19:00-20:05
7	ESB-IST	07:00-08:05	31	ESB-IST	20:00-21:05
8	ESB-IST	07:30-08:35	32	IST-ESB	20:00-21:05
9	IST-ESB	07:00-08:05	33	IST-ADB	05:00-06:05
10	ESB-IST	08:00-09:05	34	ADB-IST	07:05-08:10
11	IST-ESB	09:00-10:05	35	IST-ADB	06:00-07:05
12	IST-ESB	10:00-11:05	36	ADB-IST	08:05-09:10
13	ESB-IST	09:00-10:05	37	IST-ADB	06:40-07:45
14	IST-ESB	11:00-12:05	38	ADB-IST	08:45-09:50
15	ESB-IST	11:00-12:05	39	IST-ADB	07:00-08:05
16	IST-ESB	13:00-14:05	40	ADB-IST	09:05-10:10
17	ESB-IST	12:00-13:05	41	IST-ADB	09:00-10:05
18	IST-ESB	14:00-15:05	42	ADB-IST	11:05-12:10
19	ESB-IST	13:00-14:05	43	IST-ADB	11:00-12:05
20	ESB-IST	14:00-15:05	44	ADB-IST	13:10-14:15
21	ESB-IST	15:00-16:05	45	IST-ADB	13:00-14:05
22	IST-ESB	15:00-16:05	46	ADB-IST	15:05-16:10
23	ESB-IST	16:00-17:05	47	IST-ADB	14:00-15:05
24	IST-ESB	16:00-17:05	48	ADB-IST	16:10-17:15

Table B.1: Flight data with 96 flights (first 48 flights).

ID	**Dep-Arr**	**Dep.Time-Arr.Time**	**ID**	**Dep-Arr**	**Dep.Time-Arr.Time**
49	IST-ADB	16:00-17:05	73	IST-AYT	17:00-18:15
50	ADB-IST	18:05-19:10	74	AYT-IST	19:15-20:30
51	IST-ADB	17:00-18:05	75	IST-AYT	18:30-19:45
52	ADB-IST	19:05-20:10	76	AYT-IST	20:45-22:00
53	IST-ADB	21:45-22:50	77	IST-AYT	21:55-23:10
54	ADB-IST	23:50-00:55	78	AYT-IST	00:15-01:30
55	ESB-ADB	05:45-07:00	79	IST-ADA	06:20-07:50
56	ADB-ESB	08:00-09:15	80	ADA-IST	08:50-10:20
57	ESB-ADB	15:00-16:15	81	IST-ADA	15:00-16:30
58	ADB-ESB	17:15-18:30	82	ADA-IST	17:30-19:00
59	ESB-ADB	20:50-22:05	83	IST-ADA	17:20-18:50
60	ADB-ESB	23:10-00:25	84	ADA-IST	19:55-21:35
61	ESB-AYT	04:15-05:15	85	IST-ADA	12:15-13:45
62	AYT-ESB	06:15-07:15	86	ADA-IST	14:45-16:25
63	ESB-AYT	19:00-20:00	87	IST-ADA	14:00-15:30
64	AYT-ESB	21:00-22:00	88	ADA-IST	16:30-18:00
65	IST-AYT	06:25-07:40	89	IST-ADA	19:30-21:00
66	AYT-IST	08:40-09:55	90	ADA-IST	22:00-23:30
67	IST-AYT	09:30-10:45	91	IST-ADA	09:15-10:45
68	AYT-IST	11:45-13:00	92	ADA-IST	11:45-13:15
69	IST-AYT	12:45-14:00	93	IST-ADA	21:35-23:05
70	AYT-IST	15:00-16:15	94	ADA-IST	01:30-03:00
71	IST-AYT	15:30-16:45	95	ESB-ADA	17:30-18:30
72	AYT-IST	17:55-19:10	96	ADA-ESB	19:30-20:30

Table B.2: Flight data with 96 flights (last 48 flights).

Ex.	$fneg$	$fpos$	$fraction$	$flim$	N	**Ex.**	$fneg$	$fpos$	$fraction$	$flim$	N
1	0.2	0.1	8×10^{-4}	0.3	0	25	0.2	0.2	10^{-3}	0.3	0
2	0.2	0.1	8×10^{-4}	0.3	50	26	0.2	0.2	10^{-3}	0.3	50
3	0.2	0.1	8×10^{-4}	0.3	1000	27	0.2	0.2	10^{-3}	0.3	1000
4	0.2	0.1	8×10^{-4}	0.3	∞	28	0.2	0.2	10^{-3}	0.3	∞
5	0.2	0.1	8×10^{-4}	0.6	0	29	0.2	0.2	10^{-3}	0.6	0
6	0.2	0.1	8×10^{-4}	0.6	50	30	0.2	0.2	10^{-3}	0.6	50
7	0.2	0.1	8×10^{-4}	0.6	1000	31	0.2	0.2	10^{-3}	0.6	1000
8	0.2	0.1	8×10^{-4}	0.6	∞	32	0.2	0.2	10^{-3}	0.6	∞
9	0.2	0.1	10^{-3}	0.3	0	33	0.4	0.1	8×10^{-4}	0.3	0
10	0.2	0.1	10^{-3}	0.3	50	34	0.4	0.1	8×10^{-4}	0.3	50
11	0.2	0.1	10^{-3}	0.3	1000	35	0.4	0.1	8×10^{-4}	0.3	1000
12	0.2	0.1	10^{-3}	0.3	∞	36	0.4	0.1	8×10^{-4}	0.3	∞
13	0.2	0.1	10^{-3}	0.6	0	37	0.4	0.1	8×10^{-4}	0.6	0
14	0.2	0.1	10^{-3}	0.6	50	38	0.4	0.1	8×10^{-4}	0.6	50
15	0.2	0.1	10^{-3}	0.6	1000	39	0.4	0.1	8×10^{-4}	0.6	1000
16	0.2	0.1	10^{-3}	0.6	∞	40	0.4	0.1	8×10^{-4}	0.6	∞
17	0.2	0.2	8×10^{-4}	0.3	0	41	0.4	0.1	10^{-3}	0.3	0
18	0.2	0.2	8×10^{-4}	0.3	50	42	0.4	0.1	10^{-3}	0.3	50
19	0.2	0.2	8×10^{-4}	0.3	1000	43	0.4	0.1	10^{-3}	0.3	1000
20	0.2	0.2	8×10^{-4}	0.3	∞	44	0.4	0.1	10^{-3}	0.3	∞
21	0.2	0.2	8×10^{-4}	0.6	0	45	0.4	0.1	10^{-3}	0.6	0
22	0.2	0.2	8×10^{-4}	0.6	50	46	0.4	0.1	10^{-3}	0.6	50
23	0.2	0.2	8×10^{-4}	0.6	1000	47	0.4	0.1	10^{-3}	0.6	1000
24	0.2	0.2	8×10^{-4}	0.6	∞	48	0.4	0.1	10^{-3}	0.6	∞

Table B.3: Parameters of the first 48 experiments of the problem with 96 flights.

