DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

LARGE SCALE OPTIMIZATION APPLICATIONS ON THE AIRCRAFT AND AIRCREW RECOVERY PROBLEMS AND THEIR INTEGRATION

by
Nazan ZEYBEKCAN

October, 2011
İZMİR

LARGE SCALE OPTIMIZATION APPLICATIONS ON THE AIRCRAFT AND AIRCREW RECOVERY PROBLEMS AND THEIR INTEGRATION

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Industrial Engineering, Industrial Engineering Program

by
Nazan ZEYBEKCAN

October, 2011
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"LARGE SCALE OPTIMIZATION APPLICATIONS ON THE AIRCRAFT AND AIRCREW RECOVERY PROBLEMS AND THEIR INTEGRATION"** completed by **NAZAN ZEYBEKCAN** under supervision of **PROF. DR. HASAN ESKİ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Hasan ESKİ
Supervisor

Prof. Dr. Tatyana YAKHNO
Thesis Committee Member

Assoc.Prof.Dr. Şeyda TOPALOĞLU
Thesis Committee Member

Prof. Dr. Erol Rıfat SAYIN
Examining Committee Member

Asst.Prof.Dr. Bilge BİLGEN
Examining Committee Member

Prof.Dr. Mustafa SABUNCU
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENT

I would like to express my heart-felt thanks first and foremost to my advisor, Prof. Dr. İrem Özkarahan, for her guidance, cooperation, encouragement and advice throughout my PhD studies. This dissertation could not have been developed and written without her, who has always been helpful, dedicating and encouraging advisor and has offered invaluable assistance, support and guidance. I would also like to express my gratitude to her for never giving up being my advisor, although she transferred to USA five years ago and for all the sacrificies she made in order to come to Turkey for my all thesis examinations, which take place two times a year, during five years despite the long distance. However, I am truly sorry that she wouldn't be present at my final and the most important thesis examination, since she can not leave her duty in USA and come to Turkey on the examination date.

I would like to thank to my second advisor, Prof. Dr. Hasan Eski who provided continous support and interest throughout the progress of this dissertation. I would also like to thank to my committee members Prof. Dr. Tatyana Yakhno and Assoc. Prof. Dr. Şeyda Topaloğlu for their helpful comments and suggestions during the progress of this dissertation. In addition, I would like to thank to all the professors and instructors in the Department of Industrial Engineering at Dokuz Eylul University for them equipping me with their knowledge and academic skills.

I would like to thank all my colleagues, who are working in MIS Department at Çimentaş Türk A.Ş., Gökhan Yurtesen, İzak Ersönmez, Erdal Öcal, Selda Salman, Fırat Altınyelken, Hüseyin Güzel and Ulaş Polat, for their continuous support, encouragement, understanding and endless patience during my PhD study. I am very fortunate to work with them. I am also indebted to PhD candidate Mustafa Erhan Bilman, as one of my good friends, who has always helped me and supported me during my research.

Last but not the least; I am deeply thankful to my family. There is no way I can adequately express my gratitude to my family who have always provided me with endless love, security and incredible support. I am deeply indebted to my parents,

Hatice and Mehmet Zeybekcan, for their confidence, encouragement, and support throughout all these years of my education.

Nazan ZEYBEKCAN

LARGE SCALE OPTIMIZATION APPLICATIONS ON THE AIRCRAFT AND AIRCREW RECOVERY PROBLEMS AND THEIR INTEGRATION ABSTRACT

The air transport industry has always been an interesting research area and the researchers in the operations research field have been developing solution algorithms and optimization tools for airlines since 1950s. The airlines are now capable of managing their key planning process and schedules using these optimization tools.

Today, the airlines spend a lot of effort on recovering their schedules in case of unpredictable events, rather than managing the planning process. In a typical day, airlines face several problems causing disruptions in the planned schedules and they try to find a minimal cost reschedules taking into account the available resources and satisfying all the operational and safety rules. These recovery problems are challenging and subject to further improvements. Therefore, this dissertation focuses on developing new efficient models and solution algorithms for the aircraft and crew recovery problems and their integration.

In this dissertation several models for individual aircraft and crew recovery problems are developed using two different techniques; multi-commodity network flow and constraint programming techniques.

However, while developing recovered schedules, both crew and aircraft availabilities should be considered together. This dissertation also proposes two new solution algorithms, which solve aircraft and crew recovery problems sequentially and in iterative manner, taking into account the dependency of two problems, representing the correlation between them without needing a very large model that would result integrating the aircrew and aircraft models. These solution algorithms are developed using two different techniques; multi-commodity network flow and constraint programming techniques.

All the proposed models and solution algorithms developed in this dissertation are applied to real data obtained from the daily domestic flight schedule of major airlines

operating in Turkey. The results, computational times and quality of the solution techniques are also evaluated. The results show that, the proposed solution algorithms generate recovered schedules in real time and constraint programming embedded solution approaches produces better solutions.

Keywords: Airline operations, disruption management, aircraft recovery, crew recovery, multi-commodity network flow model, constraint programming.

UÇAK VE UÇUŞ EKİBİ YENİDEN ÇİZELGELEME PROBLEMLERİNDE VE BU PROBLEMLERİN ENTEGRASYONUNDA GENİŞ ÖLÇEKLİ OPTİMİZASYON UYGULAMALARI

ÖZ

Havayolları endüstrisi her zaman ilgi çekici bir araştırma alanı olmuştur ve yöneylem araştırmacıları 1950 lerden beri, havayolları için çeşitli çözüm algoritmları ve optimizasyon araçları geliştirmektedirler. Şimdilerde havayolları bu geliştirilmiş olana optimisasyon araçlarını kullanarak kendi ana planlama süreçlerini rahatlıkla yönetebilmektedirler.

Günümüzde, havayolları planlama süreçlerini yönetmektense beklenmeyen durumlar karşısında bozulan çizelgelerini düzeltmek için çaba harcamaktadırlar. Sıradan bir gün içinde, havayolları planlı çizelgelerininin bozulmasına sebep olan bir çok problemle karşılaşırlar. Böyle bir durumda da tüm operasyonel ve güvenlik kurallarına uyacak şekilde ellerindeki uygun kaynakları kullanarak, minimum maliyetli yeni çizelgeleri bulmaya çalışırlar. Bu tür yeniden çizelgeleme problemleri zorlu problemlerdir ve bu problemler için hala yeni araştırma alanları açıktır. Bu tez çalışmasında, uçak ve uçuş ekibi yeniden çizelgeleme problemleri ve bu problemlerin entegrasyonu için yeni ve etkin modeller ve çözüm algoritmaları geliştimek üzerine yoğunlaşılmıştır.

Bu tez çalışmasında, öncelikle hem uçak, hem de uçuş ekibi yeniden çizelgeleme problemleri için ayrı ayrı değişik modeller geliştirilmiştir. Bu modeller geliştirilirken temel olarak iki farklı yöntem kullanılmıştır: Çoklu ürün ağ akışı ve kısıt programlama.

Ancak, yeniden çizelgeleme problemleri çözülürken, hem uçuş ekibi hem de uçakların uygunluğu beraberce dikkate alınmalıdır. Bu tez çalışmasında uçak ve uçuş ekibi için ayrı ayrı geliştirilen modellerden başka, her ikisinin beraberce çözülebilmesi için de iki yeni çözüm algoritması geliştirilmiştir. Bu çözüm algoritmaları, uçak ve uçuş ekibi yeniden çizelgeleme problemlerini tek ve büyük bir modelde birleştirme ihtiyacı duymadan, ancak bu iki problemin biribiriyle olan

bağını ve aralarındaki ilişkiyi dikkate alarak, iteratif bir şekilde çözer. Bu çözüm algotimaları yine iki farlı yöntem kullanılarak geliştilmiştir: Çoklu ürün ağ akışı ve kısıt programlama.

Bu tez çalışmasında geliştirilen tüm model ve çözüm algoritmaları Türkiye'nin büyük havayollarından alınan gerçek verilerle test edilmiştir. Elde edilen sonuçlar, çözüm zamanları ve çözüm tekniklerinin kalitesi değerlendirilmiştir. Sonuçlar göstermektedir ki; önerilen çözüm algoritmları gerçek zamanda yeniden çizelgeleme yapabilmektedir ve kısıt programlama içeren çözüm yöntemi daha iyi sonuçlar üretmektedir.

Anahtar sözcükler : Havayolları operasyonları, düzensiz olaylar yönetimi, uçak yeniden çizelgeleme, uçuş ekibi yeniden çizelgeleme, çoklu ürün ağ akışı, kısıt programlama.

CONTENTS

Page

THESIS EXAMINATION RESULT FORM ii
ACKNOWLEDGEMENTS iii
ABSTRACT v
ÖZ vii

CHAPTER ONE – INTRODUCTION 1

1.1 Airline Planning Problems 2
1.1.1 Schedule Development 4
1.1.2 Fleet Assignment Problem 4
1.1.3 Aircraft Routing Problem 6
1.1.4 Crew Scheduling Problem 7
1.1.5 Crew Rostering 10
1.2 Irregular Airline Operations 12
1.2.1 The Sources of Disruptions 12
1.2.1.1 Airline Resource Shortages 13
1.2.1.2 Airport and Airspace Capacity Shortages 13
1.2.2 Recovery 14
1.2.3 Aircraft Recovery Problems 16
1.2.4 Aircrew Recovery Problems 17
1.2.4.1 Crew Problems in Case of Disruptions 19
1.3 Motivation and Contributions 22
1.4 Organization of Dissertation 25

CHAPTER TWO - LITERATURE REVIEW ON AIRLINE PLANNING AND OPERATIONS 27

2.1 The Literature Review on Airline Planning Problems 27

2.1.1 The Literature Review on Schedule Development Problems.....27
2.1.1.1 Impacts and Challenges of Schedule Development Problem 28
2.1.2 The Literature Review on Fleet Assignment Problems.....29
2.1.2.1 Integrated Fleet Assignment and Aircraft Routing Problems.....32
2.1.2.2 Impacts and Challenges of Fleet Assignment Problem 33
2.1.3 The Literature Review on Aircraft Routing Problems 35
2.1.3.1 Impacts and Challenges of Aircraft Routing Problem.....36
2.1.4 The Literature Review on Crew Scheduling Problems 37
2.1.4.1 Integrated Aircraft Routing and Crew Scheduling Problems 40
2.1.4.2 Impacts and Challenges of Airline Crew Scheduling Problem.....42
2.1.5 The Literature Review on Crew Rostering Problems.....43
2.1.5.1 Impacts and Challenges of Crew Rostering Problem 44
2.2 The Literature Review on Airline Recovery Problems 45
2.2.1 The Literature Review on Aircraft Recovery Problems 46
2.2.2 The Literature Review on Crew Recovery Problems 56
2.2.3 The Literature Review on Integrated Recovery Problems 64
2.2.4 Impacts and Challenges of Airline Recovery Problem.....65

CHAPTER THREE - OVERVIEW OF THE METHODOLOGY EMPLOYED IN THE DEVELOPED MODELS.....68

3.1. Brief Overview of Multi-Commodity Network Flow Problems 68
3.1.1 Mathematical Formulation of Multi-Commodity Network Flow Models... 69
3.1.2 Application Areas of Multi-Commodity Network Flow Models 71
3.1.3 The Multi-Commodity Network Models for Airline Optimization Problems 73
3.1.3.1 Time-Space Network in Airline Optimization Problems.....74
3.1.3.2 Connection Network in Airline Optimization Problems 78
3.2. Brief Overview of Constraint Programming 82
3.2.1 Definition of Constraint Programming.....82
3.2.2 Constraint Satisfaction Problem 84

3.2.3 Constraint Propagation and Domain Reduction 86
3.2.4 Constraint Programming Algorithms for Optimization.....88
3.2.5 Applications Areas of Constraint Programming.....89
3.2.6 Advantages of Constraint Programming 91
3.2.7 Limitations of Constraint Programming.....91
3.2.8 Constraint Programming vs. Mathematical Programming 93

CHAPTER FOUR - AIRCRAFT RECOVERY PROBLEMS 94

4.1 Model A1- Cancellation Model with Single Aircraft Unavailability Using Multi-Commodity Network Flow Model 97
4.1.1 Notation and Mathematical Formulation for Model A1 97
4.1.2 Numerical Example for Model A1 100
4.2 Model A2 - Cancellation and Delay Model with Multiple Aircraft Unavailability using Multi-Commodity Network Flow Models 102
4.2.1 The Definition of Recovery Horizon in Case of Multiple Aircraft Shortages 104
4.2.2 The Notation and Mathematical Formulation for Model A2.....105
4.2.3 The Numerical Example for Model A2 108
4.3 Model A3 - Delay and Cancellation Model with Multiple Aircraft Unavailability Including Crew Considerations using Multi-Commodity Network Flow Problem.....111
4.3.1 The Notation and Mathematical Formulation of Model A3 113
4.3.2 Numerical Example for Model A3 115
4.4 Model A4 - Aircraft Recovery Problem using Constraint Programming Embedded Solution Approach.....118
4.4.1 First Stage – Constraint Programming Problem for Model A4 121
4.4.2 Second Stage– Integer Programming Problem for Model A4 123
4.4.3 Numerical Example for Model A4 126
4.5 Evaluation and Comparison of the Developed Aircraft Recovery Models...128

CHAPTER FIVE - CREW RECOVERY PROBLEMS.....133

5.1 Model C1- The Crew Recovery Problem as Multi-Commodity Network Flow Problem.....134
5.1.1 The Notation and Mathematical Formulation of Model C1 135
5.1.2 Numerical Example for Model C1 138
5.2 Model C2- Crew Recovery Problem with Constraint Programming Embedded Solution Approach.....141
5.2.1 First Stage – Constraint Programming Problem for Model C2 143
5.2.2 Second Stage – Integer Programming Problem for Model C2 146
5.2.3 Numerical Example for Model C2 149
5.3 Evaluation and Comparison of Developed Crew Recovery Models.....151

CHAPTER SIX- ITERATIVE SOLUTION APPROACH FOR INTEGRATED AIRCRAFT AND CREW RECOVERY PROBLEMS USING MULTI-COMMODITY NETWORK FLOW MODELS 154

6.1 Solution Algorithm S1- Sequential and Iterative Solution Approach for Integrated Problem using Multi-Commodity Network Flow Models 156
6.2 The Aircraft Recovery Problem for the Solution Algorithm S1 159
6.3 The Crew Recovery Problem for the Solution Algorithm S1 162
6.4 Numerical Example for Solution Algorithm S1.....165
6.5 Evaluation of the Proposed Solution Algorithm S1 167

CHAPTER SEVEN - ITERATIVE SOLUTION APPROACH FOR INTEGRATED AIRCAFT AND CREW RECOVERY PROBLEMS USING CONSTRAINT PROGRAMMING.....170

7.1 Solution Algorithm S2 - Sequential and Iterative Solution Approach for Integrated Problem using Constraint Programming Embedded Solution Approach 173
7.2 Constraint Programming Model for Generating Crew Pairings (Step 2.1)...179

7.3 Constraint Programming Model for Generating Aircraft Routes (Step 2.2) .181
7.4 Integer Programming Model for Crew Recovery Problem (Step 3.2) 182
7.5 Integer Programming Model for Aircraft Recovery Problem (Step 6.2) 183
7.6 Numerical Example for Solution Algorithm S2 185
7.7 Evaluation of the Solution Algorithm S2 192

CHAPTER EIGHT - CONCLUSION.....196

8.1. Summary and Conclusion 196
8.1.1 Summary of the Individual Aircraft and Crew Recovery Problems.....197
8.1.2 Summary of the Integrated Aircraft and Crew Recovery Problems.....200
8.1.3 Contributions 203
8.2 Directions for Future Research.....204

REFERENCES.....206

APPENDICES 219
Appendix A 220
Appendix B.....230
Appendix C.....239
Appendix D 242
Appendix E.....247
Appendix F 253
Appendix G 265
Appendix H 269
Appendix I.....283
Appendix J.....285

CHAPTER ONE
INTRODUCTION

The air transport industry has always been an interesting research area since the first airline started to be operated at the beginning of the 1900s. Since then, the air transport industry has been very important sector for global economy in terms of annual revenues of the airlines, the number of passengers carried over a year, the number of job opportunities provided by the airlines. Furthermore, the airline industry has grown even strongly over the last 60 years especially since 1950s. During this period operations research has been playing an important role in the airline industry.

The airlines have been applying operations research techniques to their planning and operations problems since 1950s. The researchers have developed large number of mathematical techniques for airlines in order to increase their profitability. Even recently, the demand for operations research techniques from the airlines and the advances in new technologies and the speed of computers have motivated the rapid development of new algorithms and solution approaches for airline planning and operations problems. However, scheduling and planning problems still remain very complex and challenging in the airline industry. Thus, the problems in the airline industry still continue to take significant attention of the researchers in the operations research field.

The major part of the airline industry literature has focused on the airline planning problems, which are schedule development, fleet assignment, aircraft routing, crew scheduling and crew rostering problems. These problems attempt to find optimal schedules for airlines and they are solved in advance, before the actual transportation occurs.

Another important and attractive research area in the airline industry literature is the airline operations problems. The airline operations problems are solved as the plan is being implemented. The “optimal” schedules obtained from the solution of the airline planning problems can not be operated as planned in practice. Because of

the time lag between planning and operations, circumstances may have changed at the time of operations, so that the planned schedules are not no longer feasible or desirable. The irregularities in the original schedule may occur due to unpredictable weather conditions, unscheduled maintenance delays, gate delays etc. In order to deal with these disruptions, real-time algorithms are developed to re-optimize the schedule after such irregularities occur. With these real-time algorithms, the airlines try to find a minimal cost reschedules taking into account the available resources and satisfying all the operational and safety rules.

During the last decades the airlines have used several optimization tools developed in order to optimize their planning process. Currently, most airlines are capable of performing their key planning process, including revenue management and aircraft and crew scheduling, aided with sophisticated software tools. Although new contributions and further improvements are still possible for planning process, the researchers and the airlines have focused on finding real-time solutions and optimized re-schedules during their daily operations. They now try to find a way to recover their planned schedules as soon as possible when they face with a disruption.

1.1 Airline Planning Problems

An airline must make many decisions before operating a flight. It must decide when and where to fly, with which plane, with which crew, with which flight attendants, and so on. Managing these kind of decisions involves complex scheduling problems including a network of flights, differing aircraft types, gate, airport slot and air traffic control restrictions, maintenance requirements, crew work rules, and competitive, dynamic environments in which passenger demands are uncertain and pricing strategies are complex. It is very difficult to develop a single optimization model to address this complex design task in its entirety. Because of the problem's unmanageable size and complexity, the overall problem is decomposed into a sequence of smaller, more manageable subproblems and then these subproblems are solved sequentially. The subproblems are:

- Schedule Development
- Fleet Assignment
- Aircraft Routing
- Crew Scheduling
- Crew Rostering

In order to construct feasible airline problems, these problems should be solved in order, constraining the solutions to subsequent problems based on the solutions to preceding problems. Thus, an airline first determines when and where it will fly. It then determines which flights will be flown by which plane. A group of crew trips is found which ensures that exactly one crew flies every flight. Individual pilots and crew attendants are then assigned to each trip.

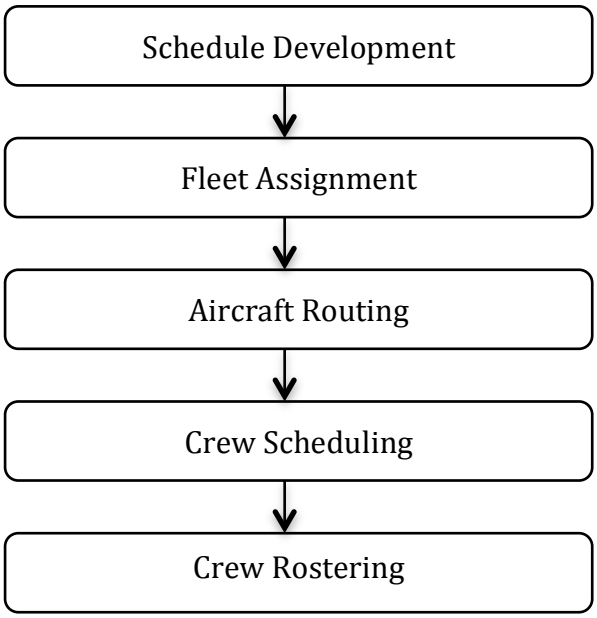

Figure 1.1 Steps in airline planning

Although these subproblems are smaller and simpler than the overall problem, they are still large-scale and complex. The researchers have been developing models and algorithms to solve them for decades.

1.1.1 Schedule Development

The flight schedule specifies the flight legs to be flown and the departure time of each flight leg. A flight leg consists of an origin station, a destination station, a departure time, and an arrival time. Flight schedule largely defines the competitive position of an airline and directly affects the airline profitability. Designing a profit maximizing flight schedule is extremely complex. It affects and is affected by all aircraft and crew scheduling decisions of the airline.

In the most major airlines, the hub-and-spoke network structure is used. A hub-and-spoke network is an airline flight schedule with a large percentage of the legs going into or out of a small subset of stations. Major airports where the activity is high are called hubs and the low activity airports, called spokes are mostly served from hubs. Hub-and-spoke networks typically allow many more passenger connections than point-to-point networks. Smaller airlines use a point-to-point flight network structure. Such a network is characterized by direct flights from airports to airports; however the number of origin destination pairs is low.

1.1.2 Fleet Assignment Problem

The fleet assignment problem (FAP) deals with assigning aircraft types, each having a different capacity, to the scheduled flights, based on equipment capabilities and availabilities, operational costs, and potential revenues. An airlines fleeting decision highly impacts their revenues: Assigning a smaller aircraft than needed on a flight will result in spilled (i.e., lost) customers due to insufficient capacity; assigning a larger aircraft will result in spoiled (i.e., unsold) seats, and presumably higher operational costs. Thus, FAP constitutes an essential component of an airlines overall scheduling process. However, due to the large number of flights scheduled each day, which can easily reach thousands for a major airline, and the dependency of the FAP on other airline processes such as schedule design, crew scheduling, aircraft routing, maintenance planning, and revenue management, solving the FAP has always been a challenging task for the airlines.

Large domestic carriers usually have more than one type of aircraft. A fleet is a set of planes of the same type. For the purposes of fleet assignment, fleets may differ in the number of seats as well as the type of plane. After the flight schedule has been found, each leg is assigned to a fleet with a specific passenger capacity.

The FAPs are generally modeled as an integer multi-commodity network flow model. The commodities are fleets and each flight leg in the schedule must have exactly one assigned fleet. Planes, in a fleet correspond to circulations, i.e. a plane landing at a station must depart from the same station. The mostly used underlying network is a time-line network. The nodes correspond to the times and locations of flight departures and arrivals. There are two types of arcs, flight arcs and ground arcs.

Flight arcs represent the schedule's flight legs, with arrival times adjusted to include the minimum amount of time aircraft must remain on the ground. This time is needed for tasks such as disembarking and embarking passengers, unloading and loading baggage, and refueling.

A ground arc represents a connection between two consecutive flights. A FAP solution obviously can not use more planes than there exist in a fleet. These constraints are called plane count constraints. They are modeled by introducing ground arcs and associated variables. Each fleet has its own set of ground arc variables. A ground arc variable counts the number of planes in the fleet on the ground in the time interval defined by the arc. The value of such a variable is called the ground arc value. For each fleet the flow conservation constraints state that the number of planes on the ground plus the number of planes arriving must be equal to the number of planes on the ground in the next time interval plus the number of planes departing. The number of planes in a fleet is the sum of all the ground arc values of those arcs at a specified point in time, e.g. midnight, plus those aircraft that are in the air at midnight.

The objective is to flow each commodity (aircraft type) through the network and with minimum cost, such that each flight leg is assigned to exactly one aircraft type. Fleeting costs are comprised of:

- Operating costs: Specified for each flight leg-aircraft type pair, representing the cost of flying that flight leg with that aircraft type.
- Spill costs: Measuring the revenue lost when passenger demand for a flight leg exceeds the assigned aircraft's seating capacity.

For each feasible assignment of a fleet type to a flight leg, the decision of whether or not to make that assignment is represented by a binary variable.

1.1.3 Aircraft Routing Problem

With schedule design and fleet assignment decisions made, the flight network decomposes into subnetworks, each one associated with aircraft of a single type. The assignment of individual aircraft to flight legs in a subnetwork occurs in the aircraft routing step. The goal is to determine routings, or rotations, for each aircraft in a fleet. An aircraft route is a sequence of flight legs that are operated by the same aircraft. The routing of a fleet typically starts and ends at the same station. A solution to the aircraft routing problem provides the routing for each aircraft consisting of a sequence of flight legs and ground connections. A feasible routing must satisfy the plane count constraints. The aircraft routing problems in the literature have generally considered integrated with fleet assignment or crew scheduling problems.

Some constraints considered in aircraft routing problems are as follows:

- The number of available aircraft for each type
- The restrictions on certain aircraft types at certain times and stations
- The minimum required turnaround times between two flight legs
- The limits on daily service at certain stations
- Flight departure time windows

Another importing airline planning problem is aircraft maintenance problem which is considered as a subproblem of aircraft routing problem. Each aircraft's routing visits maintenance stations at regular intervals. The primary objective of the aircraft-maintenance routing problem is to find a feasible solution, one ensuring sufficient maintenance opportunities for each aircraft.

The aircraft maintenance routing problem can be modeled as a network circulation problem with side constraints. The decision variables correspond to sequences (strings) of flight legs. The sequence begins and ends at a maintenance station and it satisfies the rules governing the maximum time between maintenance. If a string is included in the solution, a single aircraft flies each flight in the sequence and then undergoes maintenance.

The aircraft maintenance routing problem has always been challenging for researchers. Instead of obtaining optimized solutions they have attempt to find feasible solutions.

1.1.4 Crew Scheduling Problem

The input to the airline crew scheduling problem consists of the schedule of a fleet, which covers a week of operations and the aircraft routing of the fleet.

Crew scheduling problem is to find a minimum cost, multiple-day work schedules, called crew *pairings* that meet the crew requirements for each flight leg of a given schedule. Crew pairings are sequences of flight legs that will eventually be assigned to individual crewmembers. There are many aviation and union regulations governing what constitutes legal pairings. Standard rules include:

- A limit on the number of hours of flying time in duty,
- A limit on the total length of a duty,
- A limit on the number of duties in a pairing,
- Upper and lower bounds on the rest time between two duties,
- A limit on the number of plane changes in a duty.

Even with these limitations, the number of feasible pairings measures in the billions for major U.S. airlines. The cost structure of pairings adds further complexity, with cost typically represented as a nonlinear function of flying time, total elapsed work time, and total time away from base.

The crew scheduling problem continues to be challenging to solve due to the following reasons:

1. In the airline industry crews represent the airlines' second highest operating cost after fuel, so even slight improvements in their utilization can translate into significant savings. Because the large savings are possible while using aircrews more efficiently, many airlines have tried to develop optimization methods to solve them efficiently. For example Graves, McBride, Gershkoff, Anderson and Mahidhara (1993) developed a new crew scheduling optimization system for United Airlines and obtained annually $16.000.000 of savings in crew scheduling costs. Moreover Butchers et al. (2001) applied an effective crew schedule to Air New Zealand's national and international operations and saved $15.655.000 per year while satisfying crew members' preferences. A good crew schedule affects both an airline's profit and its flight safety.

2. Another challenge in airline crew scheduling problems is their complex structure and large size. Airline crew schedule construction is very difficult because of the size of the problem and the complexity of collective agreements and government rules that must be respected. One such rule is the 8-in-24 rule. One version of this rule requires that not more than eight hours of flying are assigned during any twenty-four hour period. The number of pairings varies from 200,000 for small fleets, to about a billion for medium size fleets and to billions for large fleets. For example, major US airlines operate up to 5000 flights per a day. 1000 daily flight legs may produce 1012 to 1015 pairings, which are so many that it is very difficult to generate all of them manually. With good scheduling algorithms these big problems can be solved in short times.

3. Crew costs are difficult to be computed since they depend on complex crew pay guarantees and are highly nonlinear.

Crew Scheduling Problems are generally formulated as set partitioning type problems in the literature. The classical crew scheduling problem's mathematical formulation is as follows:

$$Minimize \sum_{j=1}^{n} c_j X_j \qquad (1.1)$$

subject to

$$\sum_{j=1}^{n} a_{ij} X_j = 1 \qquad for\ i = 1, \dots, m \qquad (1.2)$$

$$X_j \in \{0,1\} \qquad for\ j = 1, \dots, n \qquad (1.3)$$

There are m flight legs and n pairings. In above formulation, j is the index for flight legs and i is the index for pairings. Each constraint (row) corresponds to a flight leg that must be serviced, and each variable (column) corresponds to a legal pairing. Objective function (1.1) minimizes the total crew cost, where c_j is the cost of pairing j. Constraint (1.2) ensures that each flight leg will be covered by exactly one pairing. Coefficient a_{ij} equals 1 if flight leg i covered by pairing j and 0 otherwise. X_j is a 0-1 decision variable that assumes the value 1 if pairing j is selected and 0 otherwise.

By defining variables as sequences of flight legs, only feasible pairings are considered and explicit formulation of complicated work rules is unnecessary. Moreover, nonlinear pairing costs can be computed offline and captured as a single value. The problem is then a set partitioning problem, with a linear objective function and binary decision variables as seen above. The one drawback to this modeling approach is that there are potentially many billions of possible crew

pairings, especially in hub-and- spoke flight networks with numerous aircraft at hubs at the same time, allowing many possible crew connections.

Because of this complex nature of the problem, the researchers have used various solution approaches to be able solve this problem efficiently. Mostly used solution techniques are set partitioning problem embedded with column generation techniques, network flow problems, metaheuristics and optimizers developed using heuristics.

1.1.5 Crew Rostering

Crew rostering problem combines the crew pairings into equitable and efficient month-long crew schedules, called bidlines or rosters. The problem assigns individual crewmembers (pilots, copilots, crew attendants and other airline personnel) to pairings, using a bidline or preferential model. In other words, it is the construction of individualized schedules that take into account various pre-assignments as well as crew particular needs and requests. Airline companies have the monthly task of constructing personalized monthly schedules (rosters). When assigning crew members, consideration must be taken of planned vacations, sick leave, days planned for schooling and training, requested free days, days when medical examinations are held, etc.

With bidline model, a generic set of schedules is constructed and individual employees reveal their relative preferences for these schedules through a bidding process. The specific assignment of schedules to employees is based on seniority; with the more senior crewmembers most often assigned their preferred schedules.

In preferential model, pilots place weights on characteristics, which they value in a schedule. Preferential models find the optimal schedule for each pilot in order of seniority.

Since the problem is complex, combinatorial by nature, and has large dimensions, rostering focuses on finding a feasible solution rather than maximizing any particular

utility function. The complexity of airline rostering is higher than other applications' complexity because:

- a pairing may cover a period of several days,
- a rotation is generated for a specific person with specific needs,
- depending on the aircraft type a varying number of crew with specific skills is required
- the size of a real airline rostering problem and the typically large sets of data contribute to the complexity of the rostering procedure

The pilots of most world airlines usually fly only one type of aircraft. It is very rare for a pilot to fly several types of aircraft during the month. This situation is primarily owing to safety considerations (the tendency is to adapt the pilot as much as possible to a certain type of aircraft). Therefore, when a fleet has different aircraft types, monthly schedules are first made for the pilots of one type of aircraft, then for the pilots of another type of aircraft, etc.

Unlike pilots, stewardesses and stewards most often have licenses to work on all types of aircraft in the airline’s fleet. In practice there are two possible ways to assign cabin crew.

- The first is to divide the cabin crew by type of aircraft and then assign them in the same way as pilots are assigned. After a time, the cabin crew usually change the type of aircraft on which they fly.

- The second way entails the simultaneous assignment of all cabin crew to the rotations of all types of aircraft. In this case, the dimensions of the problem become quite large.

Some tasks have a higher priority than others. Also, some crew members have a higher priority than others. In some cases the crew rostering problem is solved by assigning higher priority tasks to higher priority crew members.

1.2 Irregular Airline Operations

The airline operations problems are solved as the airline plan is being implemented. The "optimal" schedules obtained from the solution of the airline planning problems rarely operate as planned. Because of the time lag between planning and operations, circumstances may have changed at the time of operations, so that the planned schedules are not no longer feasible or desirable. These planned schedules are often disrupted by maintenance problems or severe weather conditions. In a typical day, several flights may be delayed or canceled, and aircraft and crews may miss the rest of their assigned flights. Resources in the airline industry are capital intensive, thus are tightly coupled. The removal of even a single aircraft from the flight schedule for a short period will cause flight delays and cancellations.

The most common reasons for these disruptions are crew unavailability, unscheduled maintenance problems, gate delays, bad weather conditions, station congestion and airport facility restrictions. When any of these problems occur, operations personnel must make real-time decisions that return the airline to its original schedule as soon as possible. Since these situations can be very costly for airlines, resulting in substantial loss of revenue and customer goodwill, cost-effective solutions should be obtained.