Ex.	$fneg$	$fpos$	$fraction$	$flim$	N	**Ex.**	$fneg$	$fpos$	$fraction$	$flim$	N
49	0.4	0.2	8×10^{-4}	0.3	0	73	0.8	0.1	10^{-3}	0.3	0
50	0.4	0.2	8×10^{-4}	0.3	50	74	0.8	0.1	10^{-3}	0.3	50
51	0.4	0.2	8×10^{-4}	0.3	1000	75	0.8	0.1	10^{-3}	0.3	1000
52	0.4	0.2	8×10^{-4}	0.3	∞	76	0.8	0.1	10^{-3}	0.3	∞
53	0.4	0.2	8×10^{-4}	0.6	0	77	0.8	0.1	10^{-3}	0.6	0
54	0.4	0.2	8×10^{-4}	0.6	50	78	0.8	0.1	10^{-3}	0.6	50
55	0.4	0.2	8×10^{-4}	0.6	1000	79	0.8	0.1	10^{-3}	0.6	1000
56	0.4	0.2	8×10^{-4}	0.6	∞	80	0.8	0.1	10^{-3}	0.6	∞
57	0.4	0.2	10^{-3}	0.3	0	81	0.8	0.2	8×10^{-4}	0.3	0
58	0.4	0.2	10^{-3}	0.3	50	82	0.8	0.2	8×10^{-4}	0.3	50
59	0.4	0.2	10^{-3}	0.3	1000	83	0.8	0.2	8×10^{-4}	0.3	1000
60	0.4	0.2	10^{-3}	0.3	∞	84	0.8	0.2	8×10^{-4}	0.3	∞
61	0.4	0.2	10^{-3}	0.6	0	85	0.8	0.2	8×10^{-4}	0.6	0
62	0.4	0.2	10^{-3}	0.6	50	86	0.8	0.2	8×10^{-4}	0.6	50
63	0.4	0.2	10^{-3}	0.6	1000	87	0.8	0.2	8×10^{-4}	0.6	1000
64	0.4	0.2	10^{-3}	0.6	∞	88	0.8	0.2	8×10^{-4}	0.6	∞
65	0.8	0.1	8×10^{-4}	0.3	0	89	0.8	0.2	10^{-3}	0.3	0
66	0.8	0.1	8×10^{-4}	0.3	50	90	0.8	0.2	10^{-3}	0.3	50
67	0.8	0.1	8×10^{-4}	0.3	1000	91	0.8	0.2	10^{-3}	0.3	1000
68	0.8	0.1	8×10^{-4}	0.3	∞	92	0.8	0.2	10^{-3}	0.3	∞
69	0.8	0.1	8×10^{-4}	0.6	0	93	0.8	0.2	10^{-3}	0.6	0
70	0.8	0.1	8×10^{-4}	0.6	50	94	0.8	0.2	10^{-3}	0.6	50
71	0.8	0.1	8×10^{-4}	0.6	1000	95	0.8	0.2	10^{-3}	0.6	1000
72	0.8	0.1	8×10^{-4}	0.6	∞	96	0.8	0.2	10^{-3}	0.6	∞

Table B.4: Parameters of the last 48 experiments of the problem with 96 flights.

Expr.	**Dur.(sec.)**	**Expr.**	**Dur.(sec.)**	**Expr.**	**Dur.(sec.)**
1	0.0180	33	0.0179	65	0.0183
2	0.0195	34	0.0178	66	0.0178
3	0.0196	35	0.0176	67	0.0196
4	0.0178	36	0.0198	68	0.0176
5	0.0180	37	0.018	69	0.0183
6	0.0178	38	0.0176	70	0.0175
7	0.0196	39	0.0178	71	0.0196
8	0.0198	40	0.0196	72	0.0178
9	0.0180	41	0.0183	73	0.0181
10	0.0176	42	0.0178	74	0.0176
11	0.0176	43	0.0178	75	0.0215
12	0.0176	44	0.0216	76	0.0176
13	0.0180	45	0.0181	77	0.0181
14	0.0178	46	0.0196	78	0.0176
15	0.0174	47	0.0195	79	0.0198
16	0.0179	48	0.0176	80	0.0194
17	0.0181	49	0.0181	81	0.0177
18	0.0178	50	0.0194	82	0.0215
19	0.0196	51	0.0176	83	0.0196
20	0.0178	52	0.0176	84	0.0175
21	0.0181	53	0.0180	85	0.0179
22	0.0178	54	0.0179	86	0.0179
23	0.0198	55	0.0214	87	0.0196
24	0.0195	56	0.0178	88	0.0178
25	0.0181	57	0.0177	89	0.0179
26	0.0178	58	0.0198	90	0.0178
27	0.0216	59	0.0178	91	0.0199
28	0.0178	60	0.0195	92	0.0176
29	0.0180	61	0.0180	93	0.0180
30	0.0195	62	0.0195	94	0.0179
31	0.0176	63	0.0178	95	0.0196
32	0.0196	64	0.0179	96	0.0196

Table B.5: Average CPU times for an iteration of the MLSP of the problem with 96 flights with exact approach.

Expr.	**Dur.(sec.)**	**Expr.**	**Dur.(sec.)**	**Expr.**	**Dur.(sec.)**
1	0.1633	33	0.1517	65	0.1537
2	0.1409	34	0.1366	66	0.1760
3	0.1405	35	0.1389	67	0.1408
4	0.1426	36	0.1388	68	0.1366
5	0.1589	37	0.1544	69	0.1540
6	0.1366	38	0.1370	70	0.1405
7	0.1389	39	0.1408	71	0.1426
8	0.1386	40	0.1408	72	0.1699
9	0.1541	41	0.1540	73	0.1519
10	0.1386	42	0.1388	74	0.1406
11	0.1406	43	0.1366	75	0.1366
12	0.1386	44	0.1368	76	0.1368
13	0.1564	45	0.1564	77	0.1519
14	0.1369	46	0.1368	78	0.1856
15	0.1388	47	0.1368	79	0.1683
16	0.1408	48	0.1409	80	0.1406
17	0.1543	49	0.1521	81	0.1987
18	0.1369	50	0.1368	82	0.1366
19	0.1406	51	0.1388	83	0.1660
20	0.1406	52	0.1370	84	0.1700
21	0.1541	53	0.1517	85	0.1583
22	0.1369	54	0.1369	86	0.1388
23	0.1349	55	0.1368	87	0.1425
24	0.1388	56	0.1406	88	0.1426
25	0.1541	57	0.1541	89	0.1609
26	0.1349	58	0.1368	90	0.1505
27	0.1386	59	0.1386	91	0.1524
28	0.1389	60	0.1388	92	0.1506
29	0.1541	61	0.1654	93	0.1653
30	0.1386	62	0.1428	94	0.1583
31	0.1349	63	0.1485	95	0.1525
32	0.1426	64	0.1428	96	0.1484

Table B.6: Average CPU times for an iteration of the MLSP of the problem with 96 flights with hybrid approach.

Iteration	**Dur. (sec.)**	**Method**
1	0.032	Approximate
2-45	-	Buffer
46	0.016	Approximate
47-60	-	Buffer
61	0.015	Approximate
62-73	-	Buffer
74	0.015	Approximate
75-76	-	Buffer
77	0.031	Approximate
78-79	-	Buffer
80	0.031	Approximate
81-83	-	Buffer
84	0.031	Approximate + Exact

Table B.7: The duration of each iteration of the problem with 96 flights in Experiment 2 (exact method).

Iteration	**Dur. (sec.)**	**Method**	**Last Node ***
1	0.031	Approximate	-
2-45	-	Buffer	-
46	0.016	Approximate	-
47-60	-	Buffer	-
61	0.016	Approximate	-
62-73	-	Buffer	-
74	0.016	Approximate	-
75-76	-	Buffer	-
77	0.016	Approximate	-
78-79	-	Buffer	-
80	0.016	Approximate	-
81-83	-	Buffer	-
84	1.015	Approximate + Hybrid	194 (sink node)

* : The topological order of the last node to which the exact rule is applied.

Table B.8: The duration of each iteration of the problem with 96 flights in Experiment 2 (hybrid method).

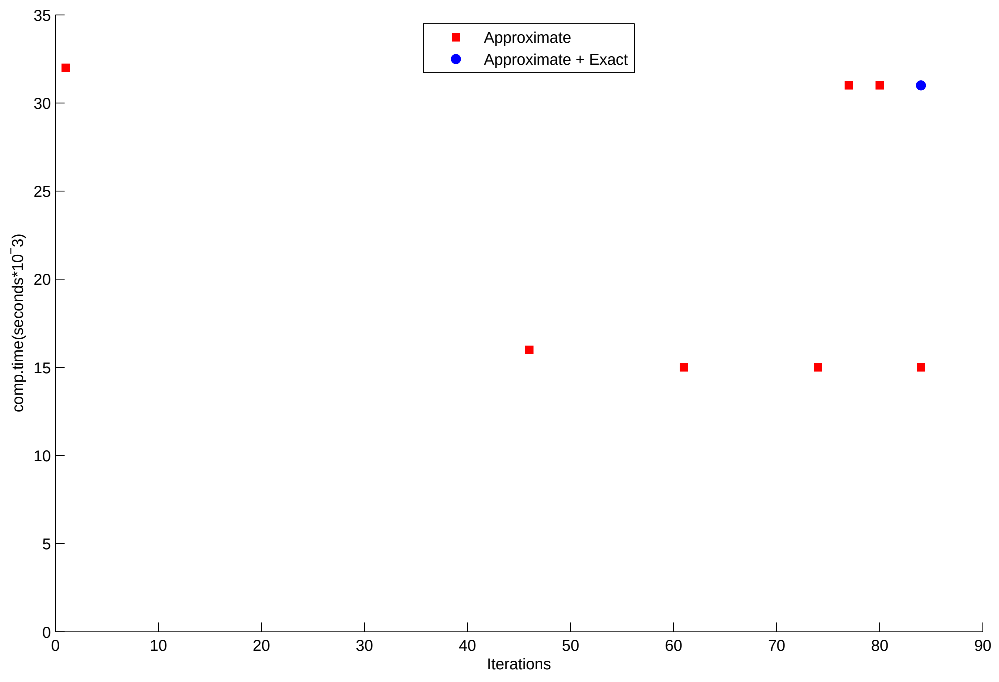

Figure B.1: Results associated with Table B.7.

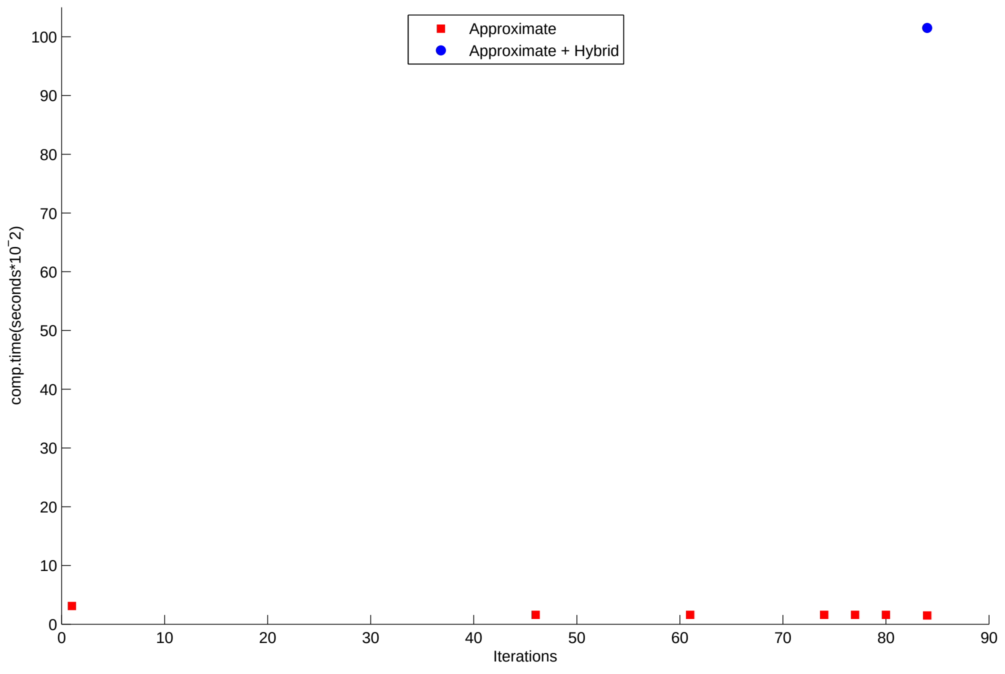

Figure B.2: Results associated with Table B.8.