It is very difficult to estimate the cost of schedule disruptions. Some researchers have been made for this. For example; New York Times reported in 1997, a major US airline has more than $400 million per year in crew overpay, lost revenue and passenger hospitality costs. Shavell (2000) gave another example that in 1998 the direct disruption costs for the domestic operations of 10 major US airlines are $1.83 billion where $858 million due to flight cancellations and $909 million due to flight delays.

1.2.1 The Sources of Disruptions

There are two major sources of disruption, namely, airline resources shortages and airport and airspace capacity shortages.

1.2.1.1 Airline Resource Shortages

i. *Crew Delays:* A crew delay occurs when the scheduled crew to fly a leg is unavailable. The crew may be delayed due to a previous flight, or there can be a delay while reserve crew is summoned.

ii. *Gate Delays:* A gate delay occurs when a disruption is due to actions occurring at the gate within an airport. Such a delay may occur due to connecting passengers, loading luggage or cargo, fueling, overbooking and so on.

iii. *Unscheduled Maintenance Delays:* Planes may have unexpected maintenance problems. In contrast to scheduled maintenance, which occurs at regular intervals, unscheduled maintenance events may occur at any time. A plane, which has an unscheduled maintenance event may be unavailable for a period of time. Unscheduled maintenance events may be relatively trivial, such as a broken seat, or serious, such as a faulty engine. The duration of unscheduled maintenance delays varies widely; it can be ranged from one minute to ten days. Lengthy disruptions may have serious consequences for an airline's operations. While airline maintains reserve crews, which fly in emergencies, they do not commonly use reserve planes. If a plane is forced to miss an extended period of time, an airline must typically cancel legs.

iv. *Lack of ground sources:* Due to lack of ground resources aircraft turn times become longer than scheduled

1.2.1.2 Airport and Airspace Capacity Shortages

i. *Weather:* Poor weather conditions may reduce the visibility at an airport. This may increase the distance required between airplanes during takeoffs and landings. Consequently, the number of takeoffs and landings during poor weather may be fewer than during normal operating conditions. Therefore, airport throughput is reduced.

Weather disruptions are seasonal. For example, snowstorms are common in winter and thunderstorms are more prominent in the summer months. They also vary by airport. A weather system may move from one region of the country to another, affecting different airports, so weather disruptions are dependent between airports.

ii. *Airport Congestion Delays:* These delays may occur due to factors such as traffic control and disasters. When the number of departures and arrivals at an airport exceed the airport's airfield capacity, flight legs get delayed or cancelled generating irregularities

1.2.2 Recovery

An airline must react to disruptions, which cause the irregularities in the planned schedules. How an airline reacts in case of the schedule irregularities is called recovery. Recovery is very important to an airline because costs may be high during disrupted operations. Most recovery decisions in current practice are made manually, without the benefit of decision support. In practice, recovery decisions are made in the *Airline Operations Control Center* (AOC). Various coordinators work together to find real-time recovery solutions when an airline is faced with a disruption. An AOC must consider planes, pilots, passengers, flight attendants and cargo.

The most commonly used techniques to recover the schedules in case of disruptions include:

- *Add Slack in the plans:* Usually, crew rules state a minimum turn time. As well as crews, aircrafts also need a turnaround time between landing and taking off in order to load/ unload the baggage, clean the aircrafts etc. Instead of operating all day at a minimum turn time, slack is incorporated into the plans such that each line of work has some degree of "self-recovery".

- *Out and back:* If an aircraft flies from a hub to a spoke and back to the same hub, these two flights can be canceled without affecting the rest of the aircraft

schedule. If the same crew is planned for these two flights, the cancellation will not affect the rest of the crew schedule either.

- *Using standby crew and aircraft:* A spare crew or an aircraft are valuable but very costly resources that can be used in case of disruption. Standby crewmembers are positioned at the major stations and ready to substitute any other crewmember that can not fly her/his flight. Standby aircrafts are positioned at the major airports in order to be used in emergency.

- *Using reserve crew:* Reserve crewmembers are positioned at their homes and ready to work after a predefined time interval to cover any open flight(s). This predefined interval is usually set to allow the crewmember to get ready and get a transport to the airport.

- *Using stranded crew:* Crewmembers that are stranded at stations due to cancellation or misconnecting their next flights can be used as recovery crew for any other crew problems at these stations. They can also be deadheaded to other stations where they can get new assignments.

- *Deadheads:* A crewmember can be deadheaded from a station to cover an open flight at another station. In general, crewmembers can be deadheaded to be repositioned at any station or returned to their domicile.

- *Crew swapping:* During system disruption, crewmembers of the same qualification can exchange their jobs. In this case, an earlier flight with crew problem is covered with the crew of a later flight with the condition that the crew of the problem flight will be able to fly this later flight.

- *Aircraft swapping:* During system disruption, aircrafts of the same fleet type or same specifications can exchange their predefined flight legs. In this case, an earlier flight with aircraft problem is covered with the aircraft of a later flight.

- *Extra buffers added to turnaround times:* Extra buffers are often added after frequently delayed flights, but they also provide slack in the schedule that can be used in recovery.

- *Increased cruise speed:* Aircraft have an interval of possible cruise speeds. Airlines will typically operate aircraft at the most economic speed, which will always be lower than top speed. Speed up, which is the name for increasing speed, represents an additional cost due to increase fuel burn, but it may avoid higher costs in itinerary repair for aircraft, crew and passengers.

- *Delaying (re-quoting):* A flight can be delayed to wait for its connecting crew or until its original crew gets their required rest period. Also, if a new crew is assigned to this flight, the flight can be delayed until the new recovery crew gets ready. A flight can also be delayed to wait its connecting aircraft or until its originally assigned aircraft gets ready.

1.2.3 Aircraft Recovery Problems

Since the aircraft is the most expensive resource of the airlines, its effective utilization is very important. The large savings are possible using aircrafts more efficiently.

When any aircraft shortage occurs due to some reasons, operations personnel use a combination of some recovery strategies. These aircraft recovery strategies are:

- delaying flights
- canceling flights
- swapping aircraft among scheduled flights
- requesting the usage of spare aircraft
- ferrying surplus aircraft

Operations personnel have to make real time decisions as to the set of flights that need to be cancelled or delayed. They choose to delay some flights in an attempt to

avoid cancellations. Spare aircraft are sometimes maintained at major hubs for use in emergencies to maintain an operable schedule. Ferrying involves flying an empty aircraft to a point of need to service a flight. But the usage of surplus aircraft is unusual due to excessive costs.

Many complex factors have to be considered before making such a decision. The chosen sets of cancellations and delays should be the ones which cause the minimum loss in direct revenue and indirect costs like costumer goodwill. Loss in direct revenue includes:

- Ticket refunds for passenger choosing other airlines
- Hotel rates for passengers choosing to wait overnight
- The profit loss by not operating scheduled flight
- Direct costs incurred due to flight delays
- Overtime pay
- Additional fuel expenses

The schedule should remain operable after the cancellation and delays. Each scheduled flights should have an aircraft.

In the literature the researchers have mostly studied on aircraft recovery problems. They generally modeled the problem as multi-commodity network flow problems.

1.2.4 Aircrew Recovery Problems

Crew recovery problem is more complex due to the number of crew and more complex rules and regulations for crew.

Available crews that could be used in the recovery process include;

- Standby crews
- Reserve crews
- Stranded crews

- Good (undisrupted) crews
- Deadheaded crews

Two main operational constraints control the use of these crewmembers:

- Crew assigned to operate a flight must be qualified for this flight and be compatible with other resources assigned to the flight. For example, if an aircraft of certain fleet is assigned to a flight, crewmembers that are assigned to the same flight have to be qualified to fly this fleet.

- Crewmembers should comply with all mandatory operation rules set by the government and the union agreements. For example, if a flight is delayed, this delay should cause no violation to the crew duty limits or the rest requirements. All regulations and crew agreement rules that govern the operation of the different resources are assumed given. These rules include crew legal rest requirements and crew duty period limits.

The objective is to develop an efficient recovery plan to fix all disrupted flights in the horizon such that the total system cost is minimized. This recovery plan is developed in terms of new crew pairs that cover all flights in the horizon under consideration.

In order to solve crew recovery problems, followings are needed to know as input of the problem:

- Current airline schedule, represented in terms of the list of flights scheduled by the airline (each flight is defined by its number, origin, destination, scheduled departure/arrival times,).
- The workloads of crewmembers defined in terms of their current pairs.
- The list of standby/reserve crew at each station.
- The horizon at which crew problems are to be considered.
- All mandatory operation rules set by the government and the union agreements that govern crew operation.

1.2.4.1 Crew Problems in Case of Disruptions

Crews are usually classified based on their qualification and experience to the positions ranked from the senior to lower. Each crewmember is qualified and trained to fly certain fleet type, and she/he can not fly any other fleet at any circumstances.

Crew schedules are typically set in pairs. Each pair consists of a sequence of flights (segments), the first of which starts at the crew base and the last ends at crew base. Each segment consists of a single take-off and landing, with no intervening stops. A pair typically extends over two to five days and consists of a successive set of duty periods and rest periods (layover). Figure 1.2 shows a typical 2-day pair that starts and ends at station A and has a layover at station H.

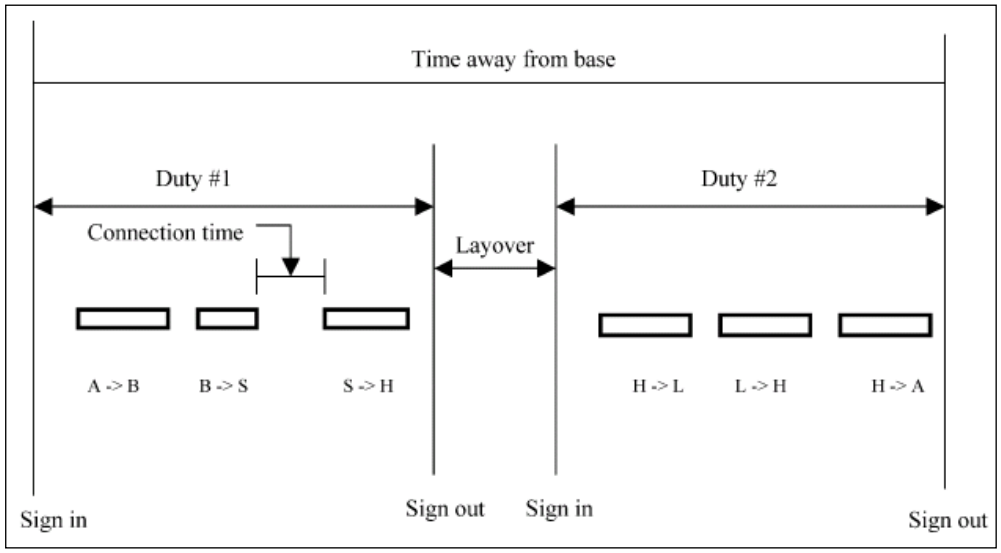

Figure 1.2 Crew pair in normal operation conditions

The duty period starts about one hour (briefing period) before the departure time of the first segment on the duty period and ends about 15 minutes (debriefing period) after the arrival time of the last segment of the duty period. The lengths of the duty and the rest periods are determined based on the governmental regulations and the union agreements to guarantee safe operation and good quality of life for the crew. Between two successive segments in the same duty period, crewmembers are given a reasonable connection time that is equal to or greater than the minimum connection time at this station. This connection time allows a crewmember to move to the

departure gate of the next segment, if it is different from the arrival gate of the current segment.

Due to the irregularity that could happen during operation, schedules could get disrupted such that crew no longer follows their planned schedules (pairs). Due to these disruptions, different possible crew problems can occur.

1. Misconnect problem : It occurs when a connecting crewmember arrives late such that she/he is unable to fly the next flight, in the same duty period, on time. Figure 1.3 shows an example of a typical misconnect problem.

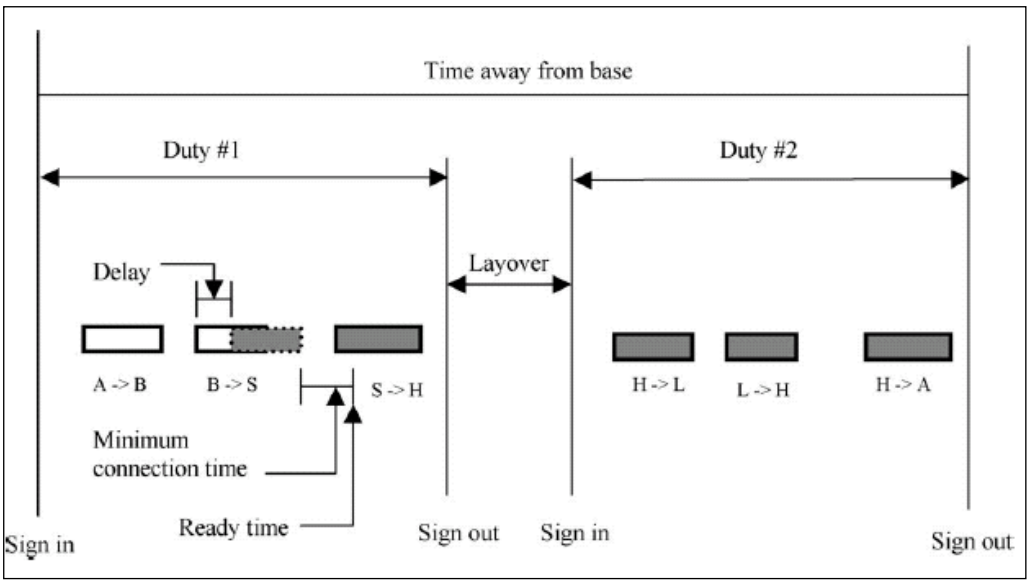

Figure 1.3 Crew pair (Misconnection problem)

When flight B–S is delayed, the new connection time became less than the minimum required connection time for the crew. Therefore, another crewmember should be found at station S for the next flight (S–H) in order to depart on time.

2. Rest problem : It occurs when a crewmember gets a rest period (layover) that is less than the minimum required rest period. This could be due to late arrival at the end of the previous duty period. In this case, the crewmember would be unable to fly the first segment of the next duty period on time. This problem is a special case of the misconnect problem by considering the required rest period as the required connection time. Figure 1.4 shows a typical example of the rest problem.

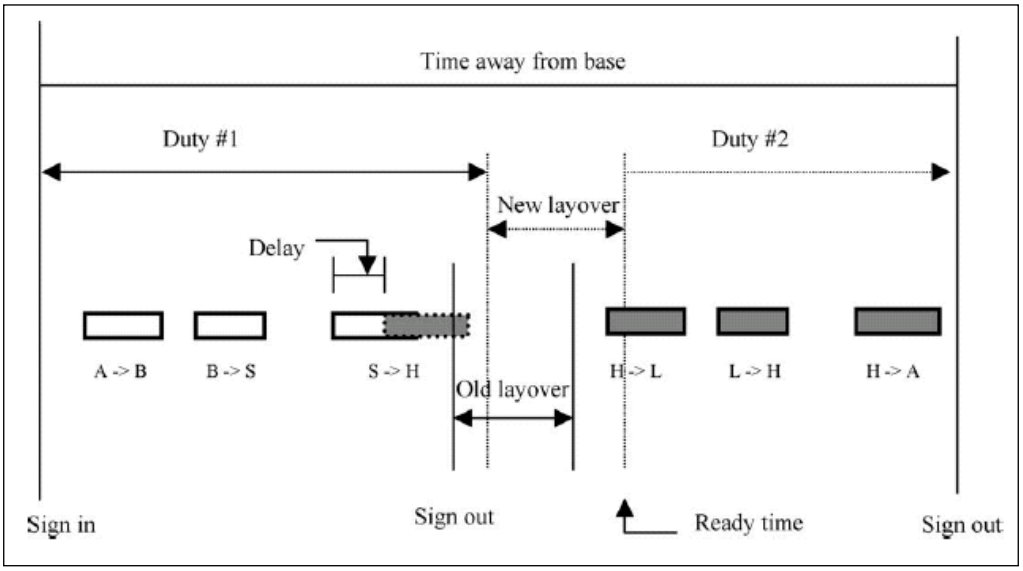

Figure 1.4 Crew pair (Rest problem)

When flight S–H is delayed, the layover becomes less than the minimum required rest period. The first flight of the next duty period (flight H–L) can not depart on time by the originally assigned crew, since the crew has to get the minimum legal rest at station H.

3. Duty problem : It occurs when a duty period limit is exceeded due to a delay for one or more of the flights of this duty period. This delay could be due to ground holding, unplanned aircraft maintenance, longer unexpected customer service time, etc. In this case, the crew can not fly the last flight(s) of the duty period, due to this duty problem. Figure 1.5 shows a typical example of a duty problem.

Flight S–H is delayed and its arrival time passes the duty limit of the crew. Therefore, another crew should be found at station S to fly flight S–H.

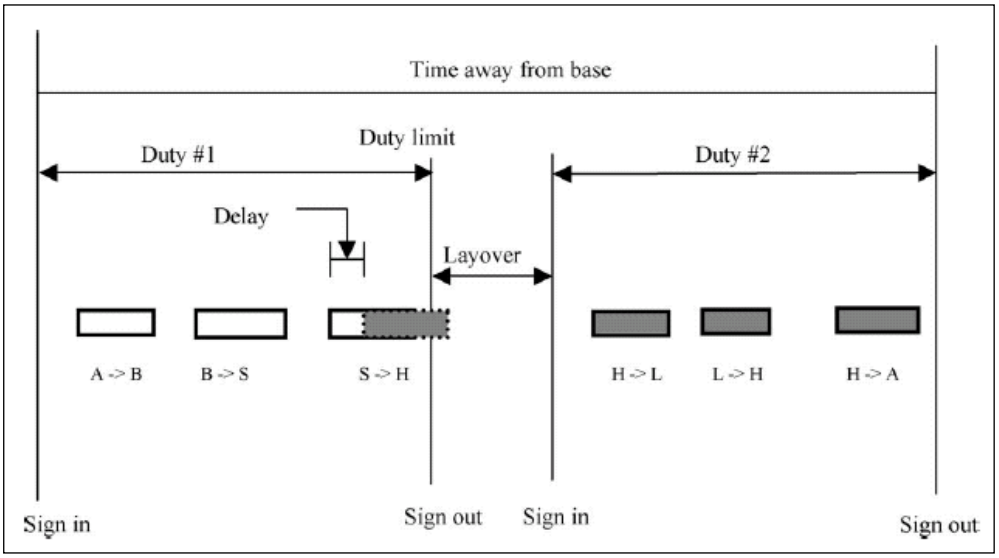

Figure 1.5 Crew pair (duty problem)

4. Unassigned problem: It occurs in case of no show for a crewmember due to cancellation of her/his inbound flight or any other possible reason such as calling sick.

If a duty problem occurs, its crewmember would not be able to fly any segment that has arrival time beyond the duty period limit. This means that there is no chance that this crewmember can fly the remaining portion of her/his pair as it was originally planned. On the other hand, a misconnecting crewmember or crewmember with rest problem may still be able to fly the remaining part of the pair, if delaying the subsequent flight(s) is acceptable. However, if the proposed delay is greater than the operationally accepted delay or if this delay would cause a crew duty violation, this crewmember will not be able to fly the remaining portion of her/his pair and another crewmember should be found.

1.3 Motivation and Contributions

Since the air transport industry has always been an interesting research area for decades, the researchers have developed several operations research techniques and optimization tools, which can be applied to airlines' planning and operations problems since 1950s. Currently, most airlines are capable of performing their key planning process, including revenue management and aircraft and crew scheduling,

aided with sophisticated software tools.

Although the airlines are capable of managing their planning process and schedules using such optimization tools, they still try to find a way to recover their planned schedules as soon as possible when they face with a disruption. There have been significant studies on the airline recovery problems during the last decades in operations research field and in the last years also commercial tools for disruption management have become available. However, these problems are still challenging and subject to further improvements. Therefore, this dissertation focuses on developing new effective models and solution algorithms for the aircraft and crew recovery problems.

The first challenge in airline recovery problems is that aircraft and crew recovery problems have mostly considered individually in the literature. The developed solution algorithms for the airline recovery problems generally involve only one resource's shortage, either aircraft or crew. However, the crew related and aircraft related decisions affect each other. In order to operate any flight leg there must be at least 1 available crew and 1 available aircraft to be assigned to the flight leg.

The second challenge in airline recovery problems is that the integration of the airline resources causes complex recovery models and requires special attention to be solved to optimality. Because the integrated crew and aircraft recovery problems includes both crew related and aircraft related constraints and variables. Thus, the number of constraints and variables become very huge compared to the individual recovery problems. Because of the increased complexity and number of variables, the integrated problem is NP-hard.

The third challenge in the airline recovery problems is that the airline recovery problems in the literature have been modeled as multi-commodity network flow problems in order to obtain feasible and optimum (or near-optimum) solutions. Therefore, the resulting mathematical programs are more or less identical. With the improvement of optimization technology and computer systems, new solution

approaches may be developed in order to solve airline recovery problems in real-time with less effort using more computer based technology.

This dissertation contributes by;

- Providing a detailed literature review for aircraft and crew recovery problems. A time framework of the problem, main authors, proposed solution methods and their contributions as well as research perspectives for the problems will be presented. The impacts and challenges in the airline recovery problems will be discussed.

- Developing an iterative solution algorithm for the integration of aircraft and crew recovery problems. By integration, we mean connecting the aircraft and crew recovery problems via linking constraints rather than combining the aircraft and crew related constraints into a big model, which can not be solved easily because of its size and complexity. The iterative and sequential solution algorithm developed for such integration considers both aircraft and crew availabilities and their correlations. In such algorithm aircraft and crew recovery problems are modeled separately. However linking constraints are used in order to link the effects of aircraft's and crew's availabilities. Therefore, our way of integration is much more manageable and effective than the classical way of integration. This contributes to the first and second challenges mentioned above.

- Proposing the application of a novel approach for aircraft recovery, crew recovery and their integration. This new approach includes both constraint programming and integer programming technique. Therefore, we take advantage of constraint programming technique, which offers a more flexible modeling framework than mathematical programming. Although constraint programming technique is widely applied to scheduling and planning problems in the literature, to best of our knowledge there has not been any published work on applying it to the airline recovery problems. This contributes to the third challenge mentioned above.

1.4 Organization of Dissertation

This dissertation includes 8 Chapters:

After this introduction chapter, a literature review on the airline planning and operations problems is presented in Chapter 2. The important publications and the most recent studies on the problems in airline planning; which are schedule development, fleet assignment, aircraft rotation, crew scheduling and crew rostering, are summarized and the impacts and challenges of each problem are discussed. Furthermore, the literature on the problems under consideration, which are aircraft recovery, crew recovery and integrated recovery problems, is given more detailed, explaining each proposed model and used solution approaches. Finally, the impacts and challenges of the airline recovery problems are discussed.

Chapter 3 presents the methodology to be employed in the proposed models and solution algorithms in this dissertation. While developing and solution algorithms for aircraft, crew and integrated recovery problems following techniques are used: Multi-commodity network flow model as a traditional solution approach and constraint programming technique as a novel solution approach. In chapter 3, brief overview of these techniques is presented.

Chapter 4 presents the mathematical models developed for aircraft recovery problems. In this dissertation, 4 different models are presented, three of which use the multi-commodity network flow models and the fourth one uses constraint programming technique. Numerical examples are given for each proposed model and lastly, evaluation and comparison of each aircraft recovery model are given at the end of the chapter.

Chapter 5 presents the mathematical models developed for crew recovery problems. In this dissertation, 2 different models are presented for crew recovery problems; the first one uses the multi-commodity network flow models and the other one uses constraint programming technique. Numerical examples are given for each

proposed model and evaluation and comparison of each crew recovery model are given at the end of the chapter.

Chapter 6 presents the novel solution algorithm for integrated aircraft and crew recovery problems, which solves aircraft and crew recovery problems in iterative and sequential manner, taking into account the correlation between aircraft and crew availabilities. The solution algorithm proposed in Chapter 6 uses multi-commodity network flow technique for modeling aircraft and crew recovery problems. Numerical examples for proposed solution algorithm are given and quality of the solution is evaluated in terms of computational times and number of impacted resources.

Chapter 7 presents the novel solution algorithm for integrated aircraft and crew recovery problems using a novel technique constraint programming, instead of traditional multi-commodity network flow models. Proposed solution algorithm solves aircraft and crew recovery problems in iterative and sequential manner, taking into account the correlation between aircraft and crew availabilities. Numerical examples for proposed solution algorithm are given and quality of the solution is evaluated in terms of computational times and number of impacted resources.

Finally, Chapter 8 presents a section for the summary, conclusions and further research areas. Also, at the end of the dissertation is devoted to appendices, where the flight schedules, initial aircraft-crew schedules of the example airlines are presented and OPL models and script codes for the developed models are included.

CHAPTER TWO
LITERATURE REVIEW ON AIRLINE PLANNING AND OPERATIONS

2.1 The Literature Review on Airline Planning Problems

The most of the literature on the airline industry focuses on airline planning problems. Since the researchers have been studying on these problems during more than 60 years, there is large number of published works in this area. Since the main focus of this dissertation is not the airline planning problems, all the literature will not be explained in detail. However, the important publications and the most recent studies will be summarized; the impacts and challenges of these studies will also be discussed.

2.1.1 The Literature Review on Schedule Development Problems

Flight schedule problem has been of interest to researchers for many years. Recent papers refer the studies of Simpson (1966), Chan (1972), Soumis, Ferlan & Rousseau (1980), and Etschmaier & Mathaisel (1984) who described the early work. Furthermore, the studies of Dobson & Lederer (1993), Berge (1994), Marsten & Berge (1994) have made significant contributions to the flight schedule problem. The most recent studies have been performed by Lohatepanont & Barnhart (2004) and Kim & Barnhart (2007). The models developed for flight schedule problem focus on to improve the existing schedule rather than creating the whole schedule from the beginning. The models and algorithms developed select the flight legs that will be added to or removed from an existing schedule, from a subset of candidate flights legs.

Dobson & Lederer (1993) study airline scheduling and routing problem in a hub-and-spoke system. The paper has three contributions: First, an expression calculating demand for each route as a function of the service quality and price of all routes is derived. Second, a mathematical programming heuristic is developed to find the flight schedules and route prices. Third, the heuristic is used for studying competition

in a hub-and-spoke system by allowing each airline to optimize its schedule and prices against the other's choices.

The recent and important study on schedule design problems is performed by Lohatepanont & Barnhart (2004). They describe integrated models and solution algorithms that simultaneously optimize the selection of flight legs and the assignment of aircraft types to the flight legs. They solve problems at one major airline that contain about 800 potential flight legs, 65,000 itineraries, and 165 aircraft, resulting in formulations with 30,000–60,000 rows and 50,000–65,000 columns, and solution times ranging from 12 hours to more than 3 days. They report potential improvements in aircraft utilization and significant increases in revenue, with an estimated impact exceeding $200 million at that airline.

Kim & Barnhart (2007) consider the problem of designing the flight schedule for a charter airline. Exploiting the network structure of the problem, they develop exact solution approaches and compare their results. They first consider the special case of a single fleet type and develop a model and fast solution approach. Then they adapt the single fleet model to create a heuristic approach to the multiple fleet problem.

2.1.1.1 Impacts and Challenges of Schedule Development Problem

Because of the complexity of the schedule design problem and immerse size of the problem, the optimization models and methods developed so far have not fully satisfied the requirements. The main challenges faced in schedule development problem are as follows:

- *Complexity and Problem Size:* Decisions involving schedule design are integrally tied to the assignment of aircraft and crews to the flight legs. Incorporating crew, fleeting, and schedule design decisions into a single model, involves numerous flight-leg options and complex constraints and interdependencies. Such a model will be intractable, containing many billions of constraints and variables.

- *Data Availability and Accuracy*: The inputs for designing the flight schedule include the airline's expected *unconstrained market demands* and *average fares*. The airline's unconstrained market demand is the maximum demand that the airline will experience for an origin-destination market. It is independent from the capacity provided in that market. Moreover, an airline's demand is influenced by the schedule of competing airlines. Average fares are affected by complex pricing and revenue management systems, the order in which customers request tickets for a flight, and the demand and capacity in a market—which are themselves dependent on the flight schedule. Further complicating the estimation of fares are the competitive pressures in the industry that force airlines to adjust their fare structures dynamically, reacting to pricing changes made by their competition.

Because of these difficulties, practically schedules are developed manually, with limited optimization. On the other hand, by limiting the complexity and scope of models, recent progress has been made in applying optimization to schedule design problems. Researchers have focused on determining incremental changes to flight schedules, producing a new schedule by applying a limited number of changes to the existing schedule. Generally, the approach is to cast these problems as network design models, with the objective of maximizing the incremental profits of the new schedule.

Schedule generation represents an important area for future research with its important strategic and financial implications. Future research is needed to capture the critical interactions among the various resources of the airline, its competitors, and airports.

2.1.2 The Literature Review on Fleet Assignment Problems

The researchers have given great attention to the solution of fleet assignment problems since the aircrafts are the most expensive resource of the airlines and its effective utilization is very important. There is a large number of published works in this area. The Table 2.1 summarizes the literature on fleet assignment problems.

Table 2.1 The summary of the literature review on fleet assignment problems

Year	Author	Basic/Integrated (1)	Model type (2)	Underlying Network (3)	Planning horizon (4)	Additional considerations					Application (5)	Solution approach				
						Time windows	Flight Schedules	Crew	Maintenance	Itinerary		LP Relaxation	Branching	Column generation	Dynamic Programming	Heuristic
1971	Levin	B	NFP			✓							✓			
1989	Abara	B	MCF	C	D						R	✓				
1989	Daskin, Panayotopoulos	B	ILP		D		✓				T	✓				✓
1993	Berge, Hopperstad	B	MCF	TS	D		✓				R					✓
1994	Subramanian et al.	B	MCF	TS	D				✓		R					
1995	Hane et al.	B	MCF	TS	D						R	✓	✓			✓
1996	Clarke et al.	B	MCF	TS	D			✓	✓		R					
1996	Yan, Young	B	MCF	TS	D		✓			✓	R					✓
1997	Rushmeier, Kontogiorgis	B	MCF	C	D			✓			R	✓	✓			
1997	Desaulniers, Solomon, Soumis	I	MCF		D	✓			✓		R	✓	✓	✓		
1998	Barnhart et al.	I	SBF		W				✓		R		✓	✓		
1999	Ioachim et al.	I	MCF		W	✓	✓		✓		R		✓	✓	✓	
2000	Rexing et al.	B	MCF	TS	D	✓	✓				R		✓			✓
2000	Moudani, Camino	B	ILP		D				✓		R				✓	✓
2002	Barnhart et al.	B	MCF	TS	D					✓	R					✓
2002	Yan,Tseng	B	MCF	TS	D		✓			✓	R	✓				✓
2004	Lohatepanont, Barnhart	I	NFP		D		✓			✓	R	✓	✓	✓		✓
2006	Bélanger et al. (a)	B	MCF	TS	W					✓	R					✓
2006	Bélanger et al. (b)	B	MCF	TS	W					✓	R		✓			
2007	Haouari, Aissaoui,Mansour	B	MCF		W						R					✓
2008	Yan, Chen, Chen	I	MCF	TS	W	✓	✓				R					✓
2008	Tang, Yan, Chen	I	MCF	TS	W		✓				R	✓				✓

(1) B: Basic, I: Integrated
(2) MCF: Multi-commodity Flow, ILP: Integer Linear Programming, NFP: Network Flow Problem, SBF: String Based Flow
(3) C: Connection type, TS: Time-space type
(4) D: Daily, W: Weekly
(5) R: Real data, T: Test data

The very first researches on fleet assignment problems were performed by Levin (1971), Abara (1989).

Levin (1971) was the first researcher to present scheduling and fleet routing model with time windows, considering only single type of aircraft. Abara (1989) was one of the first researchers to address realistically sized daily fleet assignment problems using a connection-based network structure.

In the literature, basic fleet assignment models have usually been developed using the multi-commodity network flow models. Berge & Hopperstad (1993), Hane et al. (1995) formulate the daily fleet assignment problem as a multi-commodity network flow problem with side constraints based on a time-space graph representation. However, Rushmeier & Kontogiorgis (1997) model the large scale fleet assignment problem as mixed-integer multi-commodity flow problem where the underlying network is connection network.

Some researchers solve the fleet assignment problem taking into consideration of other planning decisions. Subramanian, Scheff, Quillinan, Wiper & Marsten (1994) consider maintenance issues in the fleet assignment model. They use maintenance arcs in the time-space network and applied this model to Delta Airlines. Clarke, Hane, Johnson & Nemhauser (1996), based on Hane et al.'s (1995) basic model, develop a fleet assignment model which takes aircraft maintenance and crew scheduling into consideration. Yan & Young (1996) consider schedule design and fleet assignment problems together. Rexing, Barnhart, Kniker, Jarrah & Krishnamurthy (2000) present a generalized fleet assignment model for simultaneously assigning aircraft types to flights and scheduling flight departures. Yan & Tseng (2002) develop a model by improving Yan & Young's (1996) model to solve integrated scheduling model for multi-fleet routing and flight scheduling. Furthermore, Lohatepanont & Barnhart (2004) develop models that simultaneously solve schedule design and fleet assignment problem. This study has been mentioned in the Section 2.1.1.