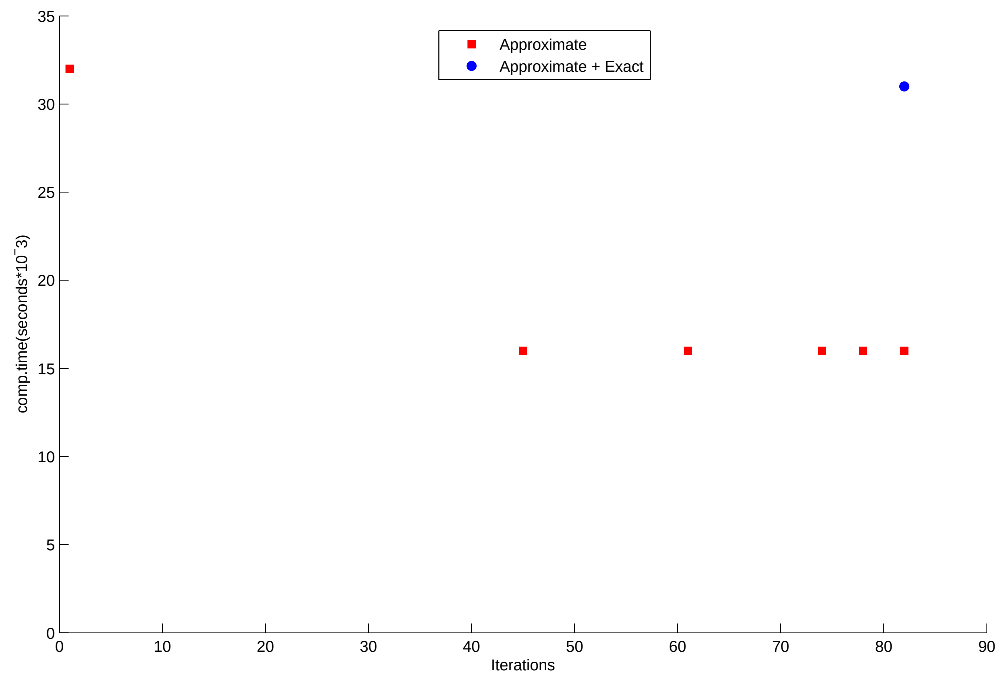

Figure B.3: Results associated with Experiment 25 (exact method).

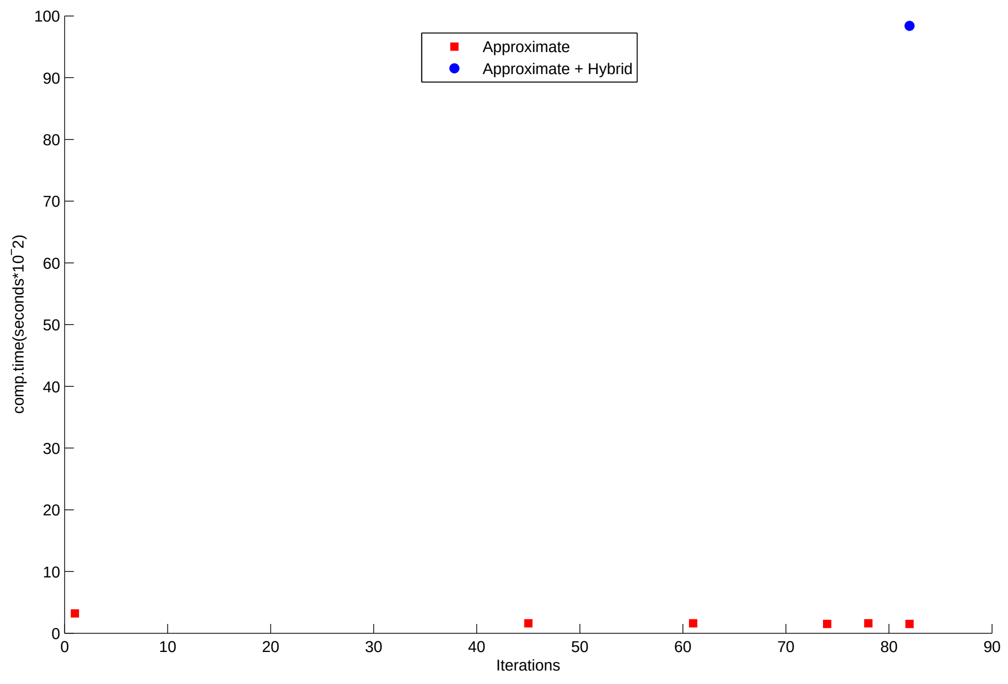

Figure B.4: Results associated with Experiment 25 (hybrid method).

Appendix C

Solution of the problem with 135 Flights

ID	Dep-Arr	DT-AT	DD-AD	ID	Dep-Arr	DT-AT	DD-AD
1	IST-ADA	08:15-09:50	Day 1-Day 1	32	FRA-IST	21:00-23:55	Day 2-Day 2
2	ADA-IST	10:45-12:20	Day 1-Day 1	33	IST-ESB	16:00-17:00	Day 3-Day 3
3	IST-DEL	15:10-21:00	Day 1-Day 1	34	ESB-IST	18:00-19:00	Day 3-Day 3
4	DEL-IST	22:30-05:45	Day 1-Day 2	35	IST-ADB	06:00-07:00	Day 3-Day 3
5	IST-ALG	07:40-11:25	Day 1-Day 1	36	ADB-IST	08:00-09:00	Day 3-Day 3
6	ALG-IST	12:25-15:55	Day 1-Day 1	37	IST-ADB	10:00-11:00	Day 3-Day 3
7	IST-DXB	17:00-21:10	Day 1-Day 1	38	ADB-IST	12:00-13:00	Day 3-Day 3
8	DXB-IST	22:50-03:40	Day 1-Day 2	39	IST-DEL	15:10-21:00	Day 3-Day 3
9	ALA-IST	00:20-06:30	Day 1-Day 1	40	DEL-IST	22:30-05:45	Day 3-Day 4
10	IST-ALA	13:55-19:00	Day 1-Day 1	41	IST-ALG	07:40-11:25	Day 3-Day 3
11	ALA-IST	20:45-02:55	Day 1-Day 2	42	ALG-IST	12:25-15:55	Day 3-Day 3
12	IST-ORY	04:50-08:25	Day 1-Day 1	43	IST-ALA	17:25-22:35	Day 3-Day 3
13	ORY-IST	09:30-12:55	Day 1-Day 1	44	ALA-IST	00:20-06:30	Day 3-Day 3
14	IST-MST	12:40-15:55	Day 1-Day 1	45	IST-GVA	07:35-10:45	Day 3-Day 3
15	MST-IST	16:50-19:50	Day 1-Day 1	46	GVA-IST	11:45-14:45	Day 3-Day 3
16	IST-LGW	01:20-05:20	Day 1-Day 1	47	IST-MST	01:45-05:00	Day 3-Day 3
17	LGW-IST	06:50-10:35	Day 1-Day 1	48	MST-IST	06:00-09:00	Day 3-Day 3
18	IST-ESB	06:00-07:00	Day 2-Day 2	49	IST-FRA	16:45-19:45	Day 3-Day 3
19	IST-ESB	08:00-09:00	Day 2-Day 2	50	FRA-IST	20:45-23:45	Day 3-Day 3
20	ESB-IST	08:00-09:00	Day 2-Day 2	51	IST-ESB	08:00-09:00	Day 4-Day 4
21	IST-ESB	10:00-11:00	Day 2-Day 2	52	ESB-IST	10:00-11:00	Day 4-Day 4
22	ESB-IST	10:00-11:00	Day 2-Day 2	53	IST-ADA	20:35-22:10	Day 4-Day 4
23	ESB-IST	12:00-13:00	Day 2-Day 2	54	IST-DEL	15:10-21:00	Day 4-Day 4
24	IST-BOM	15:45-21:45	Day 2-Day 2	55	DEL-IST	22:30-05:45	Day 4-Day 5
25	BOM-IST	23:25-05:45	Day 2-Day 3	56	IST-LOS	12:00-18:40	Day 4-Day 4
26	IST-LOS	12:00-18:40	Day 2-Day 2	57	LOS-IST	20:40-02:40	Day 4-Day 5
27	LOS-IST	20:40-02:40	Day 2-Day 3	58	IST-ALG	07:40-11:25	Day 4-Day 4
28	IST-ALG	07:40-11:25	Day 2-Day 2	59	ALG-IST	12:25-15:55	Day 4-Day 4
29	ALG-IST	12:25-15:55	Day 2-Day 2	60	IST-ALA	17:25-22:35	Day 4-Day 4
30	IST-ALA	17:25-22:35	Day 2-Day 2	61	ALA-IST	00:20-06:30	Day 4-Day 4
31	IST-FRA	17:00-20:05	Day 2-Day 2	62	IST-FRA	12:05-15:05	Day 4-Day 4

Table C.1: Flight data with 135 flights (first 62 flights).

ID	Dep-Arr	DT-AT	DD-AD	ID	Dep-Arr	DT-AT	DD-AD
63	FRA-IST	16:00-18:55	Day 4-Day 4	100	LOS-IST	20:40-02:40	Day 6-Day 7
64	IST-MUC	04:50-07:30	Day 4-Day 4	101	IST-ALA	13:55-19:00	Day 6-Day 6
65	MUC-IST	08:30-11:05	Day 4-Day 4	102	ALA-IST	20:45-02:55	Day 6-Day 7
66	IST-MST	01:15-04:30	Day 4-Day 4	103	IST-DUS	05:05-08:25	Day 6-Day 6
67	MST-IST	05:30-08:30	Day 4-Day 4	104	DUS-IST	09:25-12:35	Day 6-Day 6
68	IST-TLV	09:55-12:00	Day 4-Day 4	105	IST-FRA	05:30-08:35	Day 6-Day 6
69	TLV-IST	13:15-15:20	Day 4-Day 4	106	FRA-IST	09:45-12:40	Day 6-Day 6
70	IST-FRA	17:00-20:00	Day 4-Day 4	107	IST-ORY	04:50-08:25	Day 6-Day 6
71	FRA-IST	21:00-23:55	Day 4-Day 4	108	ORY-IST	09:30-12:55	Day 6-Day 6
72	IST-ESB	06:00-07:00	Day 5-Day 5	109	IST-MST	15:30-18:45	Day 6-Day 6
73	ESB-IST	08:00-09:00	Day 5-Day 5	110	MST-IST	19:45-22:45	Day 6-Day 6
74	IST-ESB	14:00-15:00	Day 5-Day 5	111	IST-TLV	06:30-08:30	Day 6-Day 6
75	ESB-IST	16:00-17:00	Day 5-Day 5	112	TLV-IST	09:45-11:55	Day 6-Day 6
76	IST-ESB	18:00-19:00	Day 5-Day 5	113	IST-ESB	16:00-17:00	Day 7-Day 7
77	ESB-IST	20:00-21:05	Day 5-Day 5	114	ESB-IST	18:00-19:00	Day 7-Day 7
78	ADA-IST	02:00-03:30	Day 5-Day 5	115	IST-ADB	05:30-06:30	Day 7-Day 7
79	IST-ADA	08:15-09:50	Day 5-Day 5	116	ADB-IST	07:30-08:30	Day 7-Day 7
80	ADA-IST	10:45-12:20	Day 5-Day 5	117	IST-AYT	11:25-12:40	Day 7-Day 7
81	IST-BOM	15:45-21:45	Day 5-Day 5	118	AYT-IST	13:45-15:00	Day 7-Day 7
82	BOM-IST	23:25-05:45	Day 5-Day 6	119	IST-GZT	14:20-16:05	Day 7-Day 7
83	ALA-IST	00:20-06:30	Day 5-Day 5	120	GZT-IST	17:00-18:50	Day 7-Day 7
84	IST-ALA	13:55-19:00	Day 5-Day 5	121	IST-BOM	15:45-21:45	Day 7-Day 7
85	ALA-IST	20:45-02:55	Day 5-Day 6	122	BOM-IST	23:25-05:45	Day 7-Day 8
86	IST-FRA	05:30-08:35	Day 5-Day 5	123	IST-TLV	04:50-06:45	Day 7-Day 7
87	FRA-IST	09:45-12:40	Day 5-Day 5	124	TLV-IST	07:40-09:45	Day 7-Day 7
88	IST-GVA	07:35-10:45	Day 5-Day 5	125	IST-TLV	11:15-13:20	Day 7-Day 7
89	GVA-IST	11:45-14:45	Day 5-Day 5	126	TLV-IST	14:20-16:25	Day 7-Day 7
90	IST-ORY	10:50-14:25	Day 5-Day 5	127	IST-ALA	17:25-22:35	Day 7-Day 7
91	ORY-IST	15:35-19:00	Day 5-Day 5	128	IST-GVA	07:35-10:45	Day 7-Day 7
92	ALA-DEL	12:15-16:15	Day 5-Day 5	129	GVA-IST	11:45-14:45	Day 7-Day 7
93	IST-ALA	06:10-11:15	Day 5-Day 5	130	IST-ORY	04:50-08:25	Day 7-Day 7
94	DEL-IST	17:15-00:15	Day 5-Day 6	131	ORY-IST	09:30-12:55	Day 7-Day 7
95	IST-GZT	14:20-16:05	Day 6-Day 6	132	IST-MST	13:30-16:45	Day 7-Day 7
96	GZT-IST	17:00-18:50	Day 6-Day 6	133	MST-IST	18:10-21:10	Day 7-Day 7
97	IST-DEL	15:10-21:00	Day 6-Day 6	134	IST-FRA	00:50-03:50	Day 7-Day 7
98	DEL-IST	22:30-05:45	Day 6-Day 7	135	FRA-IST	05:35-08:35	Day 7-Day 7
99	IST-LOS	12:00-18:40	Day 6-Day 6	-	-	-	-

Table C.2: Flight data with 135 flights (last 73 flights).