Most of the fleet assignment literature considers only daily fleet assignment problem assuming that the flight schedules repeat daily. But in reality, flight schedules may vary over the days. Taking into account this reality, Bélanger, Desaulniers, Soumis, Desrosiers & Lavigne (2006) and Bélanger, Desaulniers, Soumis, Desrosiers (2006) study on fleet assignment problems in different time horizons. Bélanger, Desaulniers, Soumis, Desrosiers & Lavigne (2006) address the weekly fleet assignment problem and Bélanger, Desaulniers, Soumis, Desrosiers (2006) consider a periodic scheduling horizon for fleet assignment problem.

Basic fleet assignment problems assume that the number of spilled passengers, and their associated spill costs, can be computed at a flight-leg level. In fact, passenger demand, spill, and the revenue associated with each passenger are itinerary specific, not flight-leg specific. Barnhart, Kniker & Lohatepanont (2002) are the first researchers to address itinerary based fleet assignment model. Their model is capable of capturing network effects and estimating spill and recapture of passengers more accurately. They combine basic fleet assignment model and passenger mix model.

Besides multi-commodity network flow models, some researchers have applied different techniques for solving fleet assignment problems. Daskin & Panayotopoulos (1989) present an integer program for fleet assignment problem which maximizes the profit. Heuristic algorithm is used for find primal feasible solutions and for improving solutions. Haouari, Aissaoui & Mansour (2007) study on solution techniques rather than proposing new modeling approaches. They describe two phase heuristic: First initial feasible solution is built by iteratively solving a sequence of standard linear assignment problem. Then, an improved solution is obtained by solving a minimum-cost flow problem for each aircraft type. The performance of solution approaches is tested using the six real-data instances provided by TunisAir. The near-optimum solutions are obtained in very short computational time.

2.1.2.1 Integrated Fleet Assignment and Aircraft Routing Problems

During the airline planning process, the fleet assignment information fed into aircraft maintenance routing problem. While many researchers have studied these problems separately, there has also been significant attempt to solve them simultaneously. Desaulniers, Desrosiers, Dumas, Solomon & Soumis (1997) and Barnhart, Boland et al.(1998) are the first researchers to solve simultaneously the fleet assignment and aircraft routing problems

Desaulniers et al (1997) solve the daily aircraft routing and scheduling problem (DARSP) utilizing a set of operational flight legs with known departure time

windows. They propose two integer programming models: a set partitioning type model and a time constrained multi-commodity network flow model.

Barnhart et al. (1998) develop string based model to solve simultaneously the fleet assignment and aircraft routing problems. A string represents the sequence of connected flights that begins and ends at the maintenance station. Their model is more flexible than the traditional models, because nonlinear, complex costs and constraints can be modeled easily.

Ioachim, Desroisers, Soumis & Belanger (1999) introduce a new type of constraints, related to schedule synchronization, in the problem formulation of aircraft fleet assignment and routing problems. The problem covers one week period for long haul operations. It involves times windows and also requires schedule synchronization constraints over the different days of the week.

Moudani & Camino (2000) propose a dynamic solution approach in order to solve aircraft fleet assignment and maintenance scheduling problems simultaneously. The proposed solution mixes two techniques: Dynamic programming approach is used for fleet assignment problem and a heuristic technique is used for solving the embedded maintenance schedule problem.

2.1.2.2 Impacts and Challenges of Fleet Assignment Problem

By using multi-commodity flow-based fleet assignment models significant cost savings, measuring in the millions of dollars annually, are achieved. For example, Rushmeier & Kontogiorgis (1997) report realized savings of at least $15 million annually at USAir. Abara (1989) reports a 1.4% improvement in operating margins at American Airlines.

Research on fleet assignment problems has led to advanced techniques for solving general linear programs. For example, the node consolidation idea introduced in Hane et al. (1995) reduces formulation size by more than 40% for the problems of

one major airline. This idea has been generalized and incorporated into commercial solvers, allowing more efficient solution of large-scale optimization problems.

Barnhart et al. (2002) introduce itinerary or origin-destination (O-D) based fleet assignment approaches. These approaches consider passenger fares, demand, spill, and recapture to be itinerary, not flight-leg, specific. Using their O-D-based fleet assignment approach, in a case study involving a major U.S. airline, Barnhart et al. (2002) report that itinerary-based fleet assignment improves on leg-based fleeting significantly, with estimated savings ranging from 30 million to over 100 million dollars annually.

In a first step at integrating fleet assignment and schedule design decisions, Rexing et al. (2000) simultaneously assign an aircraft type to each flight leg and select each flight leg's departure time, allowing re-timings of 5–20 minutes from the current schedule.

On the other hand, most of the traditional approaches proposed for the FAP solve the FAP in isolation from the other airline scheduling processes and under restrictive assumptions such as:

1. Many fleet assignment models assume that the flight schedules repeat daily, even though most airlines operate different schedules on the weekend.

2. Most fleeting models assume flight leg demand is known and does not vary by day of week, but day-to-day demand may vary.

3. Flying times and ground times are typically assumed to be deterministic in fleet assignment models; however, congestion on the ground and in the air, weather conditions, and new security practices produce large variations in flight and ground times.

4. Most fleet assignment models use point forecasts for flight-based demands instead of itinerary- based or path-based demands. In other words, they assume that

the number of spilled passengers, and their associated spill costs, can be computed at a flight-leg level. In fact, passenger demand, spill, and the revenue associated with each passenger are itinerary specific, not flight-leg specific. As a result, it is possible to estimate leg-specific spill costs only approximately.

5. In the fleet assignment problems, customers rejected from their requested itineraries (due to capacity restrictions) are often assumed to be lost. In reality, they may choose to take another route that is comparable to their first choice itinerary in terms of the origin, destination, and time-frame (i.e., they may be recaptured). The recapture effect has mostly been ignored in analytical studies until very recently.

2.1.3 The Literature Review on Aircraft Routing Problems

In the literature, aircraft routing problems are usually considered together with other airline decisions like fleet assignment and crew scheduling. The literature about the integrated fleet assignment and aircraft routing problems has been mentioned in Section 2.1.2.1. The literature of integrated aircraft routing and crew scheduling problems will be presented in section 2.1.4.1. On the other hand, there are also researchers who study aircraft routing problems alone:

Clarke, Johnson, Nemhauser & Zhu (1996) formulate the aircraft rotation problem as Euler tour problem. They present computational results on real data from a major US airline, Delta Airlines. Desaulniers et al. (1997) solve daily aircraft routing and scheduling problem using set partitioning type problem and a time constrained multi-commodity network flow problem. Biggs, Parkhurst & Wilson (2003) describe a global optimization problem for aircraft routing problem and discuss the performance of a number of recently proposed solution algorithms when applied to some demonstration examples. Gabteni & Grönkvist (2009) solve tail assignment problem, which is the problem of assigning flight legs to individual identified aircraft using a hybrid column generation and constraint programming solution approach.

Besides the aircraft routing problems, there are several studies also on aircraft maintenance routing problem. Feo & Bard (1989) develop minimum cost multi-

commodity network model with integral constraint that locates maintenance locations and develops flight schedules that better meet the cyclical demand for maintenance. The computational experiments are made on American Airlines for Boing 727 fleet.

Gopalan & Talluri (1998) study the maintenance routing problem for USAir. They use simple polynomial-time algorithms for finding a routing of aircraft in a graph whose routings during the day are fixed. The algorithm satisfies both the three day maintenance and the balance check visit requirements. They present a small real world example for the fleet of type 767 that consists of 12 planes and 12 "lines of flying" s each day. Talluri (1998) try to generate four days of maintenance routing by extending the three day problem of Gopalan and Talluri (1998).

Sarac, Rajan & Rump (2006) develop an operational aircraft maintenance routing problem formulation that includes maintenance resource availability constraints. They model the problem as connection based network flow problem and use a set-partitioning based formulation in which decision variables represent feasible routes for aircraft. They propose a branch-and-price algorithm for solving the problem.

2.1.3.1 Impacts and Challenges of Aircraft Routing Problem

Barnhart et al. (1998) solved aircraft-routing problems of major U.S. airlines, containing up to 200 flight legs and over 400 million string variables and obtain feasible solutions. But for aircraft maintenance problems, the feasibility is not always guaranteed, because of the sequential approach of first solving the fleet assignment problem and then solving the resulting aircraft-routing problems.

Solving the fleet assignment problem first and then the resulting aircraft routing problems can lead to violations of aircraft maintenance requirements. To guarantee feasible solutions, particularly in low frequency point-to-point networks, researchers have integrated fleet assignment and aircraft routing models, as in Desaulniers et al. (1997). Thus, they could develop aircraft routing model to include multiple fleet types, instead of a single aircraft type.

2.1.4 The Literature Review on Crew Scheduling Problems

There are a lot of studies published on airline crew scheduling. The researchers have given special attention to airline crew scheduling problems due to complex and challenging nature of the problem. The Table 2.2 summarizes the researches made in the crew scheduling literature.

The very first work on crew scheduling problems was found in by Arabeyre, Fearnley, Steiger & Teather (1969). They survey the older works on crew scheduling and different approaches that crew scheduled problem has been solved by.

Because of the immense size of airline crew scheduling problems, i.e., thousands of constraints and billions of variables, researchers initially focused on heuristic methods to achieve solutions. Many of these heuristics generate solutions by considering only a relatively small number of pairings.

The oldest known scheduling method required that the schedulers manually produce an initial solution of disjoint pairings. In the early 1970s a program called "TPACS" was developed by IBM (Rubin, 1973). Gershkoff (1989) developed the TRIP algorithm which was the generalization of TPACS algorithm.

Graves, McBride, Gershkoff, Anderson & Mahidhara (1993) developed a new crew scheduling optimization system for United Airlines. This new system permits quick response to schedule changes and is able to reduce crew scheduling cost.

Hoffman & Padberg (1993) present a branch-and-cut approach to solve large set partitioning problems arising in the airline industry. The branch-and-cut solver generates cutting planes and incorporates these cuts into a tree- search algorithm that uses automatic reformulation procedures, heuristics and linear programming technology to assist in the solution.

Although these approaches produce very improved solutions, the quality of their solutions can not be quantified relative to an optimal solution.

Table 2.2 The summary of the literature review on crew scheduling problems

Year	Author	Basic /Integrated	Modelling approach					Integrated with				Solution approach							Application (3)
			Set covering	Set partitioning	Network Flow Model	Integer Linear Prog.	Mix Integer Prog.	Flight Schedules	Fleet Assignment	Crew Rostering	Aircraft Routing	Heuristic algorithms	Column generation	LP Relaxation	Network flow based	Benders Decomposition	Metaheuristics (1)	Others (2)	
1973	Rubin	B	✓									✓							R
1985	Ball, Roberts	B		✓														GPA	R
1988	Lavoie, Minoux, Odier	B	✓										✓	✓					R
1989	Gershkoff	B		✓								✓							R
1989	Desrochers, Soumis	B	✓										✓						R
1993	Graves et al.	B		✓								✓							R
1993	Hoffman, Padberg	B		✓								✓		✓					R
1995	Vance, Barnhart, Johnson, Nemhauser	B				✓							✓	✓					R
1997	Chu, Gelman, Johnson	B																GTB	R
1997	Beasley, Caos	B																DP	T
1998	Ioachim, Gelinas, Soumis, Desrosiers	B			✓													DP	-
1998	Ftulis, Giordano, Plüss, Vota	B		✓														ES	R
1998	Barnhart, Shenoi	B			✓										✓				R
2000	Lagerholm, Peterson, Söderberg	B															NN		T
2001	Yan,Tu	B			✓										✓				R
2001	Özdemir, Mohan	B															GA		T
2001	Cordeau, Stojkovic, Soumis, Desrosiers	B					✓				✓		✓			✓			R
2001	Klabjan et al.[46]	B			✓							✓		✓					T
2001	Klabjan et al.[47]	B		✓								✓		✓					T
2002	Yan, Tung, Tu	B				✓							✓	✓					R
2002	Yan, Chang	B		✓									✓						R
2002	Kornilakis, Stamatopoulos	B															GA		R
2002	Dias, Sousa, Cunha	B															GA		R
2002	Klabjan et al.	I		✓							✓			✓					T
2003	Cohn, Barnhart	I									✓		✓						R
2004	Sandhu, Klabjan	I				✓			✓		✓		✓	✓		✓			R
2005	Mercier, Cordeau, Soumis	I					✓				✓		✓			✓			R
2006	Guo, Mellouli, Suhl, Thiel	I			✓					✓					✓				R
2007	Mercier, Soumis	I					✓	✓			✓		✓			✓			R
2010	Weide, Ryan, Ehrgott	I				✓					✓	✓	✓	✓					R

(1) GA: Genetic Algorithm, NN: Neural Networks
(2) DP: Dynamic Programming, ES: Expert Systems, GPA: Graph Partitioning Approach, GTB: Graph Theoretic Branching
(3) R: Real data, T: Test data

With advances in optimization theory and computing, the researchers have built enhanced crew scheduling capabilities and designed optimization based approaches. In the literature mostly, the crew scheduling problem is formulated as a set-partitioning problem or set covering problem and column generation method is used to find feasible pairings which will be the input of the set partitioning or set covering problems.

Lavoie, Minoux & Odier (1988) were the first researchers who propose the column generation approach to crew scheduling problem. Desrochers & Soumis (1989) model the urban transit crew scheduling (which is very similar to airline crew scheduling problem) as set covering type problem and used column generation technique. Vance, Barnhart, Johnson & Nemhauser (1995) present a new formulation (duty-period-based formulation) of the airline crew scheduling problem. They develop a column generation algorithm for solving the LP relaxation of this formulation. Yan, Tung & Tu (2002) develop eight scheduling models to minimize crew costs and to plan the proper individual pairings using real constraints for a Taiwan airline. They consider only the cabin crew members. The model is formulated as an integer linear program and the column-generation-based algorithms are developed to solve them. Yan & Chang (2002) develop a model for cockpit crew scheduling problem. They use set partitioning problem and proposed column generation approach to efficiently solve their problem.

In the literature, some researchers have modeled crew scheduling problems using network modeling approaches and develop various solution approaches to solve the network flow models. Ioachim, Gelinas, Soumis & Desrosiers (1998) model the crew scheduling problem as the shortest path problem with time windows and additional linear costs on the node service start times. Yan & Tu (2001) introduce a pure network model that can both efficiently and effectively solve crew scheduling problem. Guo, Mellouli, Suhl & Thiel (2006) consider the crew scheduling and assignment process for airlines, where crew members are stationed unevenly among home bases. They introduce a time–space based network flow approach for modeling the crew pairing chain problem. They also develop state-expanded multi-commodity aggregated time–space network optimization model.

Metaheuristics such as Genetic Algorithms, Simulated Annealing etc. are the other solution approaches for crew scheduling problems. Özdemir & Mohan (2001), Kornilakis & Stamatopoulos (2002) solve the crew scheduling problem using a genetic algorithm.

Some other heuristic procedures have also been applied to the crew scheduling problems. Chu, Gelman & Johnson (1997) introduce an improved graph theoretic branching heuristic to solve large set partitioning problems representing a collection of best pairings. Lagerholm, Peterson & Söderberg (2000) present a method and explore it within the framework of Potts Neural Networks for solving optimization problems. They choose the airline crew scheduling problem as a target application. Klabjan, Johnson, Nemhauser, Gelman & Ramaswamy (2001) integrate the heuristic consideration of subsets of columns with an optimization-based pricing approach. They produce improved solutions, compared to those achieved with branch-and-price, by initially developing hundreds of millions of columns and applying random selection and pricing techniques to prune the number of columns ultimately considered.

There have also been other different techniques, which have been applied to crew scheduling problems. Beasley & Caos (1997) develop a dynamic programming formulation to find a new lower bound for the crew scheduling problem. Yen & Birge consider the stochastic crew scheduling model rather than deterministic model and they devised the solution methodology for integrating disruptions in the evaluation of crew schedules.

2.1.4.1 Integrated Aircraft Routing and Crew Scheduling Problems

Recent studies in airline planning problems focus on integrated aircraft routing and crew scheduling problems. The most recent literature of these integrated problems as follows:

Cordeau, Stojkovic, Soumis & Desrosiers (2001) use “Benders Decomposition Algorithm” for simultaneous aircraft routing and crew scheduling problem. They use linking constraints, which impose minimum connection times for crews that depend on aircraft connections. The solution process iterates between a master problem that solves the aircraft routing problem, and a subproblem that solves the crew pairing problem.

Klabjan, Johnson, Nemhauser, Gelman & Ramaswamy (2002) address particular integration of schedule planning, aircraft routing and crew scheduling. They provide more flexibility for crew scheduling while maintaining the feasibility of aircraft routing by adding plane-count constraints to crew scheduling problem. They assume that the crew scheduling is solved before aircraft routing and the departure times of flights have not been fixed so that it is possible to move the departure time of a flight as it is within a given time window. They perform computational experiment on 4 fleets, 2 small ones with 100-200 legs and 2 larger ones with 300-450 flight legs. The approach produce very significant savings, but it is less likely to yield a feasible aircraft routing problem in an international context where maintenance does not necessarily take place at night.

Cohn & Barnhart (2003) present an integrated model, but instead of incorporating the aircraft routing formulation in the model, variables representing complete solutions to the aircraft routing problem are used. This reduces the number of constraints, but may lead to a very large number of additional variables. In this extended crew pairing model, an aircraft solution that does not use at least one short connection is not included, since short connections are the only link between the two problems.

Mercier, Cordeau & Soumis (2005) consider the similar method as Cordeau et al. (2001). Their assumptions and considerations on linking constraints are the same as Cordeau et al. (2001). They propose an enhanced model incorporating robustness to handle these linking constraints. They combine Benders decomposition and column generation to solve the problem. They present computational experiments based on data provided by two major airlines containing as many as 700 daily legs. They compare their approach to the method proposed by Cordeau et al. (2001) and the pairing model proposed by Cohn and Barnhart (2003). Their approach finds, in less time, solutions that are both more robust and of better quality.

Mercier & Soumis (2007) solve integrated aircraft routing, crew scheduling and flight retiming problem. They construct a minimum-cost set of aircraft routes and crew pairings while choosing a departure time for each flight leg within a given time

window. Linking constraints ensure that the same schedule is chosen for both the aircraft routes and the crew pairings, and impose minimum connection times for crews that depend on aircraft connections and departure times. The aircraft and the crew paths are generated with time-space networks. They propose a methodology that combines Benders decomposition, column generation and a dynamic constraint generation procedure. Computational experiments performed on test instances provided by two major airlines containing up to 500 daily legs show that allowing some flexibility on the departure times within an integrated model yields significant cost savings while ensuring the feasibility of the resulting aircraft routes and crew pairings.

2.1.4.2 Impacts and Challenges of Airline Crew Scheduling Problem

Most major airlines use optimization tools to partially or fully generate their crew schedules, with significant economic impacts. Already in the 1960s, airlines were solving crew pairing problems to reduce costs and automate their burdensome manual planning process (Arabeyre et al. 1969). With advances in optimization and computing, crew pairing solvers became increasingly sophisticated and applicable. Using advanced heuristic techniques; Anbil et al. (1991) achieved cost savings at American Airlines of $20 million dollars annually, representing an increase in crew utilization of 1.5%

Motivated by the potential for even further cost savings, researchers focused on the development of optimization-based approaches in the decades spanning the 1980s and 1990s. Barnhart et al. (2003) reported that using a branch-and-price approach, one major U.S. airline was able to produce near-optimal crew solutions costing $50 million per year less than the solutions generated by a state-of-the-art heuristic.

In addition to these economic benefits, crew scheduling optimization can be a useful tool in contract negotiations, helping airlines to quantify the impacts of proposed changes in cost structures, benefits, and work rules. The economic impact

of each proposed change can be evaluated by changing selected inputs to the crew scheduling model, and rerunning the optimization procedure.

While significant, the effects of optimization on crew scheduling are limited by the sequential schedule planning process. With the flight schedule, fleet assignment, and aircraft routing decisions fixed, the range of crew scheduling possibilities is limited. To eliminate the negative effects of the sequential solutions and to obtain more flexibility, researchers have developed extended models that incorporate crew considerations into some of the other subproblems solved.

2.1.5 The Literature Review on Crew Rostering Problems

Teodorovic & Lucic (1996) develop a model to solve the aircrew rostering problem consisting of the assignment of crew members to planned rotations. The problem is multi-objective programming problem. They solve the problem using fuzzy control methods, so that it is possible to accommodate possible criteria of the qualitative character. The algorithms are tested on the example of assigning 53 flight captains to 422 rotations to be flown during the subsequent 30 days.

Lucic & Teodorovic (1999) solve the aircrew rostering problem as a multi-objective optimization problem. The aircrew rostering problem is solved within the framework of pilot groups that fly on the same type of aircraft. The developed algorithms were tested on the example of distributing 53 flight captains to 422 rotations that are to be executed during the next 30 days.

Dowling, Krishnamoorthy, Mackenzie & Sier (1997) describe a human resource planning and scheduling system that assists in the rostering of approximately 500 staff for the airport operations of a major international airline at one of the busiest international airports. The system allocates individual tasks to the staff for any particular day, and effectively manages, in real-time, disruptions that occur due to aircraft delays and unplanned staff absences on the day of operations. The rostering system is able to develop an optimized monthly roster for 500 staff in around 5 hours.

Gamache, Soumis, Villeneuve, Desroriers & Gelinas (1998) describe the Preferential Bidding Problem. The problem consists of assigning to crew members pairings, days off, annual leaves, training periods, etc., while considering a set of weighted bids that reflect individual preferences. For each employee, from the most senior to the most junior, a generalized set partitioning problem is solved by column generation embedded in a branch-and-bound tree. The method is applied to 24 instances provided by Air Canada.

Dawid, König & Strauss (2001) introduce an efficient adaptation of the branch-and-bound technique that solves real-world rostering problems for airline crews. They present the SWIFTROSTER algorithm, a recursive implicit enumeration approach that incorporates elements of constraint-programming, i.e. propagation. They compute a sample monthly schedule on the basis of a medium-sized European airline's real data.

Kohl & Karish (2004) give a comprehensive description of real-world airline crew rostering problems and the mathematical models used to capture the various constraints and objectives found in the airline industry. They present commercial software system: The Carmen Crew Rostering system.

Chu (in press) uses goal programming (GP) for an integrated problem of crew duties assignment, for baggage services section staff at the Hong Kong International Airport. The problem is solved via decomposition into its duties generating phase (a GP planner), followed by its GP scheduling and rostering phase. The results can be adopted as a good crew schedule in the sense that it is both feasible, satisfying various work conditions, and ‘‘optimal’’ in minimizing idle shifts.

2.1.5.1 Impacts and Challenges of Crew Rostering Problem

Crew rostering problems are usually considered as the part of crew scheduling problems and solved integrated with crew pairing problems. The considerations about crew scheduling problems are also valid for crew rostering problems.

Furthermore, in crew rostering problems there are multiple objectives due to crewmembers preferences and expectations that have to be satisfied. These objectives make the optimization problem more complicated. In order to deal with various objectives some researchers develop models based on goal programming approach.

Because of the complex nature of the problem and various constraints that have to be satisfied, the researchers generally focus on finding the "satisfying and feasible" solution, rather than the optimized solution. For this purpose different heuristic approaches have been developed in order to solve problems.

On the other hand, with the improvement of the optimization based techniques and computer science, the software tools that captures the airline's requirements and crew member's preferences have been developed in the recent years. The most important and common tool is Carmen systems, which is widely used in most of the airlines in Europe and North America.

2.2 The Literature Review on Airline Recovery Problems

The literature review of airline recovery problems will be explained in detail since this dissertation focuses on this research area.

In the disruptions management literature, the aircraft recovery problem has been received significant attention because it is the easiest resource due to the lowest complexity in rules and the smallest numbers and also because for the airlines the aircraft is the "main" resource which is fixed first and foremost. Another reason is that crews can be repositioned fairly easy and standby crews are often available, the aircraft are seen as the scarce resource.

Crew recovery problems have not taken much attention in the literature. Because the aviation rules to be satisfied is very complex causing the problem becomes more complicated. Also, crews are less costly resources compared to aircrafts and it is easier to find standby/spare crew in case of need.

On the other hand, there has not been much work on integrated recovery problems in the literature and existing works have not reported any real-life results yet. Thus, the integrated recovery problems are still a challenge and are subject to further research.

2.2.1 The Literature Review on Aircraft Recovery Problems

Since the effective utilization of aircraft is very important and the shortage of aircraft causes great losses in the airlines revenue, the researcher have been studied the aircraft recovery problems since many years. The Table 2.3 summarizes the works performed in the aircraft recovery literature.

Teodorovic & Guberinic (1984) were among the first to study the aircraft recovery problem. In their paper, one or more aircraft from an airline fleet are unavailable for technical reasons and airline has to operate on the existing network with reduced number of planes. They discuss the problem of minimizing total passenger delays on an airline network for the schedule perturbation, by reassigning and retiming flights. The problem is solved separately for each fleet.

The model is based on a type of connection network, which consists of two types of nodes. The first type represents the flights to be flown whereas the other represents operational aircraft. They attempt to determine the least expensive set of aircraft routings and schedule plan using a Branch and Bound procedure. Their methodology is based on the assumption that all the aircraft in the fleet have the same capacity, and they only considered a marginally sized fleet of three aircraft operating a total of eight scheduled flights. A numerical example of 8 flights is considered and it illustrates the efficiency of the model.

Table 2.3 The summary of the literature review on aircraft recovery problems

Authors	Year	Network	Recovery Strategies: Cancellation	Delay	Swapping/Ferrying	Crew Considerations	Multiple Fleet	Passenger Recovery	Objective function	Solution Approach
Teodorovic and Guberinic	1984	CN		✓	✓				Min: Total passenger delays	Branch-and Bound
Teodorovic and Stojkovic	1990	CN	✓	✓					Min: Number of cancellations and delay minutes	Lexicographic Dynamic Programming Goal Programming Greedy Heuristics
Jarrah et al.	1993	TLN	✓	✓	✓				Min: Delay, swap and cancellation costs	Busacker-Gowen's Dual Algorithm
Teodorovic and Stojkovic	1995	CN	✓	✓	✓	✓			Min: Total number of cancelled flights Total passenger delays	Lexicographic Dynamic Programming Goal Programming Greedy Heuristics
Mathaisel	1996	TLN	✓	✓	✓				Min: Revenue loss	Out-of-Kilter algorithm
Talluri	1996	CN			✓		✓		Min: Swapping costs	Heuristic algorithm for swapping
Yan and Tu	1996	TLN	✓	✓	✓		✓		Max: Total system profit	Lagrangian relaxation with subgradient methods
Yan and Yang	1996	TLN	✓	✓					Max: Revenue - costs	Lagrangian relaxation with subgradient methods
Cao and Kanafani	1997	TLN	✓	✓	✓				Max: Revenue - costs	Algorithm for 0-1 quadratic programming
Argüello et al.	1997	TBN	✓	✓	✓		✓		Min: rerouting and cancellation costs	GRASP (Greedy Randomized Adaptive Search Procedure)
Lou and Yu	1997	Integer Programming		✓					Min: percentage of flights delayed more than 15 minutes	LP Relaxation
Thengvall et al.	2000	TLN	✓	✓	✓				Max: Revenue - costs	LP Relaxation Rounding Heuristic
Bard et al.	2000	TBN	✓	✓					Min: Delay and cancellation costs	LP Relaxation Branch and Bound
Rosenberger et al.	2003	CN	✓	✓	✓	✓			Min: Rerouting, delay and cancellation costs	Aircraft Selection Heuristic
Andersson and Varbrand	2004	CN	✓	✓	✓		✓		Max: Revenue - costs	Column Generation
Eggenberg et al.	2009	CSN	✓	✓	✓		✓	✓	Min: Operating, cancellation, delay costs Passenger inconvenience costs	Column Generation Dynamic Programming

* CN: connection network, TLN: time line netwok, TBN: time band network, CSN: constraint spesific network

Teodorovic (1985) presents research on the reliability of airline scheduling as it relates to meteorological conditions, the ability to identify an indicator for quantifying the adaptability of such airline schedules to weather conditions, and an overview of a potential solution procedure. The author outlines this heuristic algorithm for minimizing the number of aircraft required to accommodate a given traffic volume, while ensuring that aircrafts are assigned to only one flight within a given time period.

Teodorovic & Stojkovic (1990) consider aircraft shortage and discuss a greedy heuristic algorithm for solving a lexicographic optimization problem which considers aircraft scheduling and routing in a new daily schedule. The main objective is to minimize the total number of cancelled flights. The algorithm developed is based on dynamic programming, and is characterized by a sequential approach to solving the problem as flights are assigned to aircraft in sequences. If there are several airline schedules with an equal total number of cancelled flights, the schedule with the minimum total passenger delay on flights to be performed is chosen. They use a greedy heuristic for solving this goal programming problem. The heuristic processes the aircraft in sequence. First, a shortest path (schedule) for the first aircraft is generated. Nodes used in this shortest path are removed and the shortest path method is invoked again to generate the schedule for the next plane, and so forth. For each aircraft an attempt is made to assign as many flights as possible. The path with the least amount of delays that cover the same number of flights is found using a recursive delay function. The method is tested on a small example of 14 aircraft and 80 flights. The model does not consider the impact of crew scheduling in the aircraft scheduling process.

Teodorovic & Stojkovic (1995) further extend their model to include also crew considerations. The model proposed still solves the problem individually for each aircraft type. Their approach is based on two objectives where the first priority objective is to maximize the total number of flights flown and the second objective is to minimize the total passenger time loss on flights that are not canceled. The proposed framework schedules crew before aircraft. Their heuristic model based on the FIFO principle and a sequential approach based on dynamic programming, is

developed to facilitate and incorporate the work and experience of the dispatcher in the decision process regarding traffic management. The model developed is used to determine the aircraft rotations, as well as the crew rotations, while minimizing the number of cancelled flights.

Jarrah, Yu, Krishnamurthy & Rakshit (1993) present an overview of a decision support framework for airline flight cancellations and delays at United Airlines. Their underlying solution methodology is based on network flow theory. They develop two network flow models which provide solutions in the form of a set of flight delays or a set of flight cancellations. They discuss the two major techniques for solving the aircraft recovery problem: cancellation and re-timing. A timeline network is used to model the problem data and three methods are discussed: The successive shortest path method for cancellations, and two models based on the same type of network and allowing cancellations responding re-timings. Both models allow for swapping aircraft among flights and for using spare aircraft. The time-line network has two types of nodes per station - flight nodes and aircraft nodes. These are used to model the aircraft-to-flight assignments. Both models are minimum cost flow models. In the objective of both models is to minimize the costs associated with the recovery, i.e. the swap and ferry costs in both models and in addition the delay costs in the delay model and the cancellation costs in the cancellation model. The models assume that a disutility can be assigned to each flight in order to reflect the lost revenue if the flight is cancelled, and that the disutility of delaying each flight is assessable. Both models are solved using Busacker-Gowen's dual algorithm. The shortest path is solved repeatedly to achieve the necessary flow in the network. The network models presented are solved independently of each other, and do not take into consideration crew and aircraft maintenance constraints. This solution framework is deficient in that it does not allow for a trade-off between cancelling and delaying a given flight in a single decision process. In addition, the solution methodology does not allow for potential substitution of aircraft with varying capacity, and operational capabilities. The models are tested on a network with three airports each having considerable air traffic. The re-timing model can typically save part of the delay minutes and produce a substantially better solution with respect to

cost. Both minor and major disruptions are tested in the test scenarios. The results from the cancellation model are not as easy to interpret. The three test scenarios here are based on United Airlines' B737 fleet and a regional subdivision of the United States.

Mathaisel (1996) reports on the development of a decision support system for AOCC (Airline Operations Control Centers) which integrates computer science and operations research techniques. The application integrates real-time flight following, aircraft routing, maintenance, crew management, gate assignment and flight planning with dynamic aircraft rescheduling and fleet rerouting algorithms for irregular operations. As discussed by the author, the algorithms help airline controllers optimally reroute aircraft, crews and passengers when operational problems disrupt the execution of the schedule plan. The system includes a real-time, interactive, graphical aircraft routing displays; a rule system, which provides warnings of constraint violations and usual conditions; and the ability to generate what-if solution scenarios. The integrated system is demonstrated by simulating a disruption to a planned schedule and by using one of the available tools, a network flow algorithm, to determine optimal rerouting alternatives. Furthermore, the paper discusses how a simple network flow problem can be used for modeling the aircraft recovery problem. First, the non-disruptive network is constructed. Here, all planned aircraft routings are represented by setting the upper and lower bounds of the binary flow variables to "1". The network is then altered in order to describe the disruptive situation. The author discusses several types of disruptions that must be taken into account when designing the algorithm that alters the network. These include ground delays, in flight delays, cancellations, station closure and diversions. The altered network is an expanded version of the non-disruptive network usually consisting of a larger number of arcs and in some cases also additional nodes. The lower bounds of the flow variables are reset to "0" and the resulting problem is solved by the Out-of-Kilter algorithm. The model is capable of using cancellation as well as retiming. However, the paper does not discuss multiple types of aircraft, crew considerations, or solution time.