Ex.	$fneg$	$fpos$	$fraction$	$flim$	N	**Ex.**	$fneg$	$fpos$	$fraction$	$flim$	N
1	0.2	0.1	8×10^{-4}	0.3	0	25	0.2	0.2	10^{-3}	0.3	0
2	0.2	0.1	8×10^{-4}	0.3	25	26	0.2	0.2	10^{-3}	0.3	25
3	0.2	0.1	8×10^{-4}	0.3	50	27	0.2	0.2	10^{-3}	0.3	50
4	0.2	0.1	8×10^{-4}	0.3	75	28	0.2	0.2	10^{-3}	0.3	75
5	0.2	0.1	8×10^{-4}	0.6	0	29	0.2	0.2	10^{-3}	0.6	0
6	0.2	0.1	8×10^{-4}	0.6	25	30	0.2	0.2	10^{-3}	0.6	25
7	0.2	0.1	8×10^{-4}	0.6	50	31	0.2	0.2	10^{-3}	0.6	50
8	0.2	0.1	8×10^{-4}	0.6	75	32	0.2	0.2	10^{-3}	0.6	75
9	0.2	0.1	10^{-3}	0.3	0	33	0.4	0.1	8×10^{-4}	0.3	0
10	0.2	0.1	10^{-3}	0.3	25	34	0.4	0.1	8×10^{-4}	0.3	25
11	0.2	0.1	10^{-3}	0.3	50	35	0.4	0.1	8×10^{-4}	0.3	50
12	0.2	0.1	10^{-3}	0.3	75	36	0.4	0.1	8×10^{-4}	0.3	75
13	0.2	0.1	10^{-3}	0.6	0	37	0.4	0.1	8×10^{-4}	0.6	0
14	0.2	0.1	10^{-3}	0.6	25	38	0.4	0.1	8×10^{-4}	0.6	25
15	0.2	0.1	10^{-3}	0.6	50	39	0.4	0.1	8×10^{-4}	0.6	50
16	0.2	0.1	10^{-3}	0.6	75	40	0.4	0.1	8×10^{-4}	0.6	75
17	0.2	0.2	8×10^{-4}	0.3	0	41	0.4	0.1	10^{-3}	0.3	0
18	0.2	0.2	8×10^{-4}	0.3	25	42	0.4	0.1	10^{-3}	0.3	25
19	0.2	0.2	8×10^{-4}	0.3	50	43	0.4	0.1	10^{-3}	0.3	50
20	0.2	0.2	8×10^{-4}	0.3	75	44	0.4	0.1	10^{-3}	0.3	75
21	0.2	0.2	8×10^{-4}	0.6	0	45	0.4	0.1	10^{-3}	0.6	0
22	0.2	0.2	8×10^{-4}	0.6	25	46	0.4	0.1	10^{-3}	0.6	25
23	0.2	0.2	8×10^{-4}	0.6	50	47	0.4	0.1	10^{-3}	0.6	50
24	0.2	0.2	8×10^{-4}	0.6	75	48	0.4	0.1	10^{-3}	0.6	75

Table C.3: Parameters of the first 48 experiments of the problem with 135 flights.

Ex.	$fneg$	$fpos$	$fraction$	$flim$	N	**Ex.**	$fneg$	$fpos$	$fraction$	$flim$	N
49	0.4	0.2	8×10^{-4}	0.3	0	73	0.8	0.1	10^{-3}	0.3	0
50	0.4	0.2	8×10^{-4}	0.3	25	74	0.8	0.1	10^{-3}	0.3	25
51	0.4	0.2	8×10^{-4}	0.3	50	75	0.8	0.1	10^{-3}	0.3	50
52	0.4	0.2	8×10^{-4}	0.3	75	76	0.8	0.1	10^{-3}	0.3	75
53	0.4	0.2	8×10^{-4}	0.6	0	77	0.8	0.1	10^{-3}	0.6	0
54	0.4	0.2	8×10^{-4}	0.6	25	78	0.8	0.1	10^{-3}	0.6	25
55	0.4	0.2	8×10^{-4}	0.6	50	79	0.8	0.1	10^{-3}	0.6	50
56	0.4	0.2	8×10^{-4}	0.6	75	80	0.8	0.1	10^{-3}	0.6	75
57	0.4	0.2	10^{-3}	0.3	0	81	0.8	0.2	8×10^{-4}	0.3	0
58	0.4	0.2	10^{-3}	0.3	25	82	0.8	0.2	8×10^{-4}	0.3	25
59	0.4	0.2	10^{-3}	0.3	50	83	0.8	0.2	8×10^{-4}	0.3	50
60	0.4	0.2	10^{-3}	0.3	75	84	0.8	0.2	8×10^{-4}	0.3	75
61	0.4	0.2	10^{-3}	0.6	0	85	0.8	0.2	8×10^{-4}	0.6	0
62	0.4	0.2	10^{-3}	0.6	25	86	0.8	0.2	8×10^{-4}	0.6	25
63	0.4	0.2	10^{-3}	0.6	50	87	0.8	0.2	8×10^{-4}	0.6	50
64	0.4	0.2	10^{-3}	0.6	75	88	0.8	0.2	8×10^{-4}	0.6	75
65	0.8	0.1	8×10^{-4}	0.3	0	89	0.8	0.2	10^{-3}	0.3	0
66	0.8	0.1	8×10^{-4}	0.3	25	90	0.8	0.2	10^{-3}	0.3	25
67	0.8	0.1	8×10^{-4}	0.3	50	91	0.8	0.2	10^{-3}	0.3	50
68	0.8	0.1	8×10^{-4}	0.3	75	92	0.8	0.2	10^{-3}	0.3	75
69	0.8	0.1	8×10^{-4}	0.6	0	93	0.8	0.2	10^{-3}	0.6	0
70	0.8	0.1	8×10^{-4}	0.6	25	94	0.8	0.2	10^{-3}	0.6	25
71	0.8	0.1	8×10^{-4}	0.6	50	95	0.8	0.2	10^{-3}	0.6	50
72	0.8	0.1	8×10^{-4}	0.6	75	96	0.8	0.2	10^{-3}	0.6	75

Table C.4: Parameters of the last 48 experiments of the problem with 135 flights.

Expr.	Dur.(sec.)	Expr.	Dur.(sec.)	Expr.	Dur.(sec.)
1	2.8111	33	2.7246	65	3.9029
2	14.2766	34	10.0923	66	13.5635
3	35.1082	35	22.0995	67	25.5668
4	57.5636	36	38.3283	68	50.4395
5	2.1835	37	2.8766	69	3.8639
6	14.2343	38	10.0681	70	14.4574
7	34.9832	39	22.2245	71	24.5376
8	57.7464	40	38.3365	72	50.4094
9	2.7859	41	2.8705	73	3.8226
10	14.2779	42	9.9959	74	14.2411
11	34.9496	43	22.1065	75	22.1805
12	57.6683	44	38.5269	76	38.5072
13	1.9340	45	2.8312	77	3.798
14	14.2321	46	10.0752	78	15.2572
15	35.0768	47	22.2045	79	22.1462
16	57.5987	48	38.2768	80	38.3806
17	2.6264	49	2.9267	81	3.9698
18	9.9816	50	10.1334	82	15.7151
19	32.3951	51	22.1987	83	21.7783
20	56.3074	52	38.4473	84	49.6063
21	2.0086	53	2.9201	85	3.6396
22	9.1953	54	10.9576	86	14.3437
23	32.4097	55	24.1782	87	21.8142
24	56.3160	56	41.8470	88	46.8633
25	2.5982	57	2.9546	89	3.8873
26	10.0796	58	10.0725	90	13.3905
27	30.4427	59	22.2389	91	28.6989
28	56.1995	60	42.4156	92	37.5725
29	2.0139	61	2.8656	93	3.8348
30	9.1941	62	10.9188	94	14.3181
31	30.3084	63	24.1562	95	21.6719
32	56.2212	64	38.3295	96	37.5524

Table C.5: Average CPU times for an iteration of the MLSP of the problem with 135 flights with exact approach.

Expr.	Dur.(sec.)	Expr.	Dur.(sec.)	Expr.	Dur.(sec.)
1	18.8067	33	14.8608	65	7.8868
2	23.5595	34	20.8847	66	7.4228
3	80.8032	35	53.9962	67	31.7985
4	103.1261	36	107.4165	68	57.2300
5	18.9888	37	16.1177	69	9.2281
6	26.1673	38	21.8188	70	7.7960
7	75.6085	39	43.2035	71	33.2951
8	106.0957	40	83.6464	72	59.3855
9	19.8812	41	16.8170	73	10.5955
10	39.0526	42	25.3543	74	11.1179
11	75.4415	43	49.3395	75	33.9240
12	103.5241	44	100.1905	76	53.4708
13	19.4990	45	16.8145	77	9.9758
14	25.5440	46	19.3994	78	9.8241
15	74.3716	47	47.1341	79	32.6268
16	106.0361	48	97.7682	80	52.5419
17	17.5215	49	14.2593	81	7.9605
18	32.2207	50	18.8840	82	7.8956
19	61.5537	51	39.1106	83	24.5742
20	111.0655	52	88.3243	84	64.5102
21	19.3048	53	14.0829	85	8.7389
22	26.1609	54	20.7240	86	8.8754
23	57.0818	55	37.2836	87	28.5209
24	110.8236	56	80.7565	88	56.8104
25	19.4797	57	15.0449	89	9.1821
26	19.2593	58	23.3089	90	9.3550
27	65.1964	59	41.6612	91	28.2162
28	120.7756	60	88.1923	92	51.6704
29	20.2400	61	15.5369	93	9.7226
30	26.6437	62	22.1568	94	12.3865
31	63.0516	63	38.5867	95	28.7575
32	117.4522	64	73.6799	96	48.4023

Table C.6: Average CPU times for an iteration of the MLSP of the problem with 135 flights with hybrid approach.

Iteration	Dur. (sec.)	Method
1	0.282	Approximate
2-8	-	Buffer
9	0.235	Approximate
10-11	-	Buffer
12	0.406	Approximate
13	5.344	App + Exact
14-26	-	Buffer
27	0.375	Approximate
28	-	Buffer
29	10.416	App + Exact
30	10.437	App + Exact

Table C.7: The duration of each iteration of the problem with 135 flights in Experiment 1 (exact method).

Iteration	Comp. Time (sec.)	Method	Last Node *
1	0.25	Approximate	-
2-8	-	Buffer	-
9	0.266	Approximate	-
10-11	-	Buffer	-
12	0.375	Approximate	-
13	0.907	App + Hybrid	8
14	1.219	App + Hybrid	17
15	8.265	App + Hybrid	167
16	11.234	App + Hybrid	167
17	-	Buffer	-
18	1.094	App + Hybrid	15
19	11.297	App + Hybrid	186
20	0.406	Approximate	-
21-22	-	Buffer	-
23	12.124	App + Hybrid	191
24-25	-	Buffer	-
26	49.094	App + Hybrid	226
27	-	Buffer	-
28	49.656	App + Hybrid	226
29	49.156	App + Hybrid	227
30	50.125	App + Hybrid	227
31	50.454	App + Hybrid	227
32	141.797	App + Hybrid	264
33	164.094	App + Hybrid	271 (sink)

* : The topological order of the last node to which the exact rule is applied.

Table C.8: The duration of each iteration of the problem with 135 flights in Experiment 1 (hybrid method).

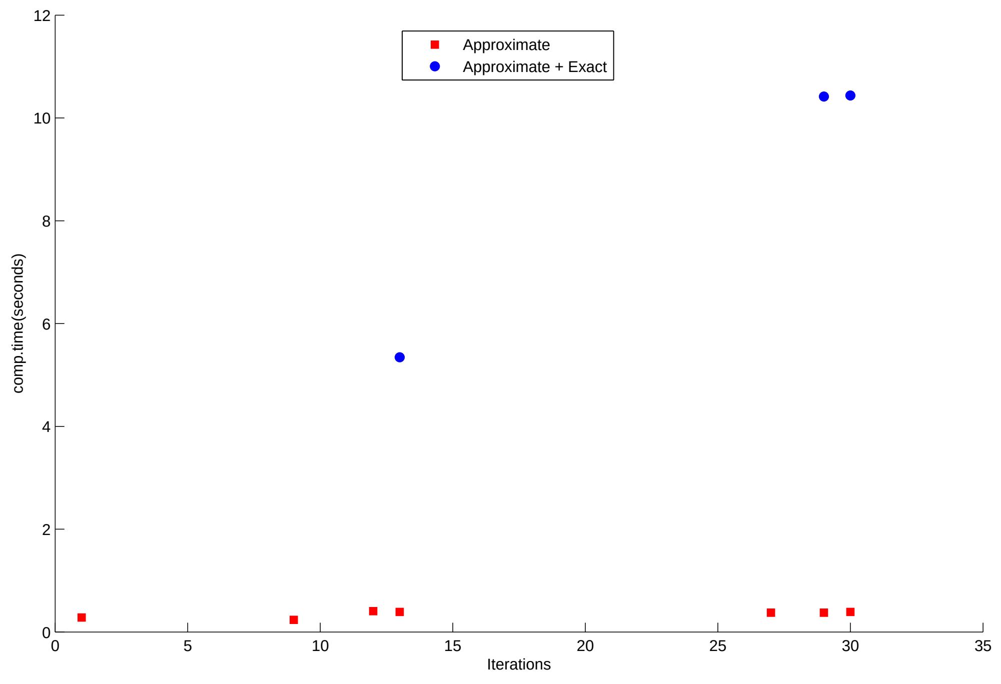

Figure C.1: Results associated with Table C.7.