Talluri (1996) deals with the problem of changing the aircraft type for a single flight while still satisfying all the constraints. The problem is to change the aircraft type on a specific flight at a minimum cost. As this process is done at operations control after the original planning phase computation speed is an important issue, and a solution must be found within 2 minutes. Solutions are categorized with respect to the number of overnight swaps needed. Being able to make the change without affecting the overnight position of an aircraft is desirable mainly due to maintenance. They give a simple algorithm for making this swap that will not affect the equipment type composition of aircraft overnighting at the various stations. They describe two further applications of the swapping procedure in the airline schedule development process. The first algorithm determines –in polynomial time- whether an aircraft type swap and a swap-back can be accomplished before the overnight stop. If the algorithm can not find a solution another algorithm allowing at most k overnight changes is invoked for ever increasing k. Maintenance and crew considerations are not implemented. Testing is very limited and only documented by a single instance. For a connection network of two equipment types, 700 arcs and 200 nodes ten swapping solutions was found.

Yan & Tu (1996) develop a framework to assist carriers in fleet routing and flight scheduling for schedule perturbations in the operations of multi-fleet and multi-stop flights. The framework is based on a basic multi-fleet schedule perturbation model constructed as multiple commodity network flow problems. They use a time-line network, in which flights are represented by edges from origin to destination. The network is modified with a supply node added at the point in time and space, where the absent aircraft is recovered. One such network is built for each fleet in question. Swapping between fleets is made possible. To allow for re-timing, edges “parallel” with the flight edges are included with specific time intervals. Lagrangian relaxation with subgradient methods accompanied by the network simplex method, a Lagrangian heuristic and a modified subgradient method are developed to solve the problems. Results are provided based on data from China Airlines with 24 stations, 273 flights and 3 types of aircraft. Several types of recovery strategies are tested including limited re-timing, positioning, and the modification of multi-stop flights.

Yan & Yang (1996) develop a decision support framework for handling schedule perturbations which incorporates concepts published by United Airlines. The framework is based on a basic schedule perturbation model constructed as a dynamic network (time-space network) from which several perturbed network models are established for scheduling following irregularities. They formulate both pure network flow problems, which are solved using a network simplex algorithm, and network flow problem with side constraints, which are solved using Lagrangian relaxation with subgradient methods. They outline the basic schedule perturbation model, which is designed to minimize the schedule-perturbed period after an incident, while maximizing profitability. In addition, they consider the effects of flight cancellations, flight delays and ferry flights as solution alternatives in the decision process. The framework is designed to aid airlines in handling schedule perturbations caused by aircraft breakdowns, and assumes scenarios with only one broken down aircraft and a single fleet type. In addition, the models do not incorporate aircraft maintenance and crew constraints in the formulation.

Argüello, Bard & Yu (1997) present a method based on the metaheuristic GRASP (Greedy Randomized Adaptive Search Procedure) to reschedule the aircraft routing during an aircraft shortage. The heuristic is based on randomized neighborhood search. The heuristic is capable of canceling, retiming and swapping the flights. Maintenance and crew issues are not considered. An initial solution to the problem consists of aircraft routes and cancellation routes (sequences of flights operated by an individual aircraft, and sequences of canceled flights, which could be operated by an individual aircraft). In each step of the solution process, all pairs of two routes (of which at least one must be an aircraft route) from the current solution are investigated. For each such pair, all sets of feasible re-routes covering the flights from the two routes are constructed respecting flight coverage and aircraft balance at stations. Each set of feasible re-routes is assigned a score reflecting the cancellation cost and delay cost of the route set. A limited number of these are stored in a restricted candidate list. The selection is based either on quality relative to the current solution or on absolute quality. Finally, a random member of the candidate list is chosen as the starting solution for the next step of the algorithm. The goal is to

produce a recovery schedule in order to restore the original schedule by the following day. The cost to be minimized includes measures of passenger inconvenience and lost flight revenue. The method is tested on B757 fleet data from Continental Airlines with 16 aircraft and 42 flights. The recovery period is set to one day. All instances grounding from 1 to 5 out of the 16 aircraft at the beginning of the day are investigated. The results obtained by the proposed method are clearly superior to just canceling the flights serviced by the grounded aircraft. In more than 70% of the instances, the GRASP solution is within 5% of optimality.

Lou & Yu (1997) address the airline schedule perturbation problem caused by the Ground Delay Program of the Federal Aviation Authorities. The goal is to improve airline dependability statistics defined by Department of Transportation as percentage of flights delayed more than 15 minutes. They design the polynomial algorithm for minimizing maximum delay among out flights. The problem is modeled as an integer program. To solve the model, they derive valid inequalities for strengthening LP relaxation bound.

Cao & Kanafani (1997a) discuss a real-time decision support tool for the integration of airline flight cancellations and delays. This research is an extension of the work of Jarrah et al. (1993), using many of the modeling concepts presented and discussed in Jarrah's paper. The authors present a quadratic 0-1 programming model for the integrated decision problem, which maximizes operating profit while taking into consideration both delay costs and penalties for flight cancellations. They discuss special properties of the Flight Operations Decision Problem (FODP) model which are exploited to develop a specialized algorithm to solve the problem in real-time. The model considers the airport network as a complete system, and traces the effect of delay and aircraft reassignment from one station to the next. The authors consider as an extension to their base model, issues of ferrying surplus aircraft and multiple aircraft type swapping capabilities. In a subsequent article, Cao & Kanafani (1997b) present an effective algorithm to solve the FODP model and discuss computational experiments with a continuous mathematical problem, derived from the 0-1 quadratic problem. In the case studies presented, aircraft ferrying, crew scheduling and airport capacity constraints are ignored in the solution procedure.

Thengvall, Bard & Yu (2000) present a network model with side constraints in which delays and cancellations are used to deal with aircraft shortages while ensuring a significant portion of the original aircraft routings remain intact. The model solves the aircraft recovery problem for a single fleet with the objective of maximizing the passenger revenues subtracting the costs of retimings. In the model, which handles cancellations, retimings and flight swaps, crew and maintenance issues are not considered. The model is an integer single-commodity network flow problem with side constraints, which is solved by standard optimization software. As a supplement, a heuristic to construct an integral solution from a fractional LP-solution is implemented. The approach is tested on real life data from Continental Airlines (B757 schedule with 16 aircraft and 13 stations, and B737-100 with 27 aircraft and 30 stations) and optimal or near-optimal solutions are obtained from the LP relaxation of network formulation. When integer solutions can not be obtained, a rounding heuristic is provided in order to find feasible solutions within a small fraction of optimum.

Bard, Yu & Argüello (2000) present the time-band optimization model for reconstructing aircraft routings in response to groundings and delays experienced in daily operations, where the objective is to minimize the costs of flight delays and cancellations. This model is constructed by transforming the aircraft routing problem into a time-based network in which the time horizon is discretized. Thus, the resulting formulation is an integral minimum cost network flow problem with side constraints ensuring that a flight is either canceled or flown by a unique aircraft. The authors develop a solution methodology with an initialization step, in which the flight schedule is input and the time bands decided followed by the generation of the time band network. Finally, the mathematical formulation of the integer programming problem is derived. This problem is then relaxed by ignoring the integrality constraints and solved to optimality. The authors outline conditions in which exact solutions are attainable, and discuss the complexity of the problem relative to the size of the underlying airline network. In addition, they present computational results for a marginally sized case study of a single fleet (on Continental Airlines B737-100) of 27 similar aircraft, serving a network of 30

stations with 162 flights. 427 test cases are reported: 27, in which one aircraft is grounded, and 100 cases for each case of two, three, four and five aircraft grounded.

Rosenberger, Johnson & Nemhauser (2003) propose a model which addresses each aircraft type as a single problem. The model principally follows an approach traditionally used in planning problems, namely a Set Partitioning master problem and a route generating procedure. The objective is to minimize the cost of cancellation and retimings, and it is the responsibility of the controllers to define the parameters accordingly. The master problem determines which legs to delay or cancelled. In order to solve the master problem, they develop a heuristic for selecting only a subset of aircraft to be involved in the Set Partition problem. The heuristic determines for each disrupted aircraft a number of other aircraft with routes allowing a swap with the disrupted aircraft. The legs of these routes are those included in the route generation procedure. The paper contains an impressive testing using SimAir simulating 500 days of operations for the three fleets ranging in size from 32 to 96 aircraft servicing 139-407 flights. Finally they revise the model to minimize crew and passenger disruptions.

Andersson & Varbrand (2004) solve the complex problem of reconstructing aircraft schedules. A mixed integer multi-commodity flow model with side constraints, that each aircraft is a commodity, is developed. Side constraints are also used to model possible delays. The model is then reformulated into a set packing model using the Dantzig–Wolfe decomposition. Cancellations, delays and aircraft swaps are used to resolve the perturbation and the model ensures that the schedule returns to normal within a certain time. Two column generation schemes for heuristically solving the model are tested on real problem data obtained from a Swedish domestic airline.

The most recent study on aircraft recovery problems has been performed by Eggenberg et al. (2009). They develop the constraint-specific recovery network model which can be seen as an extension of the time-band model by Bard et al. (2000). With this network model structural and unit- specific constraints are

separated and checked independently. They have applied the model to the aircraft recovery problem with maintenance planning and passenger recovery problem.

2.2.2 The Literature Review on Crew Recovery Problems

Since the problem complexity of the aircrew recovery problems, these problems have not been studied as much as aircraft recovery problems. Most of the research for these problems has been made in the recent years. The Table 2.4 summarizes the work in the aircrew recovery literature.

Wei, Yu & Song (1997) develop an integer programming model and an algorithm for managing crew in case of disruption. The model repairs broken pairings and assigns crew to flights that are not covered. The objective is to return the entire system to the original schedule as soon as possible and in the cheapest way. The problem is solved based on a space-time network, which is considered for a certain time window. Start of the window is the current time and the end of the window is the end of the recovery period by which the resources should have returned to their original schedule.

The model is integer multi-commodity network flow problem, where each crew member represents a commodity. Each airport is described by two columns of nodes. The first column represents crew that has arrived to the airport from another flight or that are signing in here. They are placed according to when they are ready. Flight nodes in the second column represent the departure of flights. Reserve nodes (placed in the first column) represent the availability of stand-by crew. Return nodes force the crew to return to their original schedule at the end of the window. There are four types of arcs in the network:

- flight arcs represent the flight from one airport to another,
- stand-by arcs emanates from stand-by crew nodes to those flights at the same airport that can be served by standby crew,
- scheduled arcs emanate from crew nodes to the originally scheduled flight nodes,

- return arcs represent the returning of crew to their original schedule.

The cost of the arcs reflects preferences or penalties. It is assumed that each crew member can be associated with only one fleet type. Furthermore several crew members are grouped and rostered together, i.e. they have the same roster. The network basically defines the feasible pairings. A generalized Set Covering problem with the constraints of covering all flights is then solved. Solution time is restricted to only a few minutes and the operations controller needs multiple good solutions. The solution method is a depth first Branch-and-Bound algorithm. At each node, the problem is defined by the set of (still) uncovered flights and a list of pairings that are modified. When the set of uncovered flights is empty the corresponding Branch-and-Bound node represents a feasible solution to the CPR problem. At each non-leaf node in the search tree a flight is selected among the set of uncovered flights. A candidate crew list is built and the best crew member is chosen. The change pairing of the crew member must satisfy:

- it must be possible to return the crew member to the return node,
- the pairing must be legal,
- the new pairing should be as close to the original one as possible.

The stopping criterion of the algorithm is that a predetermined time limit has been reached or a required number of solutions have been produced. Computational experiments are based on data from an unknown source. The largest instance comprises 6 airports, 51 flights in a two-day period, and 18 pairings. This rather small instance is the basis of 8 scenarios with a different number of delays and cancellations. The running times range from a fraction of a second to 6 seconds producing from 1 to 3 solutions.

Table 2.4 The summary of the literature review on crew recovery problems

Authors	Year	Models	Uncovered flights	Flight delays	Deadheading	Stand-by /Reserve crew	Modifications to schedule	Objective function	Solution Approach
Wei et al.	1997	Integer Multicommodity Network Flow problem	✓		✓	✓		Min: the number of uncovered flights	Depth-First Branch-and-Bound
Stojkovic et al.	1998	Set Partitioning Problem	✓		✓		✓	Min: pairing, deadheading, undercovering costs the disturbances of crew members	Column Generation Branch-and-Bound
Lettovsky et al.	2000	Set Covering Problem	✓				✓	Min: crew reassignment and cancellation costs	Column Generation Branch-and-Bound
Stojkovic et al.	2002	PERT/CPM (time constrained models) (dual: Network Problem)	✓	✓			✓	Min: modifications, uncovered flights, delay costs	Dual Problem
Medard and Sawhney	2003	Set Covering Problem	✓				✓	Min: illegal crew, uncovered flights, affected crew costs	Simple Tree Search Column Generation
Abdelhany et al.	2004	Mixed Integer Program		✓	✓	✓	✓	Min: deadheading, standby, swap, flight delay costs	Optimization Tool
Nissen and Haase	2006	Duty-period-based Network Model					✓	Min: the costs of changing each crew's original schedule	Branch-and Price

Stojkovic, Soumis & Desrosiers (1998) describe the operational airline crew scheduling problem. The problem consists of modifying personalized planned monthly assignments of airline crew members during day-to-day operation. The problem requires that all flights are covered at a minimum cost while minimizing the disturbances of crew members. To generate modified pairings for selected crew members, both the classical crew pairing problem and crew assignment problem is treated simultaneously. They formulate the crew recovery problem as an integer non-linear multi-commodity flow problem. The idea is that in the disrupted period, the duties of the crew are dissolved in order to make re-planning feasible. In their model a task corresponds to one crew member requirement for one flight segment. A flight segment which must be covered by x crew members is considered as x different tasks. During the planning phase, they first generate a set of non-personalized pairings. These pairings are then used to construct monthly plans for specific crew members. Since the operational airline crew scheduling problem requires the immediate production of pairings for specific crew members, they have to construct personalized crew pairings. To generate pairings for specific crew members they use an approach which simultaneously solves both the crew pairing construction problem and personalized monthly blocks construction problem.

They describe the solution of the integer non-linear formulation using a generalization of the Danzig-Wolfe decomposition embedded in a branch and bound solution tree. The master problem is defined as Set Partitioning type model which is solved by Branch-and-Bound with LP-relaxation and column generation. The subproblems (column generators) are modeled as shortest path problems based on a time line network, in which duties are represented as edges between the start node and the end node of the duty. A separate graph is constructed for each crew member. The model and method is tested on data from a major U.S. carrier, and only cockpit personnel for one fleet positioned at the carrier's base is considered. The disruptions considered consist of three delayed aircraft, and one indisposed crew member away from base. Scenarios with one or two crew members per aircraft are tested, and both one day horizons and seven days horizons are tested. The problems are solved using the GENCOL optimizer which employs a column generation procedure within

branch-and-bound search tree to obtain integer solutions. The results show that all test problems have been solved in reasonable times ranging from a few seconds to 20 minutes, depending on the number of subgraphs and number of active tasks considered in the problem. If the solution times of larger problems increase significantly, some additional acceleration strategies should be introduced to reduce them.

Lettovsky, Johnson & Nemhauser (2000) present a method based on an integer programming formulation. They develop a new solution framework. It provides, in almost real time, a recovery plan for reassigning crews to restore a disrupted crew schedule. Preprocessing techniques are applied to extract a subset of the schedule for rescheduling. A fast crew pairing generator is built that enumerates feasible continuations of partially flown crew trips. Deadheads can be given a priori. The crew recovery problem is modeled as a generalized Set Covering problem using 3 different branching strategies (branching on cancellation variables, branching on deadheading variables and branching on follow-ons) and incorporating variable fixing. A test of the method on a schedule of 1296 flight legs from a major U.S. carrier is presented. The legs are covered by 177 pairings originating from 2 crew bases. Three scenarios of irregular operations with different levels of disruptions are set up:

- scenario 1 is a small-size maintenance-related disruption commonly seen in day-to-day operations with limited impact on the airline's operational schedule.
- scenario 2 represents a weather disruption implying decreased landing capacity at an airport
- scenario 3 presents a major disruption having three airports hit by a snowstorm.

For each scenario, they include only 3 potential crews for swaps for each missed connection and kept connecting flights as one flight segment. All scenarios are solved with no flight leg connections protected and without increasing the number of crews included for potential swaps. The maximum number of potential swaps is increased to 10 for each missed connection. The first two scenarios required no branching. Solution times range from 1 to 115 seconds. For the longest solution time

4642 new pairings were generated. In scenario 1 no flight had to be canceled whereas scenario 2 and 3 resulted in up to 21 cancellations. All problems use CPLEX 4.0 as the MIP solver. The results show that medium-sized disruptions (when the number of pairings is less than 15000) to the crew schedule can be handled within an acceptable running time. Further research is required to handle large scale of disruptions. It should be noted that "crew members" are aggregated into groups of people that can not be split for the length of the pairing. This situation is seldom found among European airlines.

Stojkovic, Soumis, Desrosiers & Solomon (2002) present a model that involves determining appropriate real-time changes to planned airline schedules when perturbations occur to minimize customer inconvenience and costs to the airline. They propose a model that determines new flight schedules based on planned crew transfers, rest periods, passenger connections and maintenance. The costs of time reductions, elements of the crew costs and passenger inconvenience are included in the objective function. Their model represents an extension of simpler time-based models introduced such as PERT/CPM. They consider not only activity start times but also activity durations as variable. The dual reformulation of the model is a network problem that can be solved in time linear in problem size. The network consists of a set of origin-destination nodes, where every pair is linked by a directed arc to represent each flight leg. Additional directed arcs are then used to represent other aircraft movement for maintenance and ground service, crew movements for transfer between aircraft and rest and passenger connections. They also define reverse arc set so as to impose limits on aircraft and crew pairings length. In order to validate their model, they generated 10 problems having linear cost functions on arcs and nodes. The problems were solved using the CPLEX Linear Optimizer 3.0.

Medard & Sawhney (2003) consider the crew recovery problem and integrate both crew pairing and crew rostering to solve time critical crew recovery problems arising on the day of operations. The problem is structurally different from the crew pairing and rostering problems because contrary to the planning phase these two subproblems have to be solved at the same time in the recovery phase. This means that both rules on the pairing and the rostering level have to be respected. So they

integrate pairing and rostering models in planning and apply a generate and optimized technique that incorporates both pairing construction and pairing assignment in a single step. They consider the so-called rostering time window decomposition technique. Within the recovery window, flights are de-assigned from the disrupted crew and few other crew, referred to as helper crew. Also, newly scheduled flights may be added to the problem. Some crew members might have pre-assignments within the recovery window, like days off or training duties, which can not be changed. The researchers propose an optimization model, which is the flight-based equivalent to the original pairing-based rostering model, where the de-assigned flights replace the pairings. The optimization model is formulated as a Set Covering model and solution techniques based on simple tree search and more sophisticated column generation and shortest path algorithm are provided. The columns are generated by finding shortest paths either by the use of a Depth First Search strategy (DFS) or by a reduced cost column generator (COLGEN). They test their methods on small to medium sized scenarios ranging from 14 to 885 planned crew members with up to 77 illegal rosters. The solution quality is measured by:

- number of illegal crew remaining in the solution
- the number of remaining open time crew position
- the number of affected crew

The computation times range from 12 to 840 seconds on a 1 GHz PC. The results are encouraging however for some of the more complicated scenarios the time limit of a few minutes is not respected and the authors conclude that the column generation schemes must be refined. This can for instance be obtained by applying more crew specific information in the generation scheme.

Abdelhany, Ekollu & Narasimhan (2004) present a decision support tool that automates crew recovery during irregular operations for large scale commercial airlines. The paper discusses the effect of current disruption management practice on crew problems in detail and subdivides these problems into four categories: Misconnect problems, rest problems, duty problems, and unassigned problems. Based on detailed information regarding the current plan and pool of problems, the

recovery problem is then solved in steps. In the solution method, delaying, using stranded crew, swapping, deadheading, and using standby crew are used as means of recovery. The tool recovers projected crew problems that arise due to current system disruptions, a head of time before their occurrence. It gives a wide flexibility to react to different operation scenarios. It solves for the most efficient crew recovery plan with the least deviation from the originally planned schedule. The approach adopted in the crew recovery tool consists of 4 main steps as data input, preprocessing, optimization solver and post-processing. Their optimization model is an assignment model with side constraints, which takes into account timings and bounds on regarding the use of different means (as e.g. use of undisrupted crew members in the recovery solution). Due to the stepwise approach, the proposed solution is suboptimal. Computational results are reported for a situation from a US carrier with 18 crew problems. The solution involves 121 crew members and is found in less than 2 minutes using CPLEX to solve the mathematical model. From the information given it is hard to judge the practical applicability of the proposed method.

Nissen & Haase (2006) present a new duty-period-based formulation for airline crew rescheduling problem where the aim is to determine new crew assignments minimizing the impact on the original crew schedule, after a disturbance in the schedule. It uses new types of resource constraints to efficiently cover the various labor regulations. Their model is used for scheduling on the crew as well as the crew member level. Since the problem sizes for the model are too large to be solved directly, a column generation approach was chosen. Applying Danzig- Wolfe decomposition to the original model, they get a column generation formulation. The master problem is modeled as set covering type problem and the sub problem is a resource-constrained shortest path problem, which can be solved to optimality using dynamic programming. Since the time needed to obtain a solution is a critical factor in airline irregular operations, they did not solve the column generation model directly with a standard MIP solver, but instead embedded it in a problem specific solution method, which can deliver optimal solutions within a short time. They tested the approach using the data of the dated flight schedule of a major European carrier. They used the flight schedule to generate two crew schedules: one for short-haul and

one for medium-haul flights. Both have a hub-and-spoke structure with one hub and cover a period of 7 days. The short-haul schedule has 8 routes and covers 450 flights from one fleet and the medium-haul has 35 routes and covers 927 flights from one fleet. Then they defined 14 rescheduling scenarios to test the branch—and- price based solution method. Results show that the solution method is capable of providing solutions within the short period of time available to a reschedule after a disturbance occurs.

2.2.3 The Literature Review on Integrated Recovery Problems

Because of the complex nature of the integrated aircraft and crew recovery problem, only a few researches have been made on integrated problems.

Lettovsky's Ph.D. thesis (1997) is the first to consider truly integrated approach in the literature although only a part of it is implemented. His thesis presents a linear mixed-integer mathematical problem that maximizes total profit of the airline while capturing availability of aircrafts, crews and passengers. The mixed-integer programming model is very large and computational intractable, thus Lettowsky suggests solving problem using decomposition algorithms. The formulation has three parts corresponding to each of the resources, that is, crew assignment, aircraft routing and passenger flow. In decomposition scheme these three parts are "controlled" by a master problem denoted the Schedule Recovery Model. The Schedule Recovery Model (SRM) determines a plan for cancellations and delays that satisfy some of the constraints. The solution algorithm applies Benders' decomposition to a mixed-integer programming formulation of the master problem. Three subproblems are formulated in the thesis:

- ARM: The Aircraft Recovery Model. New flight assignments are generated that satisfy maintenance requirements.
- CRM: The Crew Recovery Model. New crew schedules complying with union and governmental regulations are created.
- PFM: The Passenger Flow Model. New passenger itineraries are generated which comply to seat availability for each flight.

In the thesis only the CRM is implemented and tested. The method appears effective.

Furthermore, Bratu & Barnhart (2006) present two models that considers aircraft and crew recovery and through the objective function focuses on passenger recovery. While reserve crews are included into the models they do not consider how to recover disrupted crews. They present two models: passenger delay metric (PDM) and disrupted passenger metric (DPM). Both have same objective function which incorporates operation costs and passenger recovery costs. They test both models on the data involving 302 aircrafts, 83869 passengers on 9925 itinaries a day, 74 airports and 3 hubs. They conclude that only the disrupted passenger metric model is fast enough to be used in a real-time environment.

One of the recent studies on integrated recovery problems has been performed by Abdelghany, Abdelghany & Ekollu. (2008). They present a decision support tool which integrates schedule simulation model and a resource assignment optimization model. The simulation model predicts the list of disrupted flights in the system. The optimization model is formulated as mixed integer programming and combines different recovery actions to minimize projected flight delays and cancellations. Although the developed tool is capable to generate efficient recovery plan, several extensions are still possible.

The most recent study on integrated aircraft and passenger recovery problem has been performed by Jafari and Zegordi (in press). They develop a model to recover flight, aircraft and passenger simultaneously. Their recovery scope is using aircraft rotations and passengers' itineraries instead of flights which helps limiting the disruption scope and is useful to return original schedule as soon as possible. However their integrated model does not consider crew recovery.

2.2.4 Impacts and Challenges of Airline Recovery Problem

The field of disruption management in the airline industry has been increasingly active over the last decade, and in the last years also commercial tools for disruption

management have become available. With the improvement of optimization technology and computer systems, computational speed is greatly increased, which allows solving the recovery problems in real time.

However, airline recovery problems solved in the literature do not really satisfy the airline's requirements. In case of disruption, the airlines need to re-organize their resources, both aircrafts and crews. All these decisions should be made in real time, considering all aircrafts, crews and also passengers. In the airline recovery literature, the researchers generally address single resource problem; either aircraft recovery or crew recovery problem, usually without considering the passengers. The major part of the literature focus on aircraft recovery problem since the aircrafts are the most expensive resource and the crew related problems are much more complicated. In the aircraft recovery problems, generally crew and maintenance considerations are ignored.

The general assumption in the aircraft recovery problem literature is that the crew members are available at any time. Most of the studies do not include the crew availability constraints in the aircraft recovery problems. As seen in Table 2.3, only a few studies involve the crew related considerations in the problems. However, these problems which involve crew considerations do not attempt to recover disrupted crew. Similarly crew recovery problems do not consider the aircraft availability. They only focus on recovering disrupted crew without considering other airline resources. However, in order to operate any flight leg there should be at least one aircraft and one crew to cover that flight leg. Thus aircraft and crew availability should be considered together in the recovery problems. Otherwise the resulting recovery problems can not satisfy the airlines' requirements at all.

On the other hand, the integrated problems consider both aircraft and crew availabilities. However, these problems become very complex due to large number of the constraints and variables coming from the integrated problem. The integrated problem is much harder to be solved. They require special attention and complex techniques in order to be solved to optimality in real time. Instead of developing an integrated models, iterative or sequential solution algorithms may be developed

which includes both crew and aircraft recovery problems considering both resources availability and their correlation. Such algorithm may find both crew and aircraft recovery solutions without dealing with the complexity of the integrated problem. Thus, this dissertation will propose a new sequential and iterative algorithm, which will solve aircraft and crew recovery problems together considering both crews and aircrafts availability and the relation between them without integrating the aircraft and crew related constraints into a big and unmanageable sized model. Such algorithm will obtain solutions for both aircraft and crew recovery problems without dealing with the complexity and huge size resulting from integrated problem.

Although a lot of work exists in the airline recovery literature, the underlying graph models in the most of the paper are more or less identical. The resulting mathematical programs are in most cases multi-commodity network flow problems with side constraints. The researchers generally use this approach to model the problems and develop various solution approaches in order to obtain feasible and optimum (or near-optimum) solutions. With the improvement of optimization technology and computer systems, new solution approaches may be developed in order to solve airline recovery problems in real-time with less effort using more computer based technology. Different from the traditional multi-commodity network flow models, this dissertation will also present new solution algorithms for airline recovery problems using constraint programming technique.

CHAPTER THREE
OVERVIEW OF THE METHODOLOGY EMPLOYED IN THE DEVELOPED MODELS

The airline recovery problems are challenging due to various complex constraints that have to be satisfied and need of real-time solutions. As explained earlier in Chapter 2, in the literature several models have been developed for aircraft and crew recovery problems using different methods including network flow models, mixed integer programming models, set covering and set partitioning type problems or heuristics.

In this dissertation, several models and the solution algorithms are developed for both aircraft and crew recovery problems and for their integration using two different techniques:

- The first technique is multi-commodity network flow model, which is widely used solution approach for modeling and solving aircraft recovery problems in the literature.
- The second technique is constraint programming, which is a new solution approach for modeling and solving airline recovery problems. To best of our knowledge, there is no published work on the application of constraint programming technique to the airline recovery problems in the literature.

This chapter will give brief overview of the above mentioned two techniques.

3.1 Brief Overview of Multi-Commodity Network Flow Problems

The multi-commodity network flow problem is defined over a network where more than one commodity needs to be shipped from specific origin nodes to destination nodes while not violating the capacity constraints associated with the arcs. It extends the single commodity network flow (SCNF) problem in a sense that if the bundle constraints, the arc capacity constraints those tie together flows of

different commodities passing through the same arc are disregarded, a MCNF problem can be viewed as several independent SCNF problems.

Multi-commodity flow problems arise when several commodities use the same underlying network. In many applications, several physical commodities, vehicles, or messages, each governed by their own network flow constraints, share the same network. The commodities may either be differentiated by their physical characteristics or simply by their origin–destination pairs. Different commodities have different origins and destinations, and commodities have separate mass balance constraints at each node. The sharing of common arc capacities binds the different commodities together.

The essential issue addressed by the multi-commodity flow problem is the allocation of the capacity of each arc to the individual commodities in a way that minimizes overall flow costs.

3.1.1 Mathematical Formulation of Multi-Commodity Network Flow Models

The objective of the multi-commodity flow problem is to determine a least cost way to move all commodities through the network in order to satisfy demands at each node subject to shared capacity constraints.

$G = (N, A)$ is a directed network defined by a set N of *n nodes* and a set A of *m directed arcs* which there will flow k different commodities. Each arc $(i, j) \in A$ has an associated *cost* c_{ij}^k that denotes the cost per unit flow on that arc for commodity k. Each arc $(i, j) \in A$ has a *capacity* u_{ij} that denotes the maximum amount that can flow on the arc and a *lower bound* l_{ij} that denotes the minimum amount that must flow on the arc.

Each node $i \in N$ for commodity $k \in K$ has an integer number b_i^k representing its supply or demand. If $b_i^k > 0$, node i is a *supply node*; if $b_i^k < 0$, it is *demand node* with a demand of $-b_i^k$; and if $b_i^k = 0$, it is a *transshipment node*. The decision

variables in the minimum cost flow problem are arc flows and they are represented by X_{ij}^{k}.

The notation used in the mathematical formulation of multi-commodity network flow problem is as follows:

N : set of nodes

A : set of arcs

K : set of commodities

n : index for nodes

k : index for commodities

i : index representing the first node in the arc

j : index representing the next node in the arc

b_i^k : integer number representing the supply or demand of node i for commodity k

l_{ij} : lower bound (capacity) that denotes the minimum amount that can flow on the arc

u_{ij} : upper bound (capacity) that denotes the maximum amount that can flow on the arc

l_{ij}^k : lower bound that denotes the minimum amount that can flow on the arc for commodity *k.*

u_{ij}^k : upper bound that denotes the maximum amount that can flow on the arc for commodity *k.*

The linear programming formulation for the multi-commodity minimal cost flow problem is as follows:

$$Minimize \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k X_{ij}^k \qquad (3.1)$$

Subject to:

$$\sum_{j:(i,j)\in A} X_{ij}^{k} - \sum_{j:(j,i)\in A} X_{ji}^{k} = b_i^k \qquad \forall\, i \in N\,, \forall k \in K \qquad (3.2)$$

$$\sum_{k\in K} X_{ij}^{k} \le u_{ij} \qquad \forall (i,j) \in A \qquad (3.3)$$

$$\sum_{k\in K} X_{ij}^{k} \ge l_{ij} \qquad \forall (i,j) \in A \qquad (3.4)$$

$$l_{ij}^{k} \le X_{ij}^{k} \le u_{ij}^{k} \qquad \forall (i,j) \in A, \forall k \in K \qquad (3.5)$$

The objective function (3.1) minimizes the total flow costs. Constraint (3.2) is the supply/demand constraint. The constraints (3.3) and (3.4) are the bundle constraints. The constraint (3.5) is the capacity constraint for each commodity.

3.1.2 Application Areas of Multi-Commodity Network Flow Models

MCNF models arise in many real-world applications. Most of them are network routing and network design problems.

- Network routing
  - The transmission of messages in a communication network between different origin–destination (O-D) pairs. (Each requested O-D pair is considered to be a commodity. The problem is to find a minimum cost flow routing for all demands of requested O-D pairs while satisfying the arc capacities.)
- Scheduling and routing in logistics and transportation:
  - The transportation of passengers from different origins to different destinations (each passenger is considered to be a commodity)

- The routing of aircrafts (assigning aircrafts to the set of sequential flight legs where destination station of the first flight leg is the same as the origin station of the second flight leg. Each aircraft is considered to be a commodity)

- The crew scheduling (assigning crews to set of flight legs where destination station of the first flight leg is the same as the origin station of the second flight leg. Each crew is considered to be a commodity

- The routing of nonhomogeneous tankers (nonhomogeneous in terms of speed, carrying capability, and operating costs)

• Production scheduling and planning:

- The worldwide shipment of different varieties of grains (such as corn, wheat, rice and soybeans) from countries that produce grains to those that consume it

- Warehousing and distribution problem for seasonal products (commodities are products to be shipped)

• Other routing problems:

- VLSI (Very-Large-Scale Integration) design problem (the process of creating integrated circuits by combining thousands of transistors into a single chip. - nodes represent collections of terminals to be wired, arcs correspond to the channels through which the wires run, and commodities are OD pairs to be routed-).

- Racial balancing of schools

- Caching and prefetching problems in disk systems

- Traffic equilibrium problem
- Graph Partitioning problem

3.1.3 The Multi-Commodity Network Models for Airline Optimization Problems

As explained in the literature review of the airline planning and operations problems in Chapter 2, multi-commodity network flow problems are mostly used technique in order to formulate the scheduling and assignment problems in the airline planning and operations. Especially, most of the models developed for fleet assignment problems uses the multi-commodity network flow problems (see Table 2.1). Also, multi-commodity network flow problems are mostly used technique used in the formulation of aircraft and crew recovery problems (see Tables 2.3 and 2.4).

In fleet assignment problems, each fleet type corresponds to each commodity. In aircraft recovery problems, each aircraft corresponds to each commodity.

While formulating the fleet assignment or aircraft and crew recovery problems as multi-commodity network flow models, underlying network structure should be developed. The most of the studies in the airline industry literature use the time-space network as underlying network for multi-commodity network flow problems. There are also some important studies, which use the connection network as underlying network for multi-commodity network flow problems.

Table 3.1 The example flight schedule and aircraft assignments for network development

Aircraft	Flight No	Origin		Destination		Departure	Arrival	Flight time
AC1	FL030	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	8:15	9:15	1:00
	FL032	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	10:45	11:45	1:00
	FL024	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	16:30	17:30	1:00
	FL026	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	18:50	19:45	0:55
AC2	FL010	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	7:30	8:25	0:55
	FL020	IST	İstanbul-Atatürk	GZT	Gaziantep-GAZIANTEP	9:30	11:15	1:45
	FL022	GZT	Gaziantep-GAZIANTEP	IST	İstanbul-Atatürk	12:15	14:00	1:45
	FL034	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	16:45	17:45	1:00
	FL036	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	18:45	19:45	1:00
	FL012	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	20:45	21:45	1:00
AC3	FL040	IST	İstanbul-Atatürk	ADA	Adana-ADANA	15:00	16:30	1:30
	FL042	ADA	Adana-ADANA	IST	İstanbul-Atatürk	17:30	19:00	1:30
	FL044	IST	İstanbul-Atatürk	ADA	Adana-ADANA	20:00	21:30	1:30
	FL046	ADA	Adana-ADANA	IST	İstanbul-Atatürk	22:30	0:00	1:30

In the following sections the usage of both time-space and connection networks in the airline optimization problems will be explained and time-space and connection network representations for the example schedule seen in Table 3.1 will be given.

3.1.3.1 Time-Space Network in Airline Optimization Problems

The time-space network consists of two dimensions. One is space, which is airport in airline scheduling applications, and the other is time.

Each node within the network represents an event taking place in a specific airport at a specific time, such as the departure of aircraft/crew from the airport or arrival of the aircraft/crew to the airport. Each airport corresponds to a line to be considered as the time line of the airport and all event nodes for that airport is positioned on the time line at the corresponding points in time. The length of the time line represents the planning period.

Each arc within the network shows the linkage among different event nodes and connects event nodes corresponding to events, which follow each other in a sequence of events for particular airline resource (aircraft or crew). The vertical arcs connecting nodes on time line for a certain airport corresponds to grounding arcs. The diagonal arcs connecting nodes on the time lines of the different airports (between the departure node of the flight i and the arrival node of the flight j) correspond to flight legs with respect to flight times and airports.

The nodes used in almost each airline optimization problems are as follows:

- *Aircraft departure node*: Aircraft departs at a specific time from a specific airport. In the network, flight link exists from the aircraft departure node and connects to aircraft arrival node at the other station.

- *Aircraft arrival node*: Aircraft arrives at a specific time in a specific airport, and this node records the arrival time and airport. In the network, a flight link will emerge into the aircraft arrival node.

- *Source (Supply) node*: Source node represents number of aircraft/crew available by request in a specific airport. The source node can be either in the beginning of the day, when the initial supply comes into use, or later at some specific moments of the day, at which time the recovery of the aircraft is assured or the spare aircraft becomes available.

- *Sink (Demand) node*: The demand node can be taken as the incident node, if it appears in the middle of the day. Incident demand node indicates a shortage of aircraft at the time moment in a specific airport. If it appears at the end of the day, this is just the demand node. This means that after all the operations we have to have as many aircraft located at the airport at the end of the operating period as that in the morning.

The arcs used in the time-space network representation in the airline optimization problems are as follows:

- *Flight arc:* Flight arc connects aircraft departure node to aircraft arrival node, which means a possible flight. Generally, the amount of flow on this arc is one, but if there are two flights, which depart at the same time and go to the same location, the flow can be greater than one. The cost of flight arc is the operating cost.

- *Ground holding arc:* This arc connects both arcs within the same airport but different time horizon. The amount of flow in the arc represents the number of aircraft kept during the ground holding time. There might be some costs of parking charges, facility charges, etc. and usually no revenue is generated in a ground holding arc.

The Figure 3.1 represents the three aircrafts' routes, seen in the example schedule in Table 3.1, in time-space network structure.

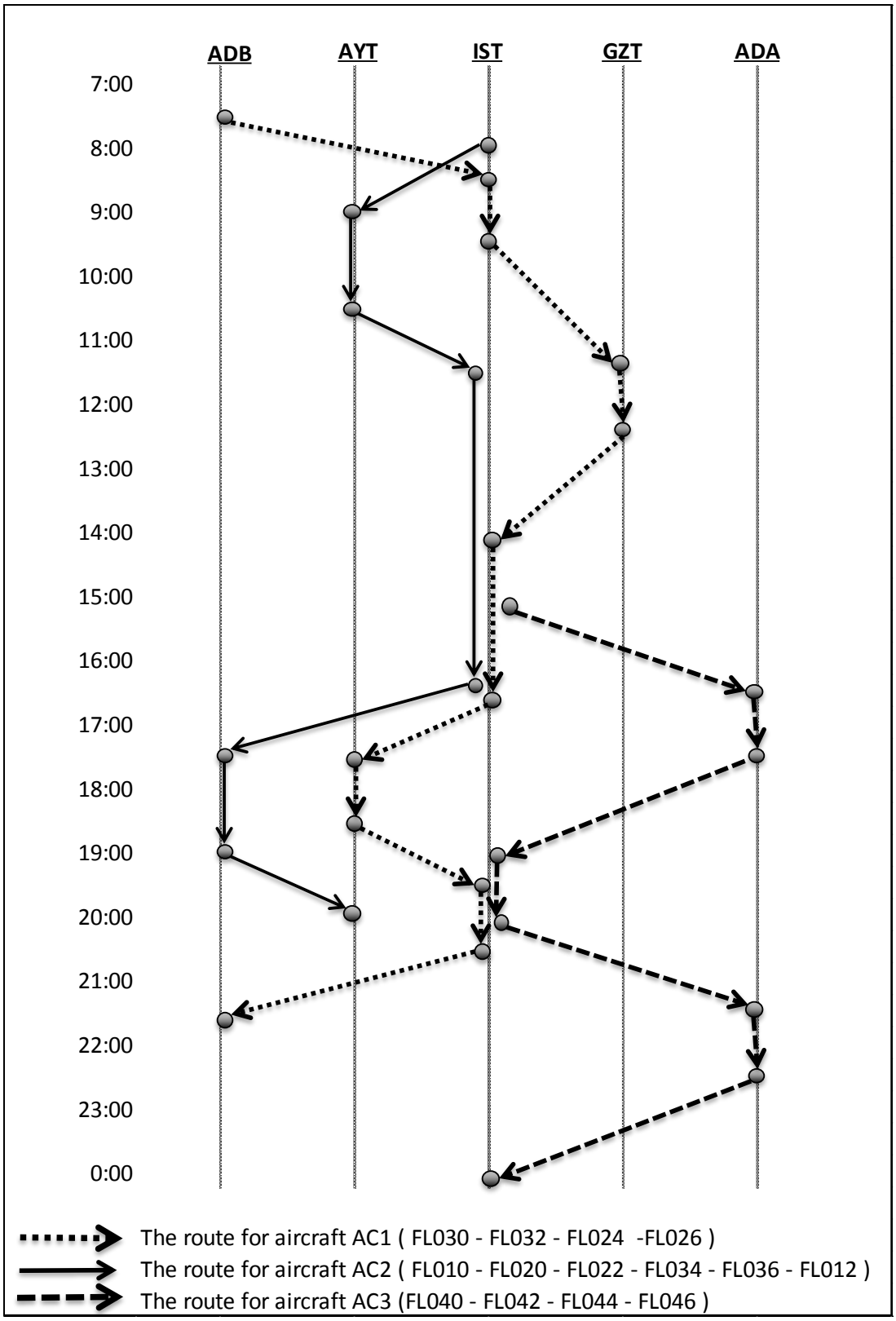

Figure 3.1 The time-space network representation of example schedule

The aircraft AC2 first flies FL010 leaving the airport ADB at 07:30 and arriving at the airport IST at 08:25. Therefore these activity is represented by a diagonal arc in the time-space network, where the arc begins at the departure airport (ADB) of flight FL010, on the time line of ADB at the point of departure time (07:30) and ends at the arrival airport (IST) of flight FL010, on the time line of IST at the point of arrival time (08:25).

Then the same aircraft AC2 stays at the airport IST until its next flight departures at 09:30. The corresponding activity of staying on the ground of aircraft AC2 is represented by a vertical arc, which begins at the arrival airport (IST) of flight FL010, on the time line of IST at the point of arrival time (08:25), and ends at the same airport (IST) which is the origin airport of the second flight of AC2 (FL020), on the time line of IST at the point of departure time of the second flight (FL020) at 09:30. The other flight arcs and ground arcs are positioned as well and can be seen in the Figure 3.1.

In the time-space network representation of the aircraft and crew recovery problems, some additional arcs are used in order to represent the activities regarding the recovery strategies.

- *Ferry arc:* Ferry arc indicates the spare aircraft flying without carrying passengers to an airport to support the operation under perturbation. Ferry arc will start from a source node (supply node), which generates one more unit (or more) of flow whenever the schedule controllers consider ferrying a spare aircraft to support the operation when there is any available. The source node of the spare aircraft is connected to all the other airports and the total flow of this bundle arcs is set less or equal to one. The cost for ferry arc is just the operating cost and some administrative costs. No revenue will be obtained.

- *Delay arc:* Delay arc is the production of the delay strategy. Some parallel arcs are added to the scheduled flight arc with some discrete interval, such as 15 minutes. For example, assuming that a flight FL020 departs from airport IST at 09:30 and arrives at airport GZT at 11:15. Delay arcs parallel to the flight arc

of FL020 can be added to the network, departing at 09:45, 10:00 PM, 10:15 ...etc., and arriving at 11:30, 11:45 and 12:00, … accordingly. The cost for the delay arc will be the delayed cost, as well as the operating cost due to the delay.

- *Speed up arc:* Speed up arc reduces the flight time even though the origin is the same with the aircraft departure node (or source node), and the arrival time with speed up arc will be earlier compared to the scheduled arrival time. Amount of time managed from speeding aircraft up is determined by the distance between the departure and arrival airports. If the aircraft is on the air, it depends on how far the aircraft is from the destination. The cost for speed up arc is just the fuel cost. No other penalty cost is generated.

In the time-space networks, arcs represent the activities and schedule information is represented explicitly in the network. Therefore, time-space network provides a clear way of viewing the structure of the problem and it can be easily comprehended due to its logical setup. However, since the number of nodes and arcs are greater than the connection network (at least 2 nodes for each flight leg, one for departure, one for arrival), the size of the resulting network flow problem is larger.

3.1.3.2 Connection Network in Airline Optimization Problems

The flights correspond to the nodes in the network. Connection network consists of a set of nodes, N, one for each flight leg. A flight leg is given by its origin, destination, departure time and arrival time. The node i representing the flight leg l_i is connected to node j representing flight leg l_j by a directed arc (i, j), if it is feasible to fly flight leg l_j after flight leg l_i with respect to turn-around times and airports. There are also source and sink nodes which represents the position of each airline resources (aircraft or crew) in the beginning and at the end of the planning period.

In the connection network, the nodes represent the flight legs. In addition, an imaginary master source node and a master sink node are conceptualized (not actually created) in the network to account for the beginning-of-the-day and the end-

of-the- day effects. The arcs connect the flight legs represented by nodes, satisfying a feasible sequence of flights for an aircraft or crew.

The nodes used in the connection network are as follows:

- *Source node:* Source node represents the initial position of the aircraft or crew at the beginning of the planning horizon. In airline planning problems, there is one source node for each hub station where the aircrafts and crews are positioned before their route or pairing begins.

- *Sink node:* Sink node represents the last position of the aircraft or crew at the end of the planning horizon. In the airline planning problems, there is one sink node for each hub station where the aircrafts and crews return after they completed their routes or pairings.

- *Flight node:* There are flight nodes between source and sink nodes which represent each flight leg. The flight nodes involve all the flighting process between the departure of the aircraft/crew from an airport and the arrival of the aircraft/crew to another airport. The departure time, the arrival time, origin and destination stations of the flight leg are not represented explicitly through the network, however they are formulated as side constraint in the mathematical representation of the problem.

There are three types of arcs representing the different types of connections:

- *The flight connection arcs*: The flight connection arcs link the arrival nodes to the departure nodes. All flight connections have to be feasible with respect to flight arrival and departure times; that is, the minimum turn-time has to be observed between the arrival flight and the following departure flight to allow for the connection.

- *The terminating (connection) arcs*: These arcs link the arrival nodes to the master sink node to represent aircraft arriving and remaining at the station for the rest of the day.

- *The originating (connection) arcs*: These arcs link the master source node to the departure nodes to represent the aircraft that are present at the station at the beginning of the day.

The connection network representation of the example schedule seen in Table 3.1 can be seen in Figure 3.2. The routes for three aircrafts are highlighted with different arcs. Therefore, the route of aircraft AC1 begins from the source node (IST) and follows node FL030, FL032, FL024, FL026. Finally it ends with the sink node (IST). The arcs in this route connect the sequential flight legs, which AC1 can fly with respect to turn-around times. Similarly the connection network seen in Figure 3.2 also represents the routes of aircrafts AC2 and AC3.

In the aircraft and crew recovery problems considering the recovery strategy of delaying flight legs, the connection network should also include the flight nodes of the delayed flights. For example if flight leg FL030 is delayed for 15 minutes, a delayed flight should be added as a new node to the network. This new node will have also new connection arcs.

The one problem with the connection network is that it is difficult to view as representation in time and space of the possible schedules. Therefore, in the recovery problems, the recovery strategies such as delays, deadheads and speed-ups can not be explicitly displayed in the network representation. Such recovery related decisions are included in the mathematical definition of the problem. However, in the problems developed with connection networks, the number of nodes and arcs are not as much as in the time-space network, therefore it is simpler to develop connection networks.

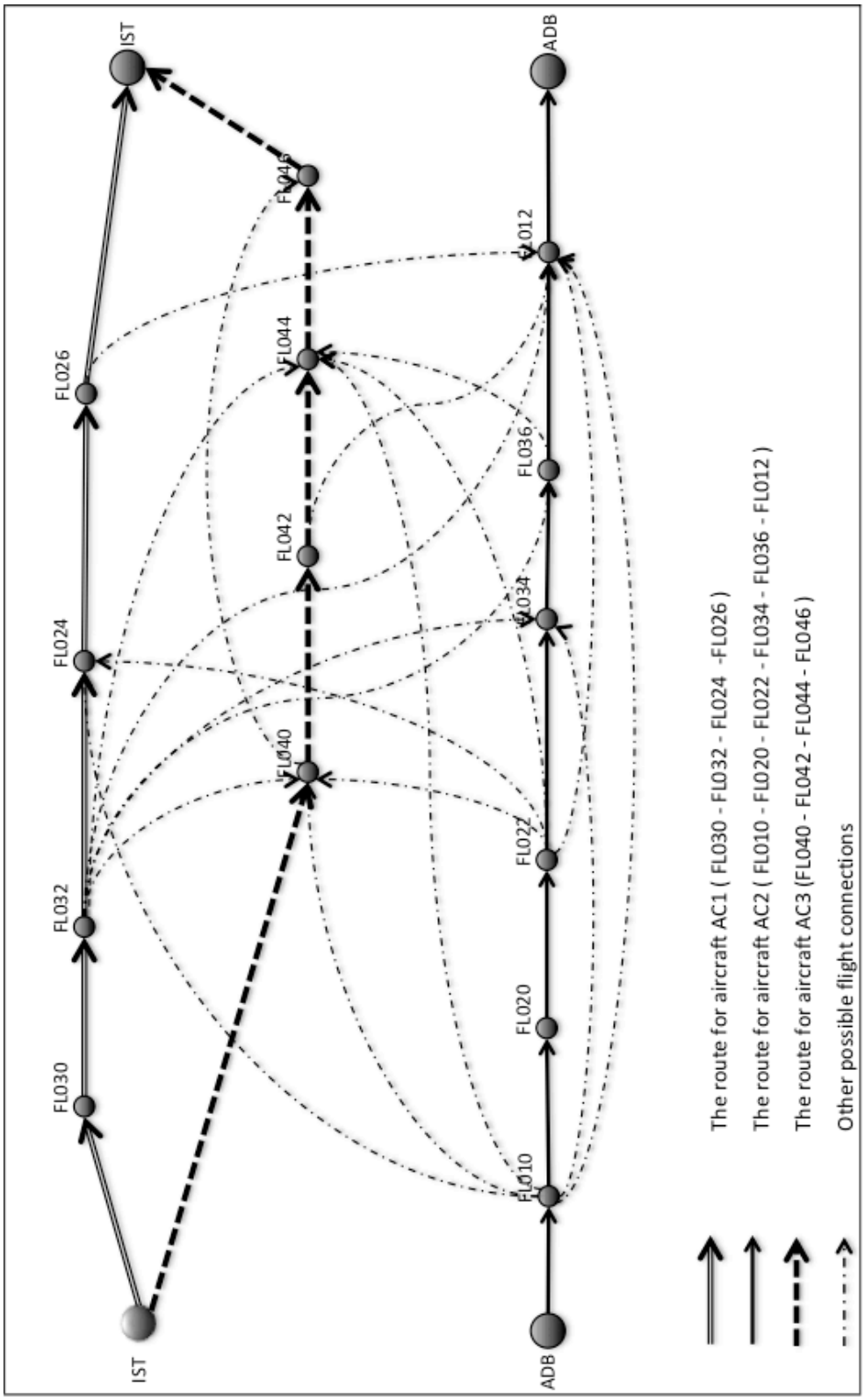

Figure 3.2 The connection network representation of the example schedule

3.2 Brief Overview of Constraint Programming

Constraint programming has emerged in the last decades as a new tool to address various classes of discrete and continuous combinatorial search problems.

Constraint programming is a powerful tool that can be used as an alternative method to mathematical programming techniques. Constraint programming can be successfully applied to various problems such as scheduling, configuration, and resource allocation. Constraint Programming offers a more flexible modeling framework than mathematical programming.

The strength of the constraint programming approach is that it allows separation between the statement of the problem such as the variables and the constraints and the algorithms of the problem.

Constraint programming is the mostly used approach for problems with different types of variables and constraints, such as those with real, integer and logical variables, and linear and logical constraints. Constraint programming enables the explicit use of logical constraints to a more compact model and allows program a solution strategy specific to the problem conveniently. Constraint programming systems also provide a uniform framework to integrate a variety of special purpose algorithms, while presenting a coherent view to the user for hybrid programming.

3.2.1 Definition of Constraint Programming

Constraint programming is the result of the study in artificial intelligence area and consists of two phases: logical programming phase and constraint satisfaction phase. Constraint programming is often called constraint logic programming, and it originates in the artificial intelligence literature in the computer science community. Here, the word programming refers to computer programming that can be viewed as a plan of action of operations of the computer.

Constraint programming is a computer programming technique, with a name that is in the spirit of other programming techniques, such as object-oriented programming, functional programming, and structured programming. Logic programming is a declarative, relational style of programming based on first-order logic, where simple resolution algorithms are used to resolve the logical statements of a problem. Constraint logic programming extends this concept by using more powerful algorithms to resolve these statements. Van Hentenryck (1999) writes:

> "The essence of constraint programming is a two-level architecture integrating a constraint and a programming component. The constraint component provides the basic operations of the architecture and consists of a system reasoning about fundamental properties of constraint systems such as satisfiability and entailment. The constraint component is often called the constraint store; by analogy to the memory store of traditional programming languages….Operating around the constraint store is a programming-language component that specifies how to combine the basic operations, often in non-deterministic ways. (Hentenryck, 1999, p:4)

Hence, a constraint program is not a statement of a problem as in mathematical programming, but is rather a computer program that indicates a method for solving a particular problem. It is important to emphasize the two-level architecture of a constraint programming system. Because it is first and foremost a computer programming system, the system contains representations of programming variables, which are representations of memory cells in a computer that can be manipulated within the system.

The first level of the constraint programming architecture allows users to state constraints over these programming variables. The second level of this architecture allows users to write a computer program that indicates how the variables should be modified so as to find values of the variables that satisfy the constraints.

The principal elements of constraint programming are:

- Variables
- Constraints, which link variables
- Solution search approaches

Variables include such types as integer, Boolean, symbolic, rational numbers, real numbers. Thus the set is richer than Integer Linear Programming (ILP) which would only include integer, Boolean and real number types.

Constraints can be of several different types such as:

- Mathematical constraints, e.g., $a + b = c$
- Disjunctive constraints, e.g., tasks A and B occur at different times
- Relational constraints, e.g., at most two jobs should be allocated to machine3.
- Explicit constraints, e.g., (x, y) must be $(1,2), (2,3)\ or\ (4,5)$
- Unary constraints, e.g., z is an integer between 5 and 10.

Again, the set of types of constraints is richer than that for ILP. In ILP all constraints must be linear constraints. Within CP constraint operators and relational operators such as $=, \leq, \geq, <, >, *, /$, s*ubset, superset, union, intersection, member*, boolean *OR*, boolean *AND*, and boolean *XOR* are all permissible.

3.2.2 Constraint Satisfaction Problem

Constraint programming (CP) techniques have been widely used to solve a large range of combinatorial problems. Constraint Satisfaction Problems (CSP) are an important class of combinatorial optimization problems.

Constraint satisfaction problem is defined as follows:

Given a set of n decision variables $x_1, x_2, x_3, \dots, x_n$ the set D_j of allowable values for each decision variable $x_j, j = 1,2, \dots, n$ is called the domain of the variable x_j.

The domain of a decision variable can be any possible set, operating over any possible set of symbols. For example, the domain of a variable could be the even integers between 0 and 100 or the set of real numbers in the interval [1,100] or a set of people {*Tom, Judy, Jim, Ann*}. There is no restriction on the type of each decision variable, and hence decision variable can take on integer values, real values, set elements, or even subsets of sets.

Formally, a constraint $c(x_1, x_2, x_3, \dots, x_n)$ is a mathematical relation, that is, a subset S of the set $D_1 \text{ X } D_2 \text{ X} \dots \text{X } D_n$, such that if $(x_1, x_2, x_3, \dots, x_n) \, S$, then the constraint is said to be satisfied. Alternatively, a constraint can be defined as a mathematical function

$f: D_1 \text{ X } D_2 \text{ X} \dots \text{X } D_n \rightarrow \{0,1\}$ such that $f(x_1, x_2, x_3, \dots, x_n) = 1$

if and only if $c(x_1, x_2, x_3, \dots, x_n)$ is satisfied. Using this functional notation, a constraint satisfaction problem can be defined as: Given n domains $D_1, D_2, \dots, D_n$, and m constraints $f_1, f_2, \dots, f_m$ find $x_1, x_2, \dots, x_n$ such that

$$f_k(x_1, x_2, x_3, \dots, x_n) = 1, \qquad 1 \leq k \leq m$$

$$x_j \in D_j \qquad 1 \leq j \leq n$$

This problem is only a feasibility problem, and no objective function is defined.

Here the functions f_k do not necessarily have closed mathematical forms (for example, functional representations) and can be defined simply by providing the set S described above. A solution to a CSP is simply a set of values of the variables such that the values are in the domains of the variables and all of the constraints are satisfied.

The solution methods of Constraint Satisfaction Problem are as follows:

- Various algorithms, Linear Programming, Dynamic Programming, etc., which have been researched by Operations Research (OR).
- Various techniques based on Tree Reference
- Techniques based on constraint rewriting such as the merging method or condensing by the programming conversion.
- Algorithms for Solution Improvement such as minimum conflict/ taboo search, Simulated Annealing.
- Search by using neural network
- Genetic Algorithm

An example for CSP is Map-Coloring Problem. It is to color the different regions in a particular map with a limited number of colors. Subject to the constraints is that no two adjacent regions have the same color. The problem is solved for 4 regions with 3 colors which are red, green, blue respectively.

- Variables: represent a region on the map: The value of variables represents the color assigned to that region

  $V = \{x_1, x_2, x_3, x_4\}$

- Domains: $D_1 = D_2 = D_3 = D_4 = \{red, green, blue\}$

  The domain $D_i \; of \; x_i$: set of available colors for region i

- Constraints: no two adjacent regions will be colored the same

  $x_1 \neq x_2,\ x_1 \neq x_3,\ x_1 \neq x_4,\ x_2 \neq x_3,\ x_2 \neq x_4,\ x_3 \neq x_4$

3.2.3 Constraint Propagation and Domain Reduction

A constraint is defined as a mathematical function $f(x_1, x_2, x_3, \dots, x_n)$ of the variables. Within this environment, it is assumed that there is an underlying mechanism that allows the domains of the variables to be maintained and updated.

When a variable's domain is modified, the effects of this modification are then communicated to any constraint that interacts with that variable. This communication is called constraint propagation.

For each constraint, a domain reduction algorithm modifies the domains of all the variables in that constraint, given the modification of one of the variables in that constraint. The domain reduction algorithm for a particular kind of constraint discovers inconsistencies among the domains of the variables in that constraint by removing values from the domains of the variables. If the algorithm discovers that a particular variable's domain becomes empty, then it can be determined that the constraint can not be satisfied, and an earlier choice can be undone.

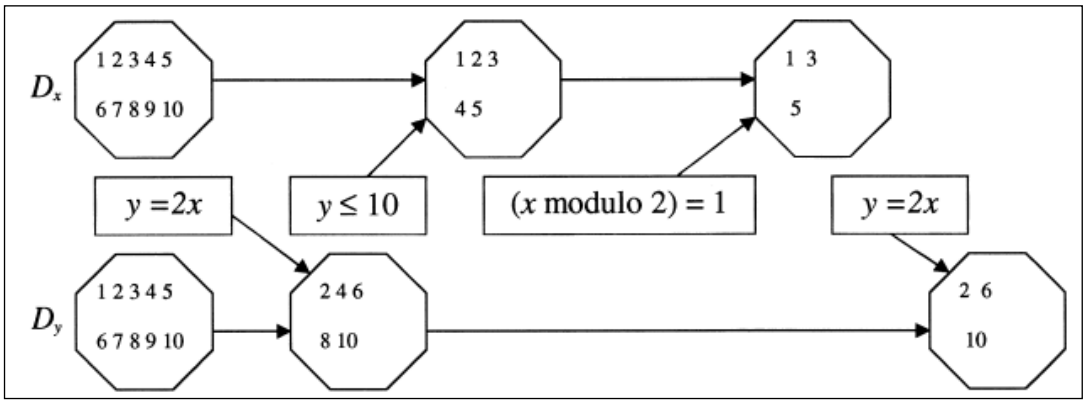

Figure 3.3 Constraint propagation and domain reduction are used to reduce the domains of the variables x and y. The constraints $y = 2x$, $y = 10$, (x modulo 2)=1, and $y = 2x$ are applied in an order determined by constraint propagation, due to reductions in the domains of each of the variables.

A similar methodology is found in the bound-strengthening algorithms used in modern mathematical programming solvers. A crucial difference between the procedures used in mathematical programming resolved implementations and domain reduction algorithms is that in constraint programming, the domains can have holes, while in mathematical programming, domains are intervals.

Some constraint programming systems (for example, ILOG Solver, Oz, and ECLiPSe) allow the programmer to take advantage of existing propagators for built-

in constraints that cause domain reductions and allow the programmer to build his or her own propagation and domain reduction schemes for user-defined constraints.

However, many constraint programming systems (for example, OPL, ILOG Solver, and CHIP) are strong enough that they provide large libraries of predefined constraints, with associated propagation and domain reduction algorithms, and it is often not necessary to create new constraints with specialized propagation and domain reduction algorithms. Given a set of variables with their domains and a set of constraints on those variables, a constraint programming system will apply the constraint propagation and domain reduction algorithm in an iterative fashion to make the domains of each variable as small as possible, while making the entire system arc consistent.

3.2.4 Constraint Programming Algorithms for Optimization

As compared to linear and mixed integer programming, a weakness of a constraint programming approach when applied to a problem with an objective function to minimize is that a lower bound may not exist. A lower bound may be available if the expression representing the objective function has a lower bound that can be derived from constraint propagation and domain reduction. This is unlike integer programming, in which a lower bound always exists because of the linear programming relaxation of the problem. Constraint programming systems offer two methods for optimizing problems, called standard and dichotomic search.

The standard search procedure is to first find a feasible solution to the CSP, while ignoring the objective function $g(x_1, x_2, \dots, x_n)$. Let $y_1, y_2, \dots, y_n$ represent such a feasible point. The search space can then be pruned by adding the constraint $g(y_1, y_2, \dots, y_n) > g(x_1, x_2, \dots, x_n)$ to the system and continuing the search. The added constraint specifies that any new feasible point must have a better objective value than the current point. Propagation of this constraint may cause the domains of the decision variables to be reduced, thus reducing the size of the search space. As the search progresses, new points will have progressively better objective values.

The procedure concludes when no feasible point is found. When this happens, the last feasible point can be taken as the optimal solution.

Dichotomic search depends on having a good lower bound L on the objective function $g(x_1, x_2, \dots, x_n)$. Before optimizing the objective function, a procedure must find an initial feasible point, which determines an upper bound U on the objective function. A dichotomic search procedure is essentially a binary search on the objective function. The procedure computes the midpoint $M = (U + L)/2$ of the two bounds and then solves a CSP by taking the original constraints and adding the constraint $g(x_1, x_2, \dots, x_n) < M$. If it finds a new feasible point, then it updates the upper bound and continues the search in the same way with a new midpoint M. If it finds the system to be infeasible, then it updates the lower bound, and the search again continues with a new midpoint M. Dichotomic search is effective when the lower bound is strong, because the computation time to prove that a CSP is infeasible can often be large. The use of dichotomic search in cooperation with a linear programming solver may be effective if the linear programming representation can provide a good lower bound. The difference between this procedure and a branch-and-bound procedure for mixed-integer programming is that the dichotomic search stresses the search for feasible solutions, whereas branch-and-bound procedures usually emphasize improving the lower bound.

3.2.5 Applications Areas of Constraint Programming

There are some types of problems, which can be solved with comparatively little effort using CP based tools. Those applications share some general characteristics:

- No general, efficient algorithms exist (NP-completeness): specific techniques/heuristics must be used. These are usually problems with a heavy combinatorial part, and enumerating solutions is often impractical altogether. A fast program using usual programming paradigms is often too hard and complicated to produce, and normally it is so tied to the particular problem that adapting it to a related problem is not easy.

- The problem specification has a dynamic component: it should be easy to change programs rapidly to adapt. This has points in common with the previous item: CP tools have built-in algorithms which have been tuned to show good behavior in a variety of scenarios, so updating the program to new conditions amounts to changing the setting up of the equations.

- Decision support required: either automatically in the program or in cooperation with the user. Many decisions can be encoded in mathematical formula, which appear as rules and which are handled by the internal solvers, so there is no need to program explicit decision trees.

According to these characteristics, application areas of CP can be listed as follows:

- Operation research problems (optimization problems such as: scheduling, sequencing, resource allocation, timetabling, job-shop, traveling salesman, crew-aircraft rotation, personnel assignment, cutting problems)
- Electrical engineering (location of faults in circuits, circuit layout, verification of circuit design,)
- Business applications (option trading, financial problems)
- Graphic systems (to maintain consistency of moving objects, computer-aided design)
- Molecular biology (search for patterns, 3D models of proteins)
- Natural language (efficient parsers, speech recognition with semantics)
- Internet (constrained web queries)
- Hardware design
- Air Traffic Control
- Frequency Allocation
- Network Configuration
- Product Design
- Satellite Tasking

3.2.6 Advantages of Constraint Programming

1. *Declarative Nature*. The main contribution of constraint programming is the significant reduction of the gap between the informal description of a problem and its computer specification. The focus in constraint programming is more on describing the problem to be solved, not on specifying how to solve it. In particular, there is a clear separation between the constraints and the heuristics used to explore the search space. Programming is still present and can be critical, but it takes place at a higher level.

2. *Cooperative Problem Solving*. A strength of constraint programming is to provide a uniform framework to integrate a variety of special-purpose algorithms, while presenting a coherent view to users. Successful languages such as CHIP, Ilog Solver, and Prolog III/IV combine problem-solving techniques from various fields (e.g., artificial intelligence, numerical analysis, operations research) to obtain solutions to complex problems. As a consequence, constraint programming is a platform for integrating and combining a variety of algorithms and for making them available to a wide audience.

3. *Applications*. Convincing applications have been developed concurrently with the design of the languages, showing the promise of the technology in a timely fashion and guiding extensions to the language.

4. *Semantic Foundations*. Although applications have driven many extensions, the languages have remained amazingly clean and elegant.

3.2.7 Limitations of Constraint Programming

1. *Constraint programming is too high-level*. There is a need for constraint programming to be more open and extensible. There is little support in many existing languages to define new constraints, new constraint solvers, and new heuristics. Concurrent constraint programming in general, addresses this issue by offering general-purpose combinators such as constructive disjunction. Oz offers a meta-level

architecture where procedures and computation spaces are first-class objects. Ilog Solver offers low-level mechanisms to define new constraints. Each of these approaches achieves a different tradeoff between expressiveness, ease of use, and efficiency.

2. *Constraint programming is too low-level*. Constraint programming is still far from the description of the problem and the gap between the description and the program is non-trivial.

3. *Constraint programs are unpredictable*. Although many researchers in constraint programming have developed intuitions on when and how to use constraints, there is a strong need for more characterizations of the performances of various approaches and modelings.

4. *Constraint programs are not incremental*. Constraint programs are in fact incremental in some sense (they can add constraints dynamically) but they have no support for online constraint solving. Most of the time they are used to produce, in an offline manner, a plan or a resource allocation which is then executed. However, machines break down, planes are delayed, new orders come at the worst possible time. Upgrading an offline solution in a reasonable fashion and in a reasonable time to absorb unexpected events is a fundamental challenge for the constraint community.

5. *Constraint propagation is too local*. It carries global information by maintaining domains or intervals. Constraint programming should have the ability to manipulate subsets of constraints globally and recent work in CHIP, Ilog Solver, and Newton has shown how beneficial global constraints can be. In addition, it is surprising to see that these global constraints can often be given a semantics as instances of general-purpose consistency conditions, making their integration in the languages natural. The main challenge here is probably to find or identify innovative ways to combine constraints to produce efficient pruning techniques.

3.2.8 Constraint Programming vs. Mathematical Programming

As a modeling tool constraint programming is more flexible than the mathematical programming. The objective functions as well as the constraints can be more general. Because the decision variables may represent not only the integer values and real values, but also set elements and even subset of sets.

Optimal solution searches can be conducted in mathematical programming and in constraint programming have similarities as well as differences.

One similarity between constraint programming and the branch-and-bound method applied to integer programming is that both approaches search through the tree of which each node represents a partial solution of a schedule.

One of the differences between the dichotomic search used in constraint programming and a branch-and-bound method applied to mixed-integer programming is that the dichotomic search stresses a search for feasible solutions, whereas a branch-and-bound method emphasizes improvements on the lower bound.

Constraint programming techniques and mathematical programming techniques can be embedded within one framework, as we do in this dissertation. Such an integrated approach has already proven useful for a variety of scheduling and routing problems. Such solution approach may integrate the strength of the both constraint programming and mathematical programming techniques, obtaining more efficient solutions in less time.

In this dissertation a novel solution algorithm is proposed including both constraint programming and integer programming techniques for solving aircraft recovery, crew recovery problems and also their integration. Therefore, we take advantage of constraint programming and use it with integer programming, inspite of constraint programming and integer programming techniques' limitations on its own.

CHAPTER FOUR
AIRCRAFT RECOVERY PROBLEMS

Since the aircraft is the most expensive resource of the airlines, its effective utilization is very important. The large savings are possible using aircrafts more efficiently. Therefore, in case of disruptions, the airlines give much effort on recovering aircrafts rather than recovering other resources such as crews and passengers.

As explained in Chapter 2, there are a lot of published works on aircraft recovery problems in the disruptions management literature. Among all other recovery problems, the aircraft recovery problems are the most studied one in the literature. The reasons why the aircraft recovery problems have been studied so much are as follows:

1. The aircrafts are the most expensive resources in the airlines and the shortage of an aircraft causes great losses in the airlines revenues.

2. The aircrafts are the "main" and most import resource of the airlines and they are fixed first and foremost.

3. The aircrafts are the easiest resource due to lowest complexity in rules and the smallest numbers. However, the rules and regulations for another important resource crews is highly complex and the number of crews in the airlines is much larger than the number of aircrafts.

4. It is more difficult to replace / substitute aircrafts in case of their shortage since they are scare resources. However, crews can be repositioned much more easily or standby crews are often available.

The aircraft recovery problems in the literature have been usually modeled as multi-commodity network flow problems where the underlying network is time-

space network. However, in a few studies the connection network has also been used as the underlying network for multi-commodity network flow problem.

In this dissertation, for modeling aircraft recovery problems both the traditional method, which is multi-commodity network flow problem and the new method, which is constraint programming will be used. While developing the models with multi-commodity network flow problem, each commodity represents each aircraft and the connection network is used as underlying network. The advantage of the connection network is that the complexity and the size of problem is reduced by reducing the number of nodes and the number of arcs, therefore reducing the number of variables. Another advantage is that with connection network it is easier and less time consuming to construct a network.

In the connection network, the nodes in the network represent scheduled flight legs with corresponding departure/arrival times and origin/destination stations. The arcs in the network represent connections between two consecutive flight legs where the destination of the first flight leg is the same as the origin of the next flight leg.

This chapter will introduce several models developed for aircraft recovery problem in this dissertation. These models are:

- Model A1: Cancellation model for single aircraft unavailability using multi-commodity network flow problem where the underlying network is connection network

- Model A2: Cancellation and delay model for multiple aircraft unavailabilities using multi-commodity network flow problem where the underlying network is connection network

- Model A3: Cancellation and delay model for multiple aircraft unavailabilities including crew considerations using multi-commodity network flow problem where the underlying network is connection network

- Model A4: Cancellation and delay model for multiple aircraft unavailabilities including crew considerations using constraint programming embedded solution approach

The developed models will be tested using the real data obtained from obtained from domestic flight schedule of OnurAir, one of the major airlines in Turkey.

- The OnurAir operates 38 flight legs in a day.
- All the flights are flown within the same day over a network of 12 stations.
- There are 2 different fleet types and total of 9 aircrafts: Boeing MD-88 and Boeing MD-83. There are 5 aircrafts of Boeing MD-88 type and 4 aircrafts of Boeing MD-83 type. Both types of aircrafts have the same technical specifications and are capable of flying all the domestic flights of the airline. Fleet information of OnurAir can be seen in Appendix A in Table A.1.

In order to solve aircraft recovery problems, the flight schedule of the airline, initial aircraft-flight assignments and initial crew schedule if crew consideration is included in aircraft recovery problem should be obtained in advance. Therefore, aircraft routing and crew scheduling problems have been solved for OnurAir.

For the Models A1, A2 and A3 which are modeled as multi-commodity network flow problems, the initial crew schedule and initial aircraft-flight assignments seen in Appendix A in Table A.2 will be used.

For Model A4, which is modeled using constraint programming, the initial crew schedule and initial aircraft-flight assignments have been modified by solving these two problems with constraint programming. For Model A4 the initial crew schedule and initial aircraft-flight assignments seen in Appendix A in Table A.3 will be used.

4.1 Model A1

Cancellation Model with Single Aircraft Unavailability Using Multi-Commodity Network Flow Model

Model A1 is developed as multi-commodity network flow model where the underlying network is connection network. The Model A1 considers two recovery strategies, which are cancelling flights and swapping aircrafts among the flights. The Model 1 is capable of solving only one aircraft unavailability. In Model A1:

- Three different flight connection networks are used in the problem formulation, which represent
  - The flight connections before recovery period
  - The flight connections during recovery period
  - The flight connections after recovery period

- It is assumed that the maintenance service of the aircrafts is made after the midnight, when no flights are scheduled to be flown.

- Only two recovery strategies are considered:
  - Cancelling flights
  - Swapping aircrafts among scheduled flights

- It is assumed that the crews are available at any time, thus the crew availability will not affect the flight's operability.

- Only one aircraft unavailability (shortage) is considered.

4.1.1 Notation and Mathematical Formulation for Model A1

The notation used in this problem is as follows:

K : set of aircrafts

R : set of available aircrafts in the recovery period

F : set of all flight legs

F^k : set of flight legs that can be flown by aircraft k

N^k : set of flight connections for aircraft k during recovery period. (Feasible connection between two consecutive flight leg)

L^k : set of flight connections for aircraft k before the recovery period begins

M^k : set of flight connections for aircraft k after the recovery period finishes

k : index representing the aircrafts

i : index representing the first flight in the connection arc

j : index representing the next flight in the connection arc

arr_i : arrival time of flight i

$dept_j$: departure time of flight j

org_j : origin station of flight j

$dest_i$: destination station of flight i

$turnaround$: minimum required turnaround time between two consecutive flight legs

$lower$: lower bound on the number of flight legs that an aircraft can fly

$upper$: upper bound on the number of flight legs that an aircraft can fly

sch_{ij}^k : the original aircraft – flight assignment. It is equal to 1, if the aircraft k is assigned to two consecutive flights i and j in the original schedule, 0 otherwise.

X_{ij}^k : the binary decision variable which takes value 1 if the aircraft k fly two consecutive flights i and j and 0 otherwise.

Y_j : the binary decision variable which takes value 1 if the flight j is cancelled and 0 otherwise.

The mathematical formulation of the problem is as follows:

$$Minimize \sum_{k \in K} \sum_{(i,j) \in N^k \cup M^k \cup L^k} c_{ij}^k \, X_{ij}^k + \sum_{i \epsilon F} b_i \, Y_i \qquad (4.1)$$

Subject to:

$$\sum_{k\epsilon K}\sum_{j:(i,j)\in N^k\cup M^k\cup L^k} X_{ij}^k + Y_i = 1 \qquad \forall i \in F^k \tag{4.2}$$

$$X_{ij}^k = 0 \qquad \forall k \in K - \{R\}\ \forall (i,j) \in N^k \tag{4.3}$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K\ \ \forall (i,j) \in L^k \tag{4.4}$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K\ \ \forall (i,j) \in M^k \tag{4.5}$$

$$\sum_{j:(i,j)\epsilon N^k} X_{ij}^k - \sum_{j:(j,i)\epsilon N^k} X_{ji}^k = 0 \qquad \forall k \in R\ \ \forall i \in F^k \tag{4.6}$$

$$X_{ij}^k\left(arr_i + turnaround - dept_j\right) \geq 0 \quad \forall k \in R\ \ \forall (i,j) \in N^k \tag{4.7}$$

$$X_{ij}^k\left(dest_i - org_j\right) = 0 \qquad \forall k \in R\ \ \forall (i,j) \in N^k \tag{4.8}$$

$$\sum_{(i,j)\in N^k\cup M^k\cup L^k} X_{ij}^k \leq lower \qquad \forall k \in K \tag{4.9}$$

$$\sum_{(i,j)\in N^k\cup M^k\cup L^k} X_{ij}^k \geq upper \qquad \forall k \in K \tag{4.10}$$

$$X_{ij}^k \in \{0,1\} \qquad \forall k \in K\ \forall (i,j) \in N^k \cup M^k \cup L^k \tag{4.11}$$

$$Y_i \in \{0,1\} \qquad \forall i \in F \tag{4.12}$$

The objective function (4.1) minimizes the total cost of rerouting aircrafts and cancelling flights. The constraint (4.2) is the flight cover constraint and ensures that all flight legs will be either covered by an aircraft or cancelled. The constraint (4.3) ensures that during recovery period no flight will be assigned to the shortage aircraft. The constraints (4.4) and (4.5) ensure that before and after the recovery period the aircraft assignment will be same as the original schedule. In other words, it will be

returned to the original flight aircraft assignment after the shortage aircraft is recovered. The constraint (4.6) ensures that the total arc flow, which enters the node *i* must be equal to the total arc flow which leaves the node *i* for the available aircrafts during recovery period. The constraint (4.7) guarantees the time requirements between consecutive flight legs: there should be at least the minimum required turnaround time between the arrival of the flight *i* and the departure of the flight *j* for the aircrafts available during recovery period. The constraint (4.8) ensures that the destination station of the first flight and the origin of the second flight are the same for the aircrafts available during recovery period. Constraints (4.9) and (4.10) guarantees the usage balance for the aircrafts ensuring the upper and lower bounds on the number of flight legs that an aircraft can fly. Constraints (4.11) and (4.12) ensure that all the decision variables are binary.

4.1.2 Numerical Example for Model A1

The developed model is solved using OPL Studio 3.7. The OPL model of the problem can be seen in Appendix C. The problem is solved using the domestic flight schedule of OnurAir as seen in Appendix A in Table A.2.

The aircraft recovery problem is solved for 2 example cases:

In the first case, there occurs a shortage on aircraft MD88d at 16:30 causing that the MD88d can not fly the flight OHY054 that is supposed to leave station IST at 16:30 and arrive to station ADB at 17:30. The recovery period begins at 16:30 end continues until the end of the day.

In the second case there occurs a shortage on aircraft MD88e at 12:15. The recovery period begins at 12:45 and continues until 16:45. During this recovery period, Aircraft MD88e can not fly the flights OHY062 and OHY063.

The aircraft recovery solutions for these example cases can be seen in the Table 4.1. As it can be seen in the table, various aircraft swapping operations have been performed in order to fly all the scheduled flight legs.

Table 4.1 Recovered schedules obtained from the solution of Model A1

Original Schedule						Shortage occurs on Aircraft MD88d at 16:30		Shortage occurs on Aircraft MD88e at 12:15	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Aircraft	Recovery strategy	Aircraft	Recovery strategy
OHY010	IST	ADA	06:45	08:15	MD83c	MD83c	-	MD83c	-
OHY011	ADA	IST	09:10	10:40	MD83c	MD83c	-	MD83c	-
OHY016	IST	ADA	15:00	16:30	MD83c	MD83c	-	MD83c	-
OHY017	ADA	IST	17:30	19:00	MD83c	MD83c	-	MD83c	-
OHY018	IST	ADA	19:00	20:30	MD88a	MD88b	S	MD88a	-
OHY019	ADA	IST	21:30	23:00	MD88a	MD88b	S	MD88a	-
OHY022	IST	AYT	08:15	09:15	MD88c	MD88c	-	MD88c	-
OHY023	AYT	IST	10:45	11:45	MD88c	MD88c	-	MD88c	-
OHY024	IST	AYT	16:45	17:45	MD83b	MD83d	S	MD83b	-
OHY025	AYT	IST	18:45	19:45	MD83b	MD83d	S	MD83b	-
OHY026	IST	AYT	20:30	21:30	MD83d	MD88e	S	MD83d	-
OHY027	AYT	IST	07:30	08:30	MD83b	MD83b	-	MD83b	-
OHY028	IST	BJV	11:55	12:55	MD83d	MD83d	-	MD83d	-
OHY029	BJV	IST	13:45	14:45	MD83d	MD83d	-	MD83d	-
OHY034	IST	DIY	06:45	08:30	MD88a	MD88a	-	MD88a	-
OHY035	DIY	IST	09:30	11:15	MD88a	MD88a	-	MD88a	-
OHY036	IST	DIY	18:45	20:30	MD88b	MD83b	S	MD88b	-
OHY037	DIY	IST	21:30	23:15	MD88b	MD83b	S	MD88b	-
OHY038	IST	ERZ	12:30	14:20	MD88c	MD88c	-	MD88c	-
OHY039	ERZ	IST	15:20	17:10	MD88c	MD88c	-	MD88c	-
OHY042	IST	GZT	09:30	11:15	MD83a	MD83a	-	MD83a	-
OHY043	GZT	IST	12:15	14:00	MD83a	MD83a	-	MD83a	-
OHY050	IST	ADB	07:45	08:45	MD88d	MD88d	-	MD88d	-
OHY051	ADB	IST	10:50	11:45	MD88d	MD88d	-	MD88d	-
OHY054	IST	ADB	16:30	17:30	MD88d	MD83a	S	MD88d	-
OHY055	ADB	IST	18:50	19:45	MD88d	MD83a	S	MD88d	-
OHY058	IST	ADB	20:45	21:45	MD88e	MD83a	S	MD88e	-
OHY059	ADB	IST	07:30	08:25	MD83d	MD83d	-	MD83d	-
OHY062	IST	MLX	12:15	13:45	MD88e	MD88e	-	MD88b	S
OHY063	MLX	IST	14:30	16:00	MD88e	MD88e	-	MD88b	S
OHY066	IST	KSY	07:15	08:30	MD88e	MD88e	-	MD88e	-
OHY067	KSY	IST	09:15	10:30	MD88e	MD88e	-	MD88e	-
OHY074	IST	SZF	09:30	10:45	MD83b	MD83b	-	MD83b	-
OHY075	SZF	IST	11:45	13:00	MD83b	MD83b	-	MD83b	-
OHY080	IST	TZX	06:50	08:25	MD88b	MD88b	-	MD88b	-
OHY081	TZX	IST	09:20	10:55	MD88b	MD88b	-	MD88b	-
OHY086	IST	TZX	18:45	20:20	MD83a	MD88a	S	MD83a	-
OHY087	TZX	IST	21:15	22:50	MD83a	MD88a	S	MD83a	-
* Recovery Strategies : S: swapping aircrafts, D: delaying flights, C: cancelling flights									

Since in the first example the flights OHY054 and OHY055 can not be flown by Aircraft MD88d, these two flights have been assigned to Aircraft MD83a. The flights OHY086 and OHY087 that are supposed to be flown by MD83a have been assigned to Aircraft MD88a instead. Similarly the flights OHY018 and OHY019 that are

originally assigned to Aircraft MD88a, have been assigned to Aircraft MD88b. Since the flight schedule of OnurAir is not too tight, the recovery period has been handled by just swapping the aircrafts among the flights. There has been no need to cancel any flight during the absence of the Aircraft MD88d.

Since in the second example the recovery period is shorter, the recovery period has been handled by performing less swapping operations. When shortage occurs on Aircraft MD88e, the flights OHY062 and OHY063 can not be flown by this aircraft. Instead, these flights have been assigned to Aircraft MD88b which was on ground at station IST from 10:55 am to 18:45 pm. Therefore the aircraft MD88d can cover the open flights OHY062 and OHY063, leaving IST at 12:15 pm and returning to IST at 16:00 pm, without delaying its next flight at 18:45. As soon as the recovery period ends, after 16:45 airline has returned to its original schedule.

4.2 Model A2

Cancellation and Delay Model with Multiple Aircraft Unavailability Using Multi-Commodity Network Flow Models

Model A2 is developed as multi-commodity network flow model where the underlying network is connection network.

The Model A2 is the modified version of Model A1. While Model A1 considers only two recovery strategies; swapping aircrafts among flights and cancelling flights, the Model A2 will include the another recovery strategy; delaying flight legs. Another addition in this Model A2 is that the problem is now capable of considering multiple aircraft shortages. As a summary, in Model A2:

- Five different flight connection networks are used in the problem formulation, which represent
  - The flight connections before recovery period
  - The flight connections after recovery period
  - The flight connections during recovery period (on time)
  - The flight connections during recovery period with a delay of 30 minutes

  - The flight connections during recovery period with a delay of 60 minutes

- It is assumed that the maintenance service of the aircrafts is made after the midnight, when no flights are scheduled to be flown.

- Three recovery strategies are considered:
  - Cancelling flights
  - Swapping aircrafts among scheduled flights
  - Delaying flights (30 minutes or 60 minutes of delay is considered)

- It is assumed that the crews are available at any time, thus the crew availability will not affect the flight's operability.

- Multiple aircraft unavailabilities (shortages) are considered.

In order to modify the model including also the delay arcs, new network connections are defined. 30 minutes of delays and 60 minutes of delays are considered. Not only the new network connections but also decision variables are modified including following situations:

- If the first flight leg is delayed 30 minutes
- If the second flight leg is delayed 30 minutes
- If both the first and second flight legs are delayed 30 minutes
- If the first flight leg is delayed 60 minutes
- If the second flight leg is delayed 60 minutes
- If both the first and second flight legs are delayed 60 minutes
- If the first flight leg is delayed 30 minutes and the second flight leg is delayed 60 minutes
- If the first flight leg is delayed 60 minutes and the second flight leg is delayed 30 minutes

4.2.1 The Definition of Recovery Horizon in Case of Multiple Aircraft Shortages

Depending on the number of unavailable aircrafts and the unavailability time, the recovery horizon has been computed as follows:

If the unavailability time of different aircrafts intersects, the recovery horizon begins at the time when the first aircraft's unavailability begins and finishes at the time when the last aircraft's unavailability finishes. For example, the shortage on Aircraft 1 begins at 07:00 and continues until 12:30 while the shortage on Aircraft 2 begins at 09:00 and continues until 14:00. In this situation total recovery period begins at 07:00 and continues until 14:00 as seen in the Figure 4.1.

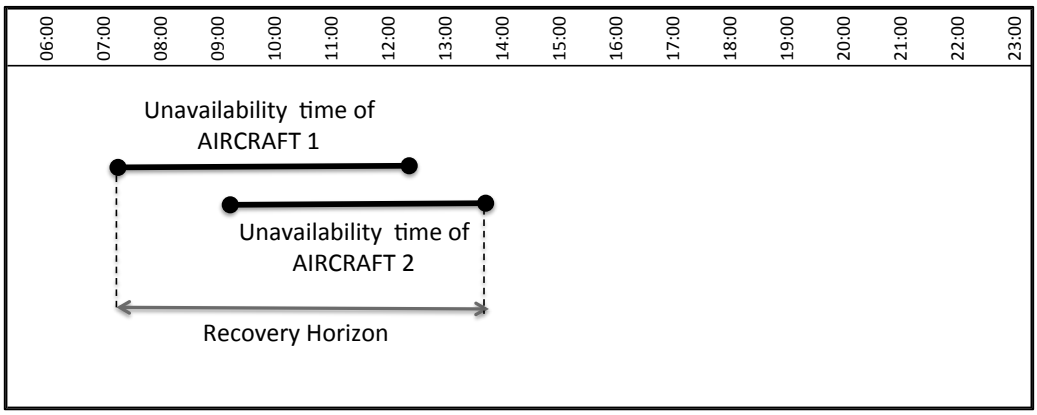

Figure 4.1 Recovery horizon for intersecting unavailability times

If the unavailability time of different aircrafts does not intersect, more than one recovery horizon may be computed depending on the time interval between two consecutive recovery horizon. If the time interval between two consecutive recovery horizon is short (eg. less than 1 hour), decision maker may consider one long combined recovery horizon. In Figure 4.2, the shortage on Aircraft begins at 07:00 and continues until 12:30 while the shortage on Aircraft 2 begins at 14:00 and continues until 18:45. In this situation the decision maker may either combine the two recovery periods and perform rescheduling between 07:00 and 18:45 or perform rescheduling for 2 different recovery periods.

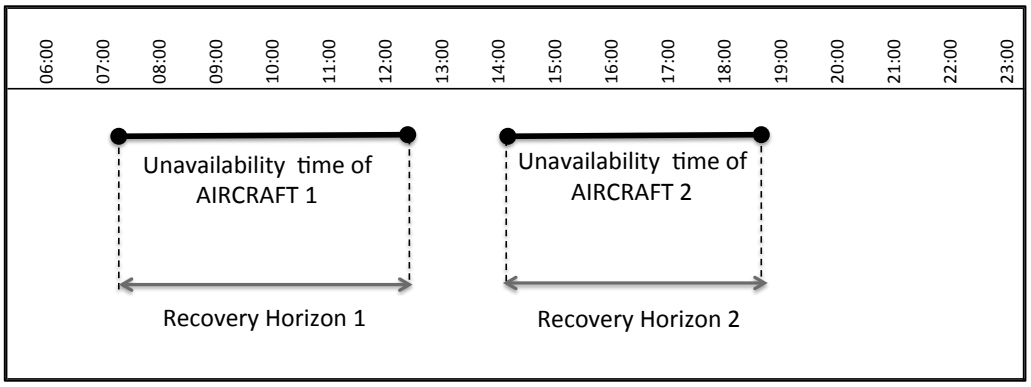

Figure 4.2 Recovery horizon for consecutive unavailability times

4.2.2 The Notation and Mathematical Formulation for Model A2

The notation used in this problem is as follows:

K : set of aircrafts

R : set of available aircrafts in the recovery period

F : set of all flight legs

F^k : set of flight legs that can be flown by aircraft k

N^k : rerouting connection network; set of flight connections for aircraft k during recovery period. (Feasible connection between two consecutive flight legs)

N_{30}^k : 30 minutes of delayed connection network; set of flight connections for aircraft k during recovery period.

N_{60}^k : 60 minutes of delayed connection network; set of flight connections for aircraft k during recovery period.

L^k : set of flight connections for aircraft k before the recovery period begins

M^k : set of flight connections for aircraft k after the recovery period finishes

k : index representing the aircrafts

i : index representing the first flight in the connection arc

j : index representing the next flight in the connection arc

arr_i : arrival time of flight i

$dept_j$: departure time of flight j

org_j : origin station of flight j

$dest_i$: destination station of flight i

$turnaround$: minimum required turnaround time between two consecutive flight legs

$lower$: lower bound on the number of flight legs that an aircraft can fly

$upper$: upper bound on the number of flight legs that an aircraft can fly

sch_{ij}^k : the original aircraft – flight assignment. It is equal to 1, if the aircraft k is assigned to two consecutive flights i and j in the original schedule, 0 otherwise.

c_{ij}^k : the cost of rerouting aircraft k between consecutive flights i and j.

d_{ij}^k : the cost of delaying aircraft k between consecutive flights i and j.

X_{ij}^k : the binary decision variable which takes value 1 if the aircraft k fly two consecutive flights i and j and 0 otherwise.

Y_j : the binary decision variable which takes value 1 if the flight j is cancelled and 0 otherwise.

The mathematical model of the aircraft recovery problem is as follows:

$$Minimize \sum_{k\in K}\sum_{(i,j)\in N^k\cup M^k\cup L^k} c_{ij}^k X_{ij}^k + \sum_{k\in K}\sum_{(i,j)\in N_{30}^k\cup N_{60}^k} d_{ij}^k X_{ij}^k + \sum_{i\in F} b_i Y_i \qquad (4.13)$$

Subject to:

$$\sum_{k\in K}\sum_{j:(i,j)\in N^k\cup M^k\cup L^k} X_{ij}^k + \sum_{k\in K}\sum_{j:(i,j)\in N_{30}^k\cup N_{60}^k} X_{ij}^k + Y_i = 1 \qquad \forall i \in F^k \qquad (4.14)$$

$$X_{ij}^k = 0 \qquad \forall k \in K-\{R\} \quad \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \qquad (4.15)$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K \ \forall (i,j) \in L^k \qquad (4.16)$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K \ \forall (i,j) \in M^k \qquad (4.17)$$

$$\sum_{j:(i,j)\epsilon N^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k - \sum_{j:(j,i)\epsilon N^k \cup N_{30}^k \cup N_{60}^k} X_{ji}^k = 0 \qquad \forall k \in R \;\; \forall i \in F^k \qquad (4.18)$$

$$X_{ij}^k\left(arr_i + turnaround - dept_j\right) \geq 0 \qquad \forall k \in R \;\; \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \qquad (4.19)$$

$$X_{ij}^k\left(dest_i - org_j\right) = 0 \qquad \forall k \in R \;\; \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \qquad (4.20)$$

$$\sum_{(i,j)\in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k \leq lower \qquad \forall k \in K \qquad (4.21)$$

$$\sum_{(i,j)\in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k \geq upper \qquad \forall k \in K \qquad (4.22)$$

$$X_{ij}^k \epsilon \{0,1\} \qquad \forall k \in K \;\; \forall (i,j) \in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k \qquad (4.23)$$

$$Y_i \epsilon \{0,1\} \qquad \forall i \in F \qquad (4.24)$$

where the objective function (4.13) minimizes the total cost of rerouting aircrafts, delaying flights and cancelling flights. The constraint (4.14) is the flight cover constraint and ensures that all flight legs will be either covered by an aircraft (delayed or on time) or cancelled. This constraint also ensures that flight legs can be either a member of rerouting connection network or delay connection network. With these constraints also logical connection between non-delayed and delayed flights is developed. For example if the flight *j* is delayed 30 minutes in route (i, j), flight *j* should also be found 30 minutes delayed in its next connection. The constraint (4.15) ensures that during recovery period no flights will be assigned to the shortage aircrafts. The constraints (4.16) and (4.17) ensure that before and after the recovery period the aircraft assignment will be same as the original schedule. In other words, it will be returned to the original flight aircraft assignment after the shortage aircrafts is recovered. The constraint (4.18) is the flow conservation constraint, which ensures that the total arc flow which, enters the node *i* must be equal to the total arc flow which leaves the node i for the available aircrafts during recovery period. The constraint (4.19) guarantees the time requirements between consecutive flight legs:

there should be at least the minimum required turnaround time between the arrival of the flight *i* and the departure of the flight *j* for the aircrafts available during recovery period. The constraint (4.20) ensures that the destination station of the first flight and the origin of the second flight are the same for the aircrafts available during recovery period. Constraints (4.21) and (4.22) guarantees the usage balance for the aircrafts ensuring the upper and lower bounds on the number of flight legs that an aircraft can fly. Constraints (4.23) and (4.24) ensure that all the decision variables are binary.

4.2.3 The Numerical Example for Model A2

The developed model is solved using OPL Studio 3.7. The OPL model of the problem can be seen in Appendix D. The problem is solved using the domestic flight schedule of OnurAir as seen in Appendix A in Table A.2.

Several cases for different recovery period horizons chosen randomly have been tested in order to evaluate the efficiency of the Model A2. However, only two of the solutions are presented as example cases.

The first example is solved considering the 2 aircrafts' shortages at different times as follows:

- There occurs a shortage on aircraft MD83b at 09:30 causing that the MD83b can not fly the flight OHY074 that is supposed to leave station IST at 09:30 and arrive at station SZF at 10:45. The shortage continues until 12:00. During this period the MD83b can not fly the flight OHY075 either.

- There occurs another shortage at MD88c at 12:30 causing that the MD88c can not fly the flight OHY038 that is supposed to leave station IST at 12:30 and arrive at station ERZ at 14:20. The shortage continues until 17:30. During this period the MD88c can not fly the flight OHY039 either.

- Since the time interval between two shortages is only half an hour, the individual recovery periods of two aircrafts are combined. The recovery period

begins at 09:30 when the first aircraft (MD83b) becomes unavailable and finishes at 17:30 when the second aircraft (MD88c) becomes available again.

The second example problem is also solved considering the 2 aircrafts' shortages at different times as follows:

- There occurs a shortage on aircraft MD88e at 07:10 causing that the MD88e can not fly the flight OHY066 that is supposed to leave station IST at 07:15 and arrive at station KSY at 08:30. The shortage continues until 12:00. During this period the MD83e can not fly the flight OHY067 either.

- There occurs another shortage at MD88c at 12:30 causing that the MD88c can nor fly the flight OHY038 that is supposed to leave station IST at 12:30 and arrive at station ERZ at 14:20. The shortage continues until 17:30. During this period the MD88c can not fly the flight OHY039 either.

- As in the first example case, the time interval between two shortages is only half an hour. Therefore, the individual recovery periods of two aircrafts are combined. The recovery period begins at 07:10 when the first aircraft (MD88e) becomes unavailable and finishes at 17:30 when the second aircraft (MD88c) becomes available again.

The solution obtained from the solution of Model A2 can be seen in Table 4.2. As it can be seen from the table, the aircrafts in the recovery period have been rescheduled without cancelling and delaying any flight leg. Since the flight schedule of OnurAir is not too tight, the remaining/available aircrafts can cover all the flight legs without delaying any of them. Instead, various aircraft swapping operations have been performed in order to fly all the scheduled flight legs.

Table 4.2 Recovered schedules obtained from the solution of Model A2

Original Schedule						Shortage occurs on Aircraft MD83b at 09:30 and on Aircraft MD88c at 12:30		Shortage occurs on Aircraft MD88e at 07:10 and on Aircraft MD88c at 12:30	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Aircraft	Recovery strategy	Aircraft	Recovery strategy
OHY010	IST	ADA	06:45	08:15	MD83c	MD83c	-	MD83c	-
OHY011	ADA	IST	09:10	10:40	MD83c	MD83c	-	MD83c	-
OHY016	IST	ADA	15:00	16:30	MD83c	MD83c	-	MD83c	-
OHY017	ADA	IST	17:30	19:00	MD83c	MD83c	-	MD83c	-
OHY018	IST	ADA	19:00	20:30	MD88a	MD88a	-	MD88a	-
OHY019	ADA	IST	21:30	23:00	MD88a	MD88a	-	MD88a	-
OHY022	IST	AYT	08:15	09:15	MD88c	MD88c	-	MD88c	-
OHY023	AYT	IST	10:45	11:45	MD88c	MD88c	-	MD88c	-
OHY024	IST	AYT	16:45	17:45	MD83b	MD83b	-	MD83b	-
OHY025	AYT	IST	18:45	19:45	MD83b	MD83b	-	MD83b	-
OHY026	IST	AYT	20:30	21:30	MD83d	MD83d	-	MD83d	-
OHY027	AYT	IST	07:30	08:30	MD83b	MD83b	-	MD88d	S
OHY028	IST	BJV	11:55	12:55	MD83d	MD88b	S	MD88b	S
OHY029	BJV	IST	13:45	14:45	MD83d	MD88b	S	MD88b	S
OHY034	IST	DIY	06:45	08:30	MD88a	MD88a	-	MD88a	-
OHY035	DIY	IST	09:30	11:15	MD88a	MD88a	-	MD88a	-
OHY036	IST	DIY	18:45	20:30	MD88b	MD88b	-	MD88b	-
OHY037	DIY	IST	21:30	23:15	MD88b	MD88b	-	MD88b	-
OHY038	IST	ERZ	12:30	14:20	MD88c	MD88e	S	MD83d	S
OHY039	ERZ	IST	15:20	17:10	MD88c	MD88e	S	MD83d	S
OHY042	IST	GZT	09:30	11:15	MD83a	MD83d	S	MD88e	S
OHY043	GZT	IST	12:15	14:00	MD83a	MD83d	S	MD88e	S
OHY050	IST	ADB	07:45	08:45	MD88d	MD88d	-	MD83d	S
OHY051	ADB	IST	10:50	11:45	MD88d	MD88d	-	MD83d	S
OHY054	IST	ADB	16:30	17:30	MD88d	MD88d	-	MD88d	-
OHY055	ADB	IST	18:50	19:45	MD88d	MD88d	-	MD88d	-
OHY058	IST	ADB	20:45	21:45	MD88e	MD88e	-	MD88e	-
OHY059	ADB	IST	07:30	08:25	MD83d	MD83d	-	MD83b	S
OHY062	IST	MLX	12:15	13:45	MD88e	MD83b	S	MD83b	S
OHY063	MLX	IST	14:30	16:00	MD88e	MD83b	S	MD83b	S
OHY066	IST	KSY	07:15	08:30	MD88e	MD88e	-	MD83a	S
OHY067	KSY	IST	09:15	10:30	MD88e	MD88e	-	MD83a	S
OHY074	IST	SZF	09:30	10:45	MD83b	MD83a	S	MD88d	S
OHY075	SZF	IST	11:45	13:00	MD83b	MD83a	S	MD88d	S
OHY080	IST	TZX	06:50	08:25	MD88b	MD88b	-	MD88b	-
OHY081	TZX	IST	09:20	10:55	MD88b	MD88b	-	MD88b	-
OHY086	IST	TZX	18:45	20:20	MD83a	MD83a	-	MD83a	-
OHY087	TZX	IST	21:15	22:50	MD83a	MD83a	-	MD83a	-
* Recovery Strategies : S: swapping aircrafts, D: delaying flights, C: cancelling flights									

In the solution of the first example;

- Since the flights OHY074 and OHY075 can not be flown by Aircraft MD83b, these two flights have been assigned to Aircraft MD83a.

- Since the flights OHY038 and OHY039 can not be flown by Aircraft MD88c, these two flights have been assigned to Aircraft MD88e.

- Several aircraft swappings have been performed between the flight legs OHY028, OHY029, OHY042, OHY043, OHY062 and OHY063.

In the solution of the second example;

- Since the flights OHY066 and OHY067 can not be flown by Aircraft MD88e, these two flights have been assigned to Aircraft MD83a.

- Since the flights OHY038 and OHY039 can not be flown by Aircraft MD88c, these two flights have been assigned to Aircraft MD83d.

- Several aircraft swappings have been performed between the flight legs OHY027, OHY028, OHY029, OHY042, OHY043, OHY050, OHY051, OHY059, OHY062, OHY063, OHY074 and OHY075.

4.3 Model A3

Delay and Cancellation Model with Multiple Aircraft Unavailability Including Crew Considerations using Multi-Commodity Network Flow Problem

Model A3 is developed as multi-commodity network flow model where the underlying network is connection network.

Model A3 is the modified version of Model A2. The Model A3 considers three recovery strategies; delaying flights, cancelling flights and swapping aircrafts among flights, as in Model A2. Model A3 is also capable of solving aircraft recovery problems in case of multiple aircraft unavailabilities as in the Model A2. Different from Model A2, Model A3 includes crew related constraints. While in Model A1 and Model A2, it is assumed that the crews are available at any time, and their availability does not affect the recovery process, in Model A3 crew related constraints directly affect the recovery process and if there is no available crew for a flight leg, the subject flight leg can not be operated. As a summary, in Model A3:

- Five different flight connection networks are used in the problem formulation, which represent
  - The flight connections before recovery period
  - The flight connections after recovery period
  - The flight connections during recovery period (on time)
  - The flight connections during recovery period with a delay of 30 minutes
  - The flight connections during recovery period with a delay of 60 minutes

- It is assumed that the maintenance service of the aircrafts is made after the midnight, when no flights are scheduled to be flown.

- Three recovery strategies are considered:
  - Cancelling flights
  - Swapping aircrafts among scheduled flights
  - Delaying flights (30 minutes or 60 minutes of delay is considered)

- The crew availability constraints are included in the model. Thus if there is not available crew to cover a light leg, the subject flight leg may be cancelled.

- Multiple aircraft unavailabilities (shortages) are considered.

Crew availability directly affects the airline operations. There should be always sufficient number of crews in order to fly all scheduled flight legs. Although there is an available aircraft to be assigned a scheduled flight leg, the flight can not be operated until the available crew is found to fly it. Thus, any aircraft recovery problem is considered uncompleted until the aircrew availability is included into the problem.

So far in the developed aircraft recovery problems, it has always been assumed that the aircrews are available at any time. In this revised model (Model A3) it is assumed that when there is a shortage in any aircraft and it can not fly a certain flight leg, the crew assigned to these certain flight leg will not be available. There will be

shortage for both aircraft and aircrew. Even if another aircraft is assigned to subject flight leg for recovery purposes, the subject flight legs may be cancelled due to lack of available crews.

In order to include crew considerations into the problem, the Model A2 will be modified by taking into account also crew availability constraints and the initial crew schedule needs to be used as input to the aircraft recovery problem. The crew schedule of OnurAir domestic flights can be seen in Appendix A in Table A.2. By using this crew schedule, the number of available crews at a certain time in a certain station is determined. In other words, for each flight leg to be flown, the number of available crews is determined according to departure time and origin station of each flight leg. While determining the number of available crew, the following issues are considered:

- The crew already assigned to the subject flight leg
- The crews positioned on ground at departure time of the subject flight leg, where the positioned station is the origin station of the subject flight leg.

The computed number of crews for each flight leg can be seen in Appendix A in Table A.4.

4.3.1 The Notation and Mathematical Formulation of Model A3

The notation and the mathematical formulation of Model A3 are similar to the Model A2 which has been explained in Section 4.2.2 on pages 105-108. Therefore, the objective function (4.13) and the problem constraints (4.14 - 4.24) will remain the same. But some additional constraints will be added into the problem which controls the availability of the crews:

$$Xac_i^k = ac_i^k \qquad \forall k \in K, \quad \forall\, i: (i,j) \in L^k \cup M^k \qquad (4.25)$$

$$Xac_i^k = ac_i^k - 1 \qquad \forall k \in K - \{R\}, \quad \forall\, i: (i,j) \in N^k \qquad (4.26)$$

$$\sum_{k \in K-\{R\}} \sum_{j:(i,j) \in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k \leq Xac_i^k \qquad \forall i \in F^k \qquad (4.27)$$

where;

ac_i^k : the number of available crew for the flight leg i flown by aircraft k .

Xac_i^k : the decision variable that determines the number of available crew for the flight leg i flown by aircraft k.

The constraint (4.25) determines the number of available crews for the flight legs to be flown before and after the recovery period. Before and after recovery period there will not be any shortage for aircrafts and crews, so the number of the available crews for the flight legs scheduled in these periods will be equal to the number of aircrafts seen in Appendix A in Table A.4.

The constraint (4.26) determines the number of available crews for the flight legs to be flown by the shortage aircraft during recovery period. As aircrafts are unavailable for the flights in recovery period, the originally assigned crew also becomes unavailable for such flight legs. Thus, the number of available crews for these flight legs will be 1 less than the number of available crews seen in Appendix A in Table A.4.

The constraint (4.27) ensures that if there is not sufficient number of crews for flying any flight legs, this flight leg can not be flown. Thus, the decision variables for the aircraft- flight leg assignment will take value 0.

4.3.2 Numerical Example for Model A3

The developed model is solved using OPL Studio 3.7. The OPL model of the problem can be seen in Appendix E. The problem is solved using the domestic flight schedule of OnurAir as seen in Appendix A in Table A.2.

The same cases, which have been solved with aircraft recovery model A2 in Section 4.2.3 on pages 108-111, is solved with Model A3 in order to compare the models. Therefore, the same example cases are solved with and without crew considerations and it can be seen that how the crew availability affects the flights operability.

The first example case in Section 4.2.3 using the Model A2 is as follows:

- There occurs a shortage on aircraft MD83b at 09:30 causing that the MD83b can not fly the flight OHY074 that is supposed to leave station IST at 09:30 and arrive at station SZF at 10:45. The shortage continues until 12:00. During this period the MD83b can not fly the flight OHY075 either.

- There occurs another shortage at MD88c at 12:30 causing that the MD88c can nor fly the flight OHY038 that is supposed to leave station IST at 12:30 and arrive at station ERZ at 14:20. The shortage continues until 17:30. During this period the MD88c can not fly the flight OHY039 either.

- The recovery period begins at 09:30 when the first aircraft (MD83b) becomes unavailable and finishes at 17:30 when the second aircraft (MD88c) becomes available again.

The second example case in Section 4.2.3 using the Model A2 is as follows:

- There occurs a shortage on aircraft MD88e at 07:10 causing that the MD88e can not fly the flight OHY066 that is supposed to leave station IST at 07:15

and arrive at station KSY at 08:30. The shortage continues until 12:00. During this period the MD83e can not fly the flight OHY067 either.

- There occurs another shortage at MD88c at 12:30 causing that the MD88c can nor fly the flight OHY038 that is supposed to leave station IST at 12:30 and arrive at station ERZ at 14:20. The shortage continues until 17:30. During this period the MD88c can not fly the flight OHY039 either.

- The recovery period begins at 07:10 when the first aircraft (MD88e) becomes unavailable and finishes at 17:30 when the second aircraft (MD88c) becomes available again.

The solutions of the example problems can be seen in Table 4.3.

In the first example problem, no flight legs have to be cancelled due to lack of available crew:

- There are 2 available crews for flight legs OHY074 and OHY075 (1 already assigned, 1 resting at station). When the original crew becomes unavailable to fly the flights OHY074 and OHY075, there are one more substitute crew to fly these flight legs.

- There are 5 available crews OHY038 and OHY039 (1 already assigned, 4 resting at station). When the original crew becomes unavailable to fly the flights OHY038 and OHY039, there are 4 more substitute crews to fly these flight legs.

In the second example problem, the flight legs OHY066 and OHY067 have to be cancelled. Because there is no available crew to cover these flight legs.

- There is only 1 available crew for flight legs OHY066 and OHY067 (1 already assigned, there is no substitute crew). When the original crew becomes unavailable to fly the flights OHY066 and OHY067, there will not be any

substitute crew to fly these flight legs and these flight legs will be cancelled due to lack of crew availability.

- There are 5 available crews OHY038 and OHY039 (1 already assigned, 4 resting at station). When the original crew becomes unavailable to fly the flights OHY038 and OHY039, there are 4 more substitute crews to fly these flight legs.

Table 4.3 Recovered schedules obtained from the solution of Model A3

Original Schedule						Shortage occurs on Aircraft MD83b at 09:30 and on Aircraft MD88c at 12:30		Shortage occurs on Aircraft MD88e at 07:10 and on Aircraft MD88c at 12:30	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Aircraft	Recovery strategy	Aircraft	Recovery strategy
OHY010	IST	ADA	06:45	08:15	MD83c	MD83c	-	MD83c	-
OHY011	ADA	IST	09:10	10:40	MD83c	MD83c	-	MD83c	-
OHY016	IST	ADA	15:00	16:30	MD83c	MD83c	-	MD83c	-
OHY017	ADA	IST	17:30	19:00	MD83c	MD83c	-	MD83c	-
OHY018	IST	ADA	19:00	20:30	MD88a	MD88a	-	MD88a	-
OHY019	ADA	IST	21:30	23:00	MD88a	MD88a	-	MD88a	-
OHY022	IST	AYT	08:15	09:15	MD88c	MD88c	-	MD88c	-
OHY023	AYT	IST	10:45	11:45	MD88c	MD88c	-	MD88c	-
OHY024	IST	AYT	16:45	17:45	MD83b	MD83b	-	MD83b	-
OHY025	AYT	IST	18:45	19:45	MD83b	MD83b	-	MD83b	-
OHY026	IST	AYT	20:30	21:30	MD83d	MD83d	-	MD83d	-
OHY027	AYT	IST	07:30	08:30	MD83b	MD83b	-	MD83d	S
OHY028	IST	BJV	11:55	12:55	MD83d	MD88b	S	MD83b	S
OHY029	BJV	IST	13:45	14:45	MD83d	MD88b	S	MD83b	S
OHY034	IST	DIY	06:45	08:30	MD88a	MD88a	-	MD88a	-
OHY035	DIY	IST	09:30	11:15	MD88a	MD88a	-	MD88a	-
OHY036	IST	DIY	18:45	20:30	MD88b	MD88b	-	MD88b	-
OHY037	DIY	IST	21:30	23:15	MD88b	MD88b	-	MD88b	-
OHY038	IST	ERZ	12:30	14:20	MD88c	MD88e	S	MD83a	S
OHY039	ERZ	IST	15:20	17:10	MD88c	MD88e	S	MD83a	S
OHY042	IST	GZT	09:30	11:15	MD83a	MD83d	S	MD88d	S
OHY043	GZT	IST	12:15	14:00	MD83a	MD83d	S	MD88d	S
OHY050	IST	ADB	07:45	08:45	MD88d	MD88d	-	MD83a	S
OHY051	ADB	IST	10:50	11:45	MD88d	MD88d	-	MD83a	S
OHY054	IST	ADB	16:30	17:30	MD88d	MD88d	-	MD88d	-
OHY055	ADB	IST	18:50	19:45	MD88d	MD88d	-	MD88d	-
OHY058	IST	ADB	20:45	21:45	MD88e	MD88e	-	MD88e	-
OHY059	ADB	IST	07:30	08:25	MD83d	MD83d	-	MD88d	S
OHY062	IST	MLX	12:15	13:45	MD88e	MD83b	S	MD88e	-
OHY063	MLX	IST	14:30	16:00	MD88e	MD83b	S	MD88e	-
OHY066	IST	KSY	07:15	08:30	MD88e	MD88e	-	Cancelled	C
OHY067	KSY	IST	09:15	10:30	MD88e	MD88e	-	Cancelled	C
OHY074	IST	SZF	09:30	10:45	MD83b	MD83a	S	MD83d	S
OHY075	SZF	IST	11:45	13:00	MD83b	MD83a	S	MD83d	S
OHY080	IST	TZX	06:50	08:25	MD88b	MD88b	-	MD88b	-
OHY081	TZX	IST	09:20	10:55	MD88b	MD88b	-	MD88b	-
OHY086	IST	TZX	18:45	20:20	MD83a	MD83a	-	MD83a	-
OHY087	TZX	IST	21:15	22:50	MD83a	MD83a	-	MD83a	-

* Recovery Strategies : S: swapping aircrafts, D: delaying flights, C: cancelling flights

The first example has given the same results with the problem solved with Model A2 in section 4.2.2. Because there is not a crew bottleneck for the flight legs originally assigned to shortage aircrafts.

The second example has given different results from the problem solved with model A2 in section 4.2.2. While solving with Model A2 the flight legs OHY066 and OHY067 have been assigned to different aircrafts since the crew considerations are not included in the model. However, there is a crew bottleneck for the flight legs OHY066 and OHY067 and subject flight legs have to be cancelled when solved with Model A3.

4.4 Model A4

Aircraft Recovery Problem Using Constraint Programming Embedded Solution Approach

Model A4 is developed with using the constraint programming technique instead of traditional multi-commodity network models. In Model A4:

- Three recovery strategies are considered:
  - Cancelling flights
  - Swapping aircrafts among scheduled flights
  - Delaying flights (until 90 minutes)

- The crew availability constraints are included in the model. Thus if there is not available crew to cover a light leg, the subject flight leg may be cancelled.

- Multiple aircraft unavailabilities (shortages) are considered.

- It is assumed that the maintenance service of the aircrafts is made after the midnight, when no flights are scheduled to be flown.

Constraint programming technique reduces the negative effects coming from the drawbacks of the traditional methods. Constraint programming is a powerful tool that

can be used as an alternative method to mathematical programming techniques and offers a more flexible modeling framework than mathematical programming. When applying constraint programming technique to the aircraft recovery problem, the modeling flexibility of constraint programming make easier to find feasible and effective aircraft reassignments in real time. When well modeled, constraint programming technique makes possible to generate huge number of feasible aircraft routes in seconds without spending too much effort, even if the number of flight legs is large.

Modeling constraint programming is less complex than developing network flow models. While in network flow problems, it is difficult to satisfy all constraints and to find feasible solutions without using complex solution techniques, the constraint programming approach can generate feasible routes with respect to all constraints and does not require complex solution techniques. Furthermore, the constraint programming approach is very adaptable in case of schedule changes. The developed constraint programming model is not airline dependent and it can be easily adapted to different airlines' schedule without much effort. Due to such benefits and advantages this dissertation uses the constraint programming involved solution procedure for aircraft recovery problems.

The proposed solution approach for aircraft recovery problems includes both integer programming and constraint programming techniques. The overall solution algorithm is divided into two main problems.

- First one is the constraint programming problem which finds all the feasible routes for each aircraft covering each flight leg to be flown during recovery period.

- The second one is the integer programming problem which selects the best set of routes minimizing the recovery costs and the deviation from the original schedule.

The solution procedure is as follows:

1. Aircraft routing problem is solved using constraint programming in order to provide initial conditions of the problem.

2. The initial aircraft-flights assignments are defined as input data for aircraft recovery problem.

3. The first-stage problem which is constraint programming based model has been developed for generating routes for each aircraft including each flight leg scheduled to be flown during recovery period. (Details are in Section 4.4.1 First-stage problem- Constraint programming)

4. Newly generated aircraft routes are sent to second-stage problem, which selects the optimum set of aircrafts routes while satisfying recovery period start/ end conditions and minimizing the recovery costs. (Details are in Section 4.4.2 Second-stage problem- Integer programming)

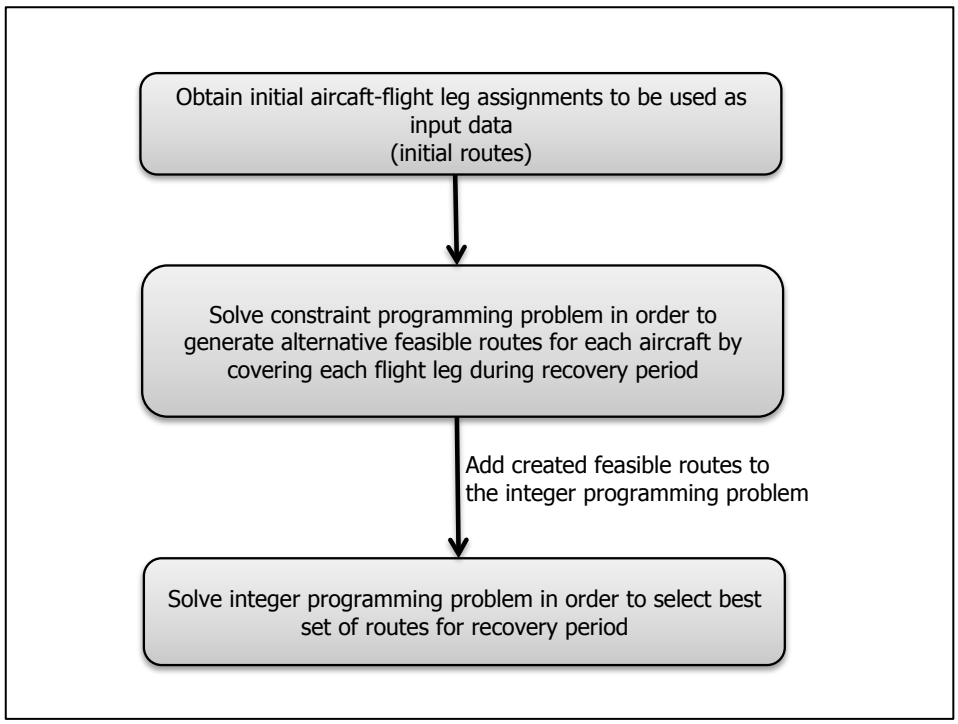

Figure 4.3 Solution Procedure Steps for Model A4

In order to provide interaction between two stages (constraint programming part of the problem and inter programming part of the problem), a script code is written using OPL Script tool. The script code of the problem can be seen in Appendix F1.

The written script code first creates input data of initial aircraft-flight assignments. Then it calls the constraint programming model sequentially for each flight leg in the recovery period. This model generates feasible routes for each aircraft and for every selected flight leg (flight legs in the recovery period). When the routes are generated for each aircraft and for all selected flight legs, the script calls the second-stage problem-integer programming model. This model selects the optimum set of routes while minimizing the changes in original schedule.

4.4.1 First Stage - Constraint Programming Problem for Model A4

As mentioned earlier, the constraint programming model generates feasible routes sequentially for each aircraft and for every selected flight leg (flight legs in the recovery period). For each aircraft the routes including 4 flight legs and 6 flight legs are considered.

The constraint programming model is called for each flight leg in the recovery period. The variable “cover” representing flight legs during recovery period is imported from OPL Script to the constraint programming model for each flight leg in the recovery period.

The notation used in the problem is as follows:

F : set of flight legs

R : set of flight legs which constitutes a legal route

A : set of aircrafts

i : the index representing the flight leg position in a route

a : the index representing the aircrafts

f : the index representing the flight legs

$cover$: the variable represents the flight legs during recovery period (transferred from OPL script to the constraint programming problem)

$dest_f$: the destination station of flight leg f

org_f : the origin station of flight leg f

arr_f : the scheduled arrival time of flight leg f

$dept_f$: the scheduled departure time of flight leg f

$AcArr_f$: the actual arrival time of flight leg f

$AcDept_f$: the actual departure time of flight leg f

$Delay_f$: the delay time of flight leg f

$MinStop$: the minimum required turnaround time between two consecutive flight legs.

$CitySeq_{a,i}$: the origin station of sequence i in the route of aircraft a .

$Seq_{a,i}$: the flight leg of sequence i in the route of aircraft a .

The mathematical formulation of the model is as follows:

$$CitySeq_{a,i-1} = dest_{seq_{a,i}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.28)$$

$$CitySeq_{a,i} = org_{seq_{a,i}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.29)$$

$$CitySeq_{a,i-1} = CitySeq_{a,i} \qquad \forall a \in A, \ \forall i \in R \qquad (4.30)$$

$$AcArr_{seq_{a,i}} + MinStop \leq AcDept_{seq_{a,i+1}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.31)$$

$$Delay_{seq_{a,i}} = AcDept_{seq_{a,i+1}} - dept_{seq_{a,i+1}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.32)$$

$$Delay_{seq_{a,i}} \leq 90 \qquad \forall a \in A, \ \forall i \in R \qquad (4.33)$$

$$CitySeq_{a,0} = CitySeq_{a,n} \qquad \forall a \in A, \quad n = 4 \; or \; n = 6 \qquad (4.34)$$

$$\left(\sum_{i \in R} seq_{a,i} = f \right) \leq 1 \qquad \forall a \in A, \quad \forall f \in F \qquad (4.35)$$

$$\left(\sum_{i \in R} seq_{a,i} = cover \right) = 1 \qquad \forall a \in A \qquad (4.36)$$

Constraints (4.28 - 4.30) ensure that the destination station of the preceding flight leg will be same as the origin station of the subsequent flight leg in the flight sequence. Constraint (4.31) satisfies that there should be at least minimum turnaround time between two consecutive flight legs. Constraint (4.32) computes the delay time for each flight leg. Constraint (4.33) limits the total delay time for each flight leg. Constraint (4.34) satisfies that the aircraft route should begin and end at the same station. Constraint (4.35) satisfies that same flight leg will not be included in a route. Constraint (4.36) ensures that one particular flight (one of the flight legs scheduled to be flown in the recovery horizon) is included in the route for each aircraft.

4.4.2 Second Stage - Integer Programming Problem for Model A4

After the completion of aircraft route generation during recovery period, OPL script calls the integer programming model. This model selects the optimum set of routes while:

- minimizing the changes in original schedule (minimizing the deviation between original schedule and recovered schedule)
- minimizing the total cancellation costs
- minimizing the total delay costs
- minimizing the total aircraft swapping costs
- satisfying that before and after recovery periods aircrafts will return their original schedule

The notation used in this problem is as follows:

K : set of aircrafts

R : set of available aircrafts in the recovery period

F : set of all flight legs

F_r^k : set of flight legs that can be flown by aircraft k in the recovery period

k : index representing the aircrafts

i : index representing the flight legs

A_{ki} : the original aircraft-flight assignment. It is equal to 1, if the aircraft k is assigned to flights i in the original schedule, 0 otherwise.

C_{ki} : the cost of reassigning aircraft k to flight i.

AC_{ki} : the number of available crew for the flight leg i flown by aircraft k.

W : the coefficient for minimizing the deviation between original schedule and recovered schedule.

D_i : the cost of cancelling flight i.

DC_i : the cost of delaying flight i (in terms of delay minutes)

X_{ki} : the binary decision variable which takes value 1 if the aircraft k is assigned to flight i and 0 otherwise.

Y_i : the binary decision variable which takes value 1 if the flight i is cancelled and 0 otherwise.

$recStart$: Recovery period start time

$recFinish$: Recovery period finish time

The mathematical formulation of the model is as follows:

$$Minimize \quad \sum_{k\in R}\sum_{i\in F_r^k} C_{ki}X_{ki} + \sum_{k\in K}\sum_{i\in F_r^k} DC_iX_{ki} + \sum_{k\in R}\sum_{i\in F} W[X_{ki} - A_{ki}] + \sum_{i\in F} D_iY_i \tag{4.37}$$

Subject to:

$$\sum_{k \in K} X_{ki} + Y_i = 1 \qquad \forall i \in F \tag{4.38}$$

$$\sum_{i \in F_r^k} X_{ki} = 1 \qquad \forall k \in R \tag{4.39}$$

$$\sum_{i \in F_r^k} X_{ki} \leq AC_{ki} \qquad \forall k \in R \tag{4.40}$$

$$X_{ki} = A_{ki} \qquad \forall k \in K - \{R\} \qquad \forall i \in F : dept_i \leq recStart \,\&\, arr_i \geq recFinish \tag{4.41}$$

$$X_{ki} = 0 \qquad \forall k \in R \qquad \forall i \in F : dept_i \geq recStart \,\&\, arr_i \leq recFinish \tag{4.42}$$

$$X_{ik} \in \{0,1\} \qquad \forall k \in K \qquad \forall i \in F \tag{4.43}$$

$$Y_i \in \{0,1\} \qquad \forall i \in F \tag{4.44}$$

The objective function (4.37) minimizes the total cost of reassigning aircrafts to flight legs, delaying flight legs, cancelling flight legs and deviation between original schedule and recovered schedule. The constraint (4.38) ensures that all flight legs will be either covered by an aircraft or cancelled. The constraint (4.39) ensures that all available aircrafts, which are assigned to the flights in the initial aircraft schedule, are assigned to flights during recovery period. The constraint (4.40) is the crew availability constraint. If there is not sufficient number of crews for flying any flight legs, this flight leg can not be flown. Thus, the decision variables for the aircraft-flight leg assignment will take value 0. The constraint (4.41) ensures that before and after the recovery period the aircraft-flight assignment will be same as the original schedule. In other words, it will be returned to the original aircraft- flight assignment after the shortage aircraft becomes available again. Constraint (4.42) ensures that no

flight will be assigned to shortage aircraft during recovery period. Constraints (4.43) and (4.44) ensure that all decision variables are binary.

4.4.3 Numerical Example for Model A4

The developed model is solved using OPL Studio 3.7. The OPL script of the problem can be seen in Appendix F1. The OPL model of the first-stage constraint programming model can be seen in Appendix F2. The OPL model of the second-stage integer programming model can be seen in Appendix F3. The problem is solved using the domestic flight schedule of OnurAir as seen Appendix A in Table A.3.

Several cases for different recovery period horizons chosen randomly have been tested in order to evaluate the efficiency of the approach. However, only two of the solutions are presented as example cases.

The first example considers the recovery period between 06:40 and 12:00 and the shortage on aircraft MD88a. Thus MD88a can not fly flights OHY034 (which is supposed to leave station IST at 6:45 and arrive at station DIY at 08:30) and OHY035 (which is supposed to leave station DIY at 9:30 and arrive at station IST at 11:15).

The second example considers the recovery period between 12:00 and 20:00 and the shortage on aircraft MD88c. Thus MD88c can not fly flights OHY038 (which is supposed to leave station IST at 12:30 and arrive at station ERZ at 14:20) and OHY039 (which is supposed to leave station ERZ at 15:20 and arrive at station IST at 17:10).

Table 4.4 Recovered schedules obtained from the solution of Model A4

Original Schedule						Shortage occurs on Aircraft MD88a at 06:40		Shortage occurs on Aircraft MD88c at 12:00	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Aircraft	Recovery strategy	Aircraft	Recovery strategy
OHY010	IST	ADA	06:45	08:15	MD88d	MD88d	-	MD88d	-
OHY011	ADA	IST	09:10	10:40	MD88d	MD88d	-	MD88d	-
OHY016	IST	ADA	15:00	16:30	MD88a	MD88a	-	MD88a	-
OHY017	ADA	IST	17:30	19:00	MD88a	MD88a	-	MD88a	-
OHY018	IST	ADA	19:00	20:30	MD83c	MD83c	-	MD83c	-
OHY019	ADA	IST	21:30	23:00	MD83c	MD83c	-	MD83c	-
OHY022	IST	AYT	08:15	09:15	MD88b	MD88b	-	MD88b	-
OHY023	AYT	IST	10:45	11:45	MD88b	MD88b	-	MD88b	-
OHY024	IST	AYT	16:45	17:45	MD83b	MD83b	-	MD83b	-
OHY025	AYT	IST	18:45	19:45	MD83b	MD83b	-	MD83b	-
OHY026	IST	AYT	20:30	21:30	MD88e	MD88e	-	MD88e	-
OHY027	AYT	IST	07:30	08:30	MD88e	MD88e	-	MD88e	-
OHY028	IST	BJV	11:55	12:55	MD88d	MD88d	-	MD88d	-
OHY029	BJV	IST	13:45	14:45	MD88d	MD88d	-	MD88d	-
OHY034	IST	DIY	06:45	08:30	MD88a	MD88c	S	MD88a	-
OHY035	DIY	IST	09:30	11:15	MD88a	MD88c	S	MD88a	-
OHY036	IST	DIY	18:45	20:30	MD83d	MD83d	-	MD83d	-
OHY037	DIY	IST	21:30	23:15	MD83d	MD83d	-	MD83d	-
OHY038	IST	ERZ	12:30	14:20	MD88c	MD88c	-	MD83c	S
OHY039	ERZ	IST	15:20	17:10	MD88c	MD88c	-	MD83c	S
OHY042	IST	GZT	09:30	11:15	MD83b	MD83b	-	MD83b	-
OHY043	GZT	IST	12:15	14:00	MD83b	MD83b	-	MD83b	-
OHY050	IST	ADB	07:45	08:45	MD83d	MD83d	-	MD83d	-
OHY051	ADB	IST	10:50	11:45	MD83d	MD83d	-	MD83d	-
OHY054	IST	ADB	16:30	17:30	MD88b	MD88b	-	MD88e	S
OHY055	ADB	IST	18:50	19:45	MD88b	MD88b	-	MD88e	S
OHY058	IST	ADB	20:45	21:45	MD83b	MD83b	-	MD83b	-
OHY059	ADB	IST	07:30	08:25	MD83b	MD83b	-	MD83b	-
OHY062	IST	MLX	12:15	13:45	MD83a	MD83a	-	MD83a	-
OHY063	MLX	IST	14:30	16:00	MD83a	MD83a	-	MD83a	-
OHY066	IST	KSY	07:15	08:30	MD83a	MD83a	-	MD83a	-
OHY067	KSY	IST	09:15	10:30	MD83a	MD83a	-	MD83a	-
OHY074	IST	SZF	09:30	10:45	MD88e	MD88e	-	MD88e	-
OHY075	SZF	IST	11:45	13:00	MD88e	MD88e	-	MD88e	-
OHY080	IST	TZX	06:50	08:25	MD83c	MD83c	-	MD83c	-
OHY081	TZX	IST	09:20	10:55	MD83c	MD83c	-	MD83c	-
OHY086	IST	TZX	18:45	20:20	MD88c	MD88c	-	MD88c	-
OHY087	TZX	IST	21:15	22:50	MD88c	MD88c	-	MD88c	-
* Recovery Strategies : S: swapping aircrafts, D: delaying flights, C: cancelling flights									

The solution of example problems can be seen in Table 4.4. As it can be seen from the Table 4.4, no flight leg has to be cancelled or delayed because of aircraft unavailability. Since the original flight schedule is not too tight, the remaining/available aircrafts can cover all the flight legs with aircraft swapping operations and without changing the original schedule too much.

- In the first example solution, the flight legs OHY034 and OHY035 are assigned to another available aircraft MD88c without changing any other aircrafts planned flights.

- In the second example, the flight legs OHY038 and OHY039 are assigned to another available aircraft MD83c.

4.5 Evaluation and Comparison of the Developed Aircraft Recovery Models

In this chapter various models developed for aircraft recovery problems have been presented.

Model A1 solves aircraft recovery problems using multi-commodity network flow models, including two recovery strategies, which are cancelling flights and swapping aircrafts between flight legs and considering single aircraft shortages.

Model A2 solves aircraft recovery problems using multi-commodity network flow models, including three recovery strategies, which are cancelling flights, delaying flights and swapping aircrafts between flight legs and considering multiple aircraft shortages.

Model A3 solves aircraft recovery problems using multi-commodity network flow models, including three recovery strategies, which are cancelling flights, delaying flights and swapping aircrafts between flight legs and considering multiple aircraft shortages and crew availability constraints.

Model A4 solves aircraft recovery problems using constraint programming embedded solution approach, including three recovery strategies, which are cancelling flights, delaying flights and swapping aircrafts between flight legs and considering multiple aircraft shortages and crew availability constraints.

In order to evaluate the above mentioned models' efficiency, several cases have been solved with each of the models and the solution times and the problem sizes have been recorded.

Table 4.5 The size of the aircraft recovery problems and solution times (Model A1 & A2 & A3)

Aircraft Recovery Solutions with Multi-Commodity Network Flow Models Aircraft Recovery Problem Models A1 & A2 & A3					
Aircraft Recovery Model	Recovery Period	# of flight legs in recovery period	#of variables	#of constraints	Solution time (in sec)
A1	16:45 - 19:45	4	2119	1255	0.10
A1	06:00 - 11:30	13	2119	2047	0.19
A1	08:00 - 12:00	9	2119	1712	0.20
A1	12:15 - 16:45	6	2119	1216	0.10
A1	12:00 - 18:00	9	2119	1216	0.10
A1	16:30 - 20:00	5	2119	1169	0.09
A2	07:10 - 17:30	21	18679	6068	1.03
A2	09:30 - 17:30	13	18679	5517	0.95
A2	09:30 - 19:00	10	18679	5420	0.94
A2	07:10 - 19:00	18	18679	5981	1.03
A2	06:00 - 20:00	17	18679	6064	1.48
A2	08:00 - 20:00	13	18679	5714	0.98
A2	06:00 - 12:00	15	18679	6496	1.03
A2	12:15 - 19:00	11	18679	5130	0.89
A2	12:15 - 17:30	8	18679	5245	0.89
A2	09:30 - 23:00	10	18679	5419	0.97
A3	07:10 - 17:30	21	18719	6552	1.06
A3	09:30 - 17:30	13	18719	6188	0.98
A3	09:30 - 19:00	10	18719	5888	0.98
A3	07:10 - 19:00	18	18719	6283	1.48
A3	06:00 - 20:00	17	18719	6059	1.00
A3	08:00 - 20:00	13	18719	6057	1.00
A3	06:00 - 12:00	15	18719	7686	1.05
A3	12:15 - 19:00	11	18719	5891	0.92
A3	12:15 - 17:30	8	18719	6363	0.92
A3	09:30 - 23:00	10	18719	5845	1.03

The solution statistics of the first three models (A1, A2 and A3), which uses multi-commodity network flow models, can be seen in Table 4.5. As it can be seen from the table the models developed with multi-commodity network flow models generates aircraft recovery solution in real time. Furthermore, the solution times of the example cases are almost 1 second, even if the number of variables reaches 18,719. However, this solution times do not include the network construction time, which is very time consuming and important part of the problem.

Table 4.6 The size of the aircraft recovery problems and solution times (Model A4)

Aircraft Recovery Solutions with Constraint Programming Embedded Solution Approach Aircraft Recovery Problem Model A4					
Recovery Period	# of flight legs in recovery period	# of generated routes	route generation time (constraint prog.) (in sec)	reschedule optimization time (integer prog.) (in sec)	total solution time (in sec)
15:00 - 19:00	5	181	1.7180	0.0780	1.7960
09:30 - 14:00	9	2241	1.8250	0.0930	1.9180
18:00 - 00:00	10	2745	1.7170	0.1090	1.8260
12:00 - 20:00	12	3465	1.8870	0.1250	2.0120
07:00 - 12:00	13	2619	1.8100	0.1100	1.9200
09:30 - 18:00	13	4050	2.1210	0.1710	2.2920
09:00 - 15:00	14	3375	2.0730	0.1410	2.2140
06:40 - 12:00	16	3096	1.9960	0.0950	2.0910
07:00 - 15:00	21	4059	2.2000	0.1710	2.3710
09:00 - 20:00	22	5139	2.1520	0.2340	2.3860

Table 4.6 summarizes the solution times of the aircraft recovery problem using constraint programming embedded approach (Model A4) for different recovery periods. The results show that the constraint programming technique generates real time solutions, less than 2.5 seconds. The results also show that the Model A4 efficiently generates large number of alternative feasible aircrafts routes in very short time.

The solution times listed in Table 4.6 include route generation times plus best route selection times. Although solution times of the overall procedure are very short, most of the time is dedicated to route generation. The route selection times in the constraint programming involved approach is even shorter, less than a 0.1 second.

When compared the solution times in Table 4.5 and Table 4.6, the solution times generated by the Models A1, A2 and A3, which are developed using multi-commodity network flow models, seem shorter than the solution times generated by the Model A4, constraint programming involved approach. However, this conclusion is deceptive. Because the solution times of the models A1, A2 and A3 do not involve the network structure creation times, but involves only selection of best routes satisfying given constraints. However, the solution times obtained from Model 4, the constraint programming involved approach, includes route generation time plus best route selection times and most of the time is dedicated to route generation. The route selection times in the constraint programming involved approach is less than a 0.1 second, much less than the solution times of the Models A1, A2 and A3.

The quality of the proposed models is also evaluated in terms of the number of impacted aircrafts. In the recovered solutions, it can be seen that, with Model A4, the number of impacted aircrafts is much less than the other models' solution. Model A4, which includes constraint programming technique is capable of reassigning aircrafts with little change in the initial schedule.

Constraint programming embedded approach is more efficient and stronger method compared to the traditional multi-commodity network flow models. The main reasons can be summarized as follows:

While solving aircraft recovery problems modeled as multi-commodity network flow problem, network structure of the problem needs to be prepared manually and given as input data to the problem. This process is very time consuming and it is hard to adjust in case of flight schedule changes. However, constraint programming approach generate feasible routes, which correspond to network structure in the multi-commodity network flow problem, with reference to the given constraints

without any manual intervention to the problem. The constraint programming generates large number of routes in very short time (see Table 4.6). The approach is very adaptable in case of schedule changes.

The other strength of the constraint programming involved approach is that constraint programming part of the problem finds the feasible routes satisfying all the constraints. Since the problem related constraints are satisfied in the constraint programming part of the problem, the integer programming problem only selects the best set of routes, which makes the integer programming model less complex than the traditional network flow models. Since the complexity of the integer programming model is reduced, obtaining real time solutions become easier. On the other hand, in the traditional multi-commodity network flow problems, all the complex constraints have to be modeled in the network flow model formulation. Because of the complex constraints, the problem does not always give the feasible solution. Therefore in order to obtain feasible solutions, special attention is required and constraint relaxation techniques should be used.

CHAPTER FIVE
CREW RECOVERY PROBLEMS

In the airline industry crews represent the airlines' second highest operating cost after fuel, so even slight improvements in their utilization can translate into significant savings. Because the large savings are possible while using aircrews more efficiently, many airlines have tried to develop optimization methods to solve them efficiently.

Although a lot of works have been published on planning crews (crew scheduling problems) in the literature, recovering crews (crew recovery problems) have not taken much attention. As explained in Chapter 2, in the disruptions management literature most of the work has focused on aircraft recovery problems and crew recovery problems have not been given a priority. The reasons why the crew recovery problems have not been studied so much are as follows:

1. Crew recovery problem is more complex due to the number of crew and more complex rules and regulations for crew. Because these complex rules the resulting crew related problem becomes more complicated.

2. Crews are less costly resources compared to aircrafts. Since the most expensive resource in the airline is the aircrafts, they are fixed first and foremost. Therefore, the crews being the second costly resources in the airlines are not taken much attention.

3. It is easier to find standby/spare crew in case of need. While it is difficult to find any spare aircraft in case of a shortage, crews can be repositioned much more easily or standby crews are often available.

In the literature crew recovery problems have been modeled using different methods such as network flow problems, mix-integer programming problem, set partitioning or set covering problem. In this dissertation, for modeling crew

recovery problems two different technique will be used resulting two different crew recovery models:

Model C1 is developed using multi-commodity network flow problem where the each commodity represents each crew. The underlying network is the connection network as in the aircraft recovery models A1, A2 and A3. In the connection network, the nodes in the network represent scheduled flight legs with corresponding departure/arrival times and origin/destination stations. The arcs in the network represent connections between two consecutive flight legs where the destination of the first flight leg is the same as the origin of the next flight leg.

Model C2 is developed using constraint programming embedded solution approach as in Model A4 of the aircraft recovery problem. This model both integer programming and constraint programming techniques.

5.1 Model C1

The Crew Recovery Problem as Multi-Commodity Network Flow Problem

Model C1 for crew recovery problem is modeled similar to the Model A2 developed for aircraft recovery problem. Thus, Model C1 is multi-commodity network flow problem where the underlying network is connection network. In this case, each commodity represents each crew and crew related rules and regulations are involved in the model as new constraints. In Model C1;

- Five different flight connection networks are used in the problem formulation, which represent
  - The flight connections before recovery period
  - The flight connections after recovery period
  - The flight connections during recovery period (on time)
  - The flight connections during recovery period with a delay of 30 minutes
  - The flight connections during recovery period with a delay of 60 minutes

- Following recovery strategies are considered:

- Reserve/stand by crews can be used to fly uncovered flight leg.
- Good (undisrupted) crews can be swapped to cover broken pairs.
- Flights can be cancelled.
- Flights can be delayed for 30 minutes or 60 minutes.

- Aircraft availabilities are not included in the model.

5.1.1 The Notation and Mathematical Formulation of Model C1

The notation used in the problem formulation is as follows:

K : set of crews

A : set of available crews in the recovery period

F : set of all flight legs

F^k : set of flight legs that can be flown by crew k

N^k : rescheduling network; set of flight connections for crew k during recovery period. (Feasible connection between two consecutive flight leg)

N_{30}^k : 30 minutes of delayed connection network; set of flight connections for aircraft k during recovery period.

N_{60}^k : 60 minutes of delayed connection network; set of flight connections for aircraft k during recovery period.

L^k : set of flight connections for crew k before the recovery period begins

M^k : set of flight connections for crew k after the recovery period finishes

k : index representing the crews

i : index representing the first flight in the connection arc

j : index representing the next flight in the connection arc

arr_i : arrival time of flight i

$dept_j$: departure time of flight j

org_j : origin station of flight j

$dest_i$: destination station of flight i

$sittime$: minimum required sit time time between two consecutive flight legs

$flytime$: maximum allowed flying time for a crew

$dutytime$: maximum allowed duty time for a crew

$lower$: minimum number of flight legs assigned to crews

$upper$: maximum number of flight legs assigned to crews

$pairing_{ij}^{k}$: the original crew – flight assignment. It is equal to 1, if the crew k is assigned to two consecutive flights i and j in the original schedule, 0 otherwise.

c_{ij}^{k} : the cost of reassigning crew k between flights i and j

r_j : the cost of using reserve/standby crew for flight j.

b_j : the cost of cancelling flight j.

X_{ij}^{k} : the binary decision variable which takes value 1 if the crew k fly two consecutive flights i and j
and 0 otherwise.

Y_j : the binary decision variable which takes value 1 if the flight j is cancelled and 0 otherwise.

Z_j : the binary decision variable represents the usage of reserve/standby crew. It takes value 1 if the flight j is assigned to reserve crew and 0 otherwise.

The mathematical formulation of the problem is as follows:

$$Minimize \sum_{k\in K}\sum_{(i,j)\in N^k\cup M^k\cup L^k} c_{ij}^{k}\, X_{ij}^{k} + \sum_{k\in K}\sum_{(i,j)\in N_{30}^{k}\cup N_{60}^{k}} d_{ij}^{k}\, X_{ij}^{k} + \sum_{i\epsilon F} b_i\, Y_i + \sum_{i\epsilon F} r_i\, Z_i \tag{5.1}$$

Subject to:

$$\sum_{k\in K}\sum_{j:(i,j)\in N^k\cup M^k\cup L^k} X_{ij}^{k} + \sum_{k\in K}\sum_{(i,j)\in N_{30}^{k}\cup N_{60}^{k}} X_{ij}^{k} + Y_i + Z_i = 1 \qquad \forall i \in F^k \tag{5.2}$$

$$X_{ij}^{k} = 0 \qquad \forall k \in K - \{R\}\ \forall (i,j) \in N^{k} \cup N_{30}^{k} \cup N_{60}^{k} \qquad (5.3)$$

$$X_{ij}^{k} = sch_{ij}^{k} \qquad \forall k \in K\ \ \forall (i,j) \in L^{k} \qquad (5.4)$$

$$X_{ij}^{k} = sch_{ij}^{k} \qquad \forall k \in K\ \ \forall (i,j) \in M^{k} \qquad (5.5)$$

$$X_{ij}^{k}\left(arr_{i} + sittime - dept_{j}\right) \geq 0 \qquad \forall k \in R\ \ \forall (i,j) \in N^{k} \cup N_{30}^{k} \cup N_{60}^{k} \qquad (5.6)$$

$$X_{ij}^{k}\left(dest_{i} - org_{j}\right) = 0 \qquad \forall k \in R\ \ \forall (i,j) \in N^{k} \cup N_{30}^{k} \cup N_{60}^{k} \qquad (5.7)$$

$$\sum_{j:(i,j)\in N^{k}\cup N_{30}^{k}\cup N_{60}^{k}} X_{ij}^{k} - \sum_{j:(j,i)\in N^{k}\cup N_{30}^{k}\cup N_{60}^{k}} X_{ji}^{k} = 0 \qquad \forall k \in R\ \ \forall i \in F^{k} \qquad (5.8)$$

$$\sum_{(i,j)\in N^{k}\cup M^{k}\cup L^{k}\cup N_{30}^{k}\cup N_{60}^{k}} X_{ij}^{k} \leq lower \qquad \forall k \in K \qquad (5.9)$$

$$\sum_{(i,j)\in N^{k}\cup M^{k}\cup L^{k}\cup N_{30}^{k}\cup N_{60}^{k}} X_{ij}^{k} \geq upper \qquad \forall k \in K \qquad (5.10)$$

$$\sum_{i:(i,j)\in N^{k}\cup M^{k}\cup L^{k}\cup N_{30}^{k}\cup N_{60}^{k}} X_{ij}^{k}\,(arr_{i} - dept_{i}) \leq flytime \qquad \forall k \in K \qquad (5.11)$$

$$\sum_{i:(i,j)\in N^{k}\cup M^{k}\cup L^{k}\cup N_{30}^{k}\cup N_{60}^{k}} X_{ij}^{k}\,(dept_{j} - dept_{i}) \leq dutytime \qquad \forall k \in K \qquad (5.12)$$

$$\sum_{i\in F} Z_{i}\,(arr_{i} - dept_{i}) \leq flytime \qquad (5.13)$$

$$\sum_{i:(i,j)\in N^{k}} Z_{i}\,(dept_{j} - dept_{i}) \leq dutytime \qquad (5.14)$$

$$X_{ij}^{k} \in \{0,1\} \qquad \forall k \in K\ \forall (i,j) \in N^{k} \cup M^{k} \cup L^{k} \cup N_{30}^{k} \cup N_{60}^{k} \qquad (5.15)$$

$$Y_{i} \in \{0,1\} \qquad \forall i \in F \qquad (5.16)$$

$Z_i \in \{0,1\}$ $\forall i \in F$ (5.17)

where the objective function (5.1) minimizes the total cost of reassigning crews to the flight legs, using reserve crew, delaying flight legs and cancelling flights. The constraint (5.2) is the flight cover constraint and ensures that all flight legs will be either covered by a good crew (delayed or on time) or by a reserve crew or cancelled. The constraint (5.3) ensures that during recovery period no flights will be assigned to the shortage crew. The constraints (5.4) and (5.5) ensure that before and after the recovery period the crew assignment will be same as the original schedule. In other words, it will be returned to the original crew flight assignment after the shortage crew becomes available again. The constraint (5.6) guarantees the time requirements between consecutive flight legs: there should be at least the minimum required sit time between the arrival of the flight *i* and the departure of the flight *j* for the available crews during recovery period. The constraint (5.7) ensures that the destination station of the first flight and the origin of the second flight are the same for the available crews during recovery period. The constraint (5.8) ensures that the total arc flow, which enters the node *i* must be equal to the total arc flow which leaves the node *i* for the available crews during recovery period. Constraints (5.9) and (5.10) limit the number of flight legs to be flown by a crew. Constraint (5.11) limits the total flying time for a crew. A crew can not fly more than predefined total flying time in one pairing. Constraint (5.12) limits the duty time for a crew. Total time spent on duty by a crew can not be more than predefined total duty time. Constraint (5.13) limits the total flying time for a reserve/standby crew. Constraint (5.14) limits the total duty time for a reserve/standby crew. Finally, constraints (5.15) - (5.17) ensure that all decision variables are binary.

5.1.2 Numerical Example for Model C1

The developed model C1 is solved using OPL Studio 3.7. The OPL model of the problem can be seen in Appendix G.

In order to solve the crew recovery problem, the initial crew schedule should be known in advance and should be used as input to the crew recovery problem. In

order to develop the initial crew schedule, the crew scheduling problem which was modeled as multi-commodity network flow problem, has been solved and the initial schedule seen in Appendix A in Table A.2 has been obtained.

Model C1 for crew recovery problem is solved using the real data obtained from OnurAir domestic flight schedule, as in the aircraft recovery models presented in Appendix A in Table A.2. Therefore the flight schedule seen in Table A.2 is used also in this Model C1.

Several cases for different recovery period horizons chosen randomly have been tested in order to evaluate the efficiency of the Model C1. However, only two of the solutions are presented as example cases.

In the first example, there occurs a shortage on crew “C03” at 07:10 am causing that the crew “C03” can not fly the flight legs OHY066 that is supposed to leave station IST at 07:15 and arrive at station KSY at 08:30. The shortage continues until 12:00. During this period the crew “C03” can not fly the flight OHY067 either.

In the second example, there occurs a shortage on crew “C02” at 08:00 am causing that the crew “C02” con not fly the flight legs OHY022 that is supposed to leave station IST at 08:15 and arrive at station AYT at 09:15. The shortage continues until 12:00. During this period the crew “C02” can not fly the flight OHY023 either.

The solution obtained from above example cases can be seen in Table 5.1.

In the first example, no flight leg has to be cancelled because of crew unavailability. The flight legs OHY066 and OHY067 are assigned to another crew “C09”, instead of shortage crew “C03”. During the recovery period between 07:10 and 12:00 some crews are swapped among the flight legs and OHY074 and OHY067 are assigned to another crew “C11” instead of their original crew "C09". Finally, the flight legs OHY034 and OHY035 are assigned to the reserve/standby crew, because its original crew are assigned to other flight legs as a result of crew swapping and

there is no available crew to able to fly these two flights without violating crew related rules and regulations.

Table 5.1 Recovered crew schedule obtained from the solution of Model C1

Original Schedule						Shortage occurs on Crew 3 at 07:10		Shortage occurs on Crew 2 at 08:00	
Flight No	Origin	Destination	Departure	Arrival	Crew	Crew	Recovery strategy	Crew	Recovery strategy
OHY010	IST	ADA	06:45	08:15	C10	C10	-	C10	-
OHY011	ADA	IST	09:10	10:40	C10	C10	-	C10	-
OHY016	IST	ADA	15:00	16:30	C04	C04	-	C04	-
OHY017	ADA	IST	17:30	19:00	C04	C04	-	C04	-
OHY018	IST	ADA	19:00	20:30	C02	C02	-	C02	-
OHY019	ADA	IST	21:30	23:00	C02	C02	-	C02	-
OHY022	IST	AYT	08:15	09:15	C02	C02	-	C04	S
OHY023	AYT	IST	10:45	11:45	C02	C02	-	C04	S
OHY024	IST	AYT	16:45	17:45	C09	C09	-	C09	-
OHY025	AYT	IST	18:45	19:45	C09	C09	-	C09	-
OHY026	IST	AYT	20:30	21:30	C01	C01	-	C01	-
OHY027	AYT	IST	07:30	08:30	C01	C01	-	C01	-
OHY028	IST	BJV	11:55	12:55	C10	C10	-	C10	-
OHY029	BJV	IST	13:45	14:45	C10	C10	-	C10	-
OHY034	IST	DIY	06:45	08:30	C11	Reserve	R	Reserve	R
OHY035	DIY	IST	09:30	11:15	C11	Reserve	R	Reserve	R
OHY036	IST	DIY	18:45	20:30	C06	C06	-	C06	-
OHY037	DIY	IST	21:30	23:15	C06	C06	-	C06	-
OHY038	IST	ERZ	12:30	14:20	C08	C08	-	C08	-
OHY039	ERZ	IST	15:20	17:10	C08	C08	-	C08	-
OHY042	IST	GZT	09:30	11:15	C05	C05	-	C05	-
OHY043	GZT	IST	12:15	14:00	C05	C05	-	C05	-
OHY050	IST	ADB	07:45	08:45	C04	C04	-	C09	S
OHY051	ADB	IST	10:50	11:45	C04	C04	-	C09	S
OHY054	IST	ADB	16:30	17:30	C03	C03	-	C03	-
OHY055	ADB	IST	18:50	19:45	C03	C03	-	C03	-
OHY058	IST	ADB	20:45	21:45	C11	C11	-	C11	-
OHY059	ADB	IST	07:30	08:25	C06	C11	S	C11	S
OHY062	IST	MLX	12:15	13:45	C01	C01	-	C01	-
OHY063	MLX	IST	14:30	16:00	C01	C01	-	C01	-
OHY066	IST	KSY	07:15	08:30	C03	C09	S	C03	-
OHY067	KSY	IST	09:15	10:30	C03	C09	S	C03	-
OHY074	IST	SZF	09:30	10:45	C09	C11	S	C11	S
OHY075	SZF	IST	11:45	13:00	C09	C11	S	C11	S
OHY080	IST	TZX	06:50	08:25	C07	C07	-	C07	-
OHY081	TZX	IST	09:20	10:55	C07	C07	-	C07	-
OHY086	IST	TZX	18:45	20:20	C12	C12	-	C12	-
OHY087	TZX	IST	21:15	22:50	C12	C12	-	C12	-

* Recovery Strategies :
S: swapping crews, R: using reserve/standby crew D: delaying flights, C: cancelling flights

In the second example, no flight leg has to be cancelled either. The flight legs OHY022 and OHY023 have been assigned to another crew “C04”, instead of shortage crew “C02”. During the recovery period between 08:00 and 12:00 some other crews have been swapped among the flight legs. Finally, the flight legs

OHY034 and OHY035 are assigned to the reserve/standby crew, because its original crew are assigned to other flight legs as a result of crew swapping and there is no available crew to able to fly these two flights without violating crew related rules and regulations.

5.2 Model C2

Crew Recovery Problem with Constraint Programming Embedded Solution Approach

Model C2 is developed with using the constraint programming technique instead as in Model A4 of the aircraft recovery problems. The solution procedure of C2 is similar to the solution procedure of Model A4, presented in Chapter 4, in Section 4.4, on pages 118-121. However, Model C2 is more complicated than the Model A4 in terms of the number of constraints. Because the crew related complex rules and regulations lead more complicated problem constraints. For example, there are duty time and flight time restrictions for crews, which have to be satisfied. Such constraints cause more complex pairing generation model. Flexible modeling framework of constraint programming make easier to model crew related complex constraints and to find feasible and effective crew reschedules in real time.

In Model C2;

- Following recovery strategies are considered:
  - Reserve/stand by crews can be used to fly uncovered flight leg.
  - Good (undisrupted) crews can be swapped to cover broken pairs.
  - Flights can be cancelled.
  - Flights can be delayed until 90 minutes.

- Aircraft availabilities are not included in the model.

Model C2 includes both integer programming and constraint programming techniques. The overall solution algorithm is divided into two main problems.

- First one is the constraint programming problem which finds all the feasible pairings for each crew covering each flight leg to be flown during recovery period while satisfying all crew related rules and regulations.

- The second one is the integer programming problem, which selects the best set of pairings minimizing the recovery costs and the changes in the original schedule.

The solution procedure, as seen in Figure 5.1, is as follows:

1. Crew scheduling problem is solved using constraint programming in order to provide initial conditions of the problem.

2. The initial crew schedule is defined as input data for crew recovery problem.

3. The first-stage problem constraint programming model is developed for generating pairings for each crew including each flight leg scheduled to be flown during recovery period. (Details are in Section 5.2.1 First-stage problem- Constraint programming for Model C2)

4. Newly generated crew pairings are sent to second-stage integer programming problem, which selects the optimum set of crew pairings while satisfying recovery period start/ end conditions, minimizing the recovery costs and the changes in the original schedule. (Details are in Section 5.2.2 Second-stage problem- Integer programming for Model C2)

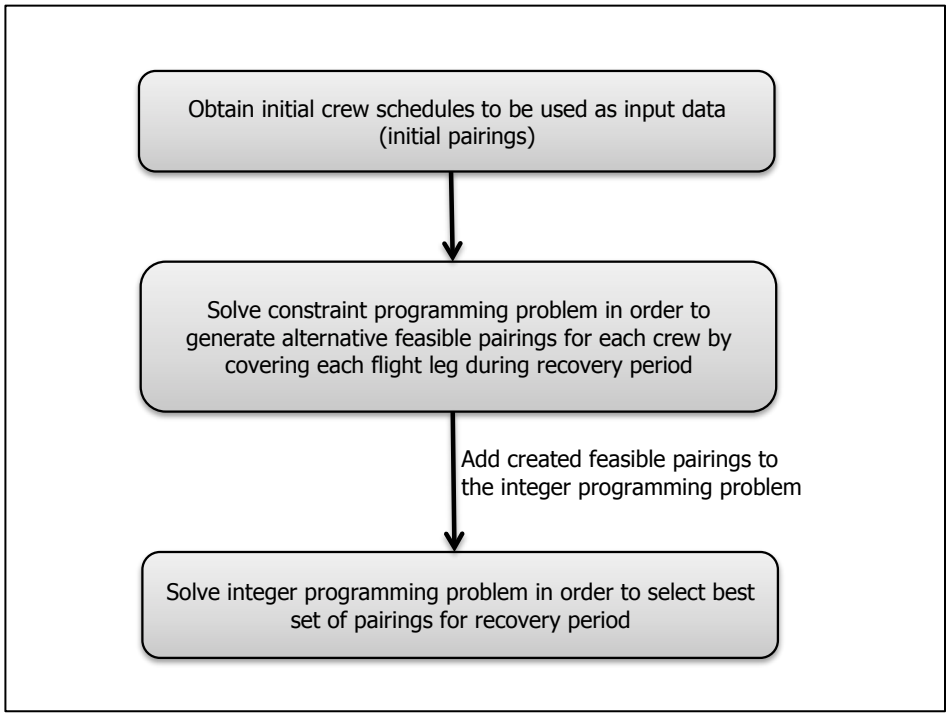

Figure 5.1 Solution Procedure Steps for Model C2

As in Model A4 presented in Chapter 4, in order to provide interaction between two stages (constraint programming part of the problem and inter programming part of the problem), a script code is written using OPL Script tool. The script code of the problem can be seen in Appendix H1.

The written script code first creates input data of initial crew schedule then it calls the constraint programming model sequentially for each flight leg in the recovery period. This model generates feasible crew pairings for each crew and for every selected flight legs (flight legs in the recovery period). When the crew pairings are generated for each crew and for all selected flight legs, the script calls the second-stage problem-integer programming model. This model selects the optimum set of crew pairings while minimizing recovery costs and the changes in original schedule.

5.2.1 First Stage - Constraint Programming Problem for Model C2

As mentioned earlier, this constraint programming model generates feasible crew pairings sequentially for each crew and for every selected flight legs (flight legs in

the recovery period). For each crew the pairings including 4 flight legs and 2 flight legs will be developed.

The constraint programming model is called for each flight leg in the recovery period. The variable "cover" representing flight legs during recovery period is imported from OPL Script to the constraint programming model for each flight leg in the recovery period.

The notation used in the problem is as follows:

F : set of flight legs

P : set of flight legs, which constitutes a legal pairing

S : set of crews

i : the index representing the flight leg position in a route

c : the index representing the crews

f : the index representing the flight legs

$cover$: the variable represents the flight legs during recovery period (transferred from OPL script to the constraint programming problem)

$dest_f$: the destination station of flight leg f

org_f : the origin station of flight leg f

arr_f : the arrival time of flight leg f

$dept_f$: the departure time of flight leg f

$AcArr_f$: the actual arrival time of flight leg f

$AcDept_f$: the actual departure time of flight leg f

$Delay_f$: the delay time of flight leg f

$MinStop$: the minimum required sit time between two consecutive flight legs.

$CitySeq_{c,i}$: the origin station of sequence i in the route of crew c .

$Seq_{c,i}$: the flight leg of sequence i in the route of crew c .

$MaxDuty$: the maximum duty time allowed for crews.

$MaxFlight$: the maximum flight time allowed for crews

The mathematical formulation of the model is as follows:

$$CitySeq_{c,i-1} = dest_{seq_{c,i}} \qquad \forall c \in S, \ \forall i \in P \qquad (5.18)$$

$$CitySeq_{c,i} = org_{seq_{c,i}} \qquad \forall c \in S, \ \forall i \in P \qquad (5.19)$$

$$CitySeq_{c,i-1} = CitySeq_{c,i} \qquad \forall c \in S, \ \forall i \in P \qquad (5.20)$$

$$AcArr_{seq_{c,i}} + MinStop \leq AcDept_{seq_{c,i+1}} \qquad \forall c \in S, \ \forall i \in R \qquad (5.21)$$

$$Delay_{seq_{c,i}} = AcDept_{seq_{c,i+1}} - dept_{seq_{c,i+1}} \qquad \forall c \in S, \ \forall i \in R \qquad (5.22)$$

$$Delay_{seq_{c,i}} \leq 90 \qquad \forall c \in S, \ \forall i \in R \qquad (5.23)$$

$$CitySeq_{c,0} = CitySeq_{c,n} \qquad \forall c \in S, \ n = 2 \ or \ n = 4 \qquad (5.24)$$

$$\left(\sum_{i \in P} seq_{c,i} = f \right) \leq 1 \qquad \forall c \in S, \ \forall f \in F \qquad (5.25)$$

$$\left(\sum_{i \in P} seq_{c,i} = cover \right) = 1 \qquad \forall c \in S \qquad (5.26)$$

$$\sum_{i \in P} \left(arr_{seq_{c,i}} - dept_{seq_{c,i}} \right) \leq MaxFlight \qquad \forall c \in S \qquad (5.27)$$

$$arr_{seq_{c,n}} - dept_{seq_{c,0}} \leq MaxDuty \qquad \forall c \in S, \ n = 2 \ or \ n = 4 \qquad (5.28)$$

Constraints (5.18 - 5.20) the destination station of the preceding flight leg will be same as the origin station of the subsequent flight leg in the flight sequence. Constraint (5.21) satisfies that there should be at least minimum sit time between two consecutive flight legs. Constraint