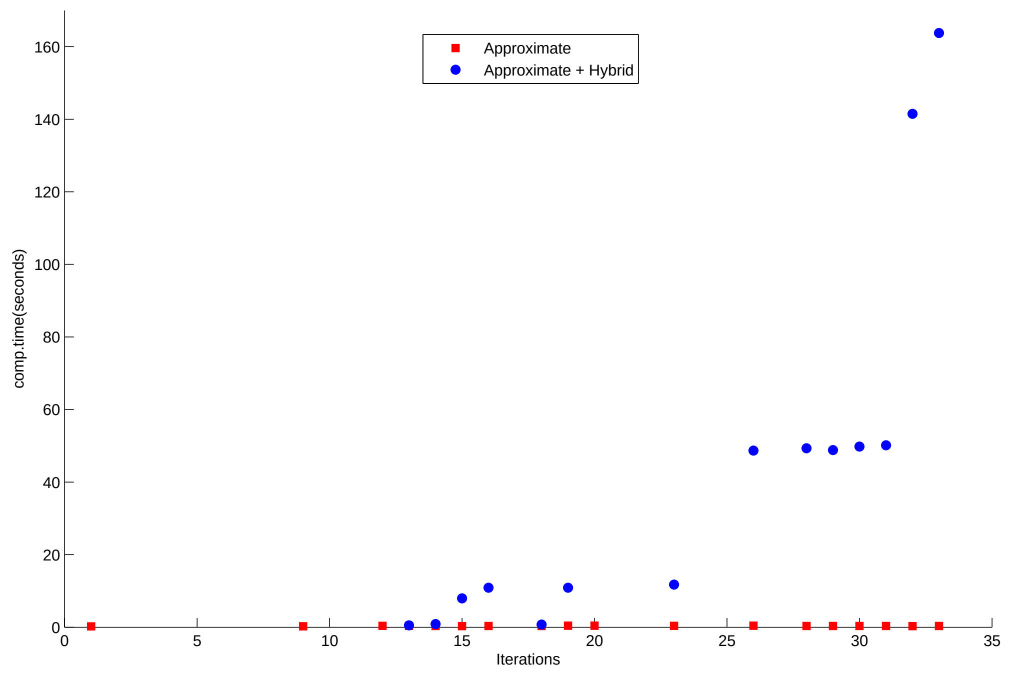

Figure C.2: Results associated with Table C.8.

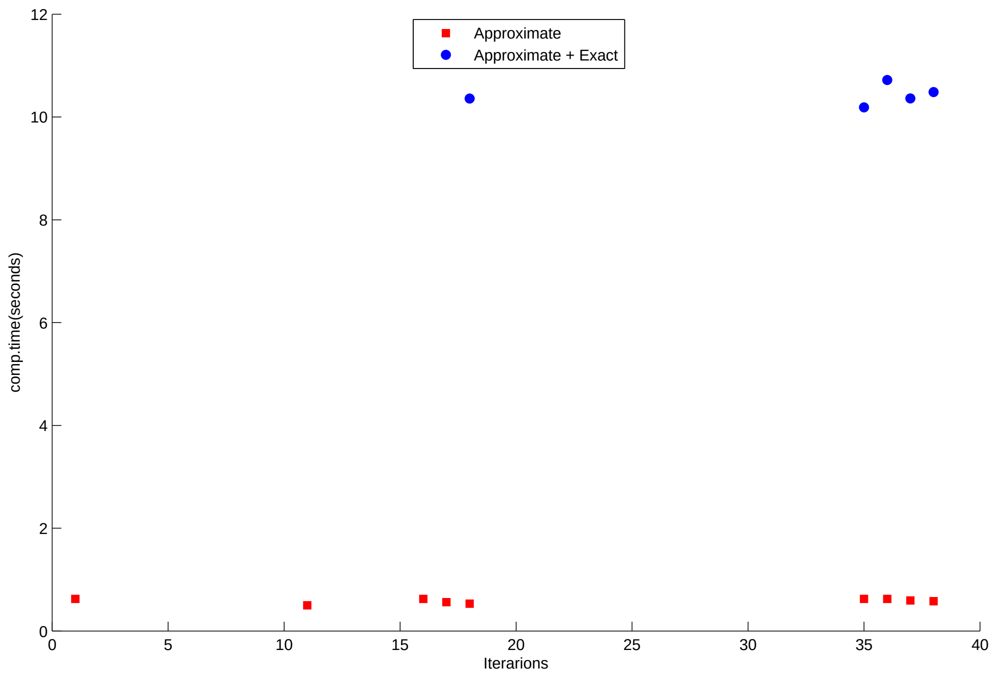

Figure C.3: Results associated with Experiment 89 (exact method).

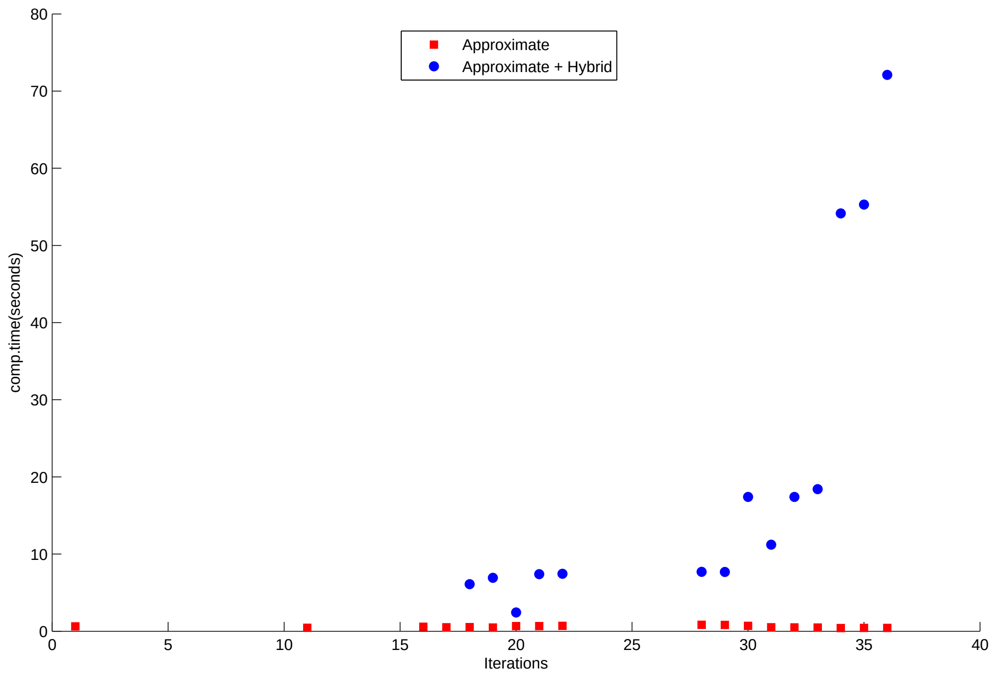

Figure C.4: Results associated with Experiment 89 (hybrid method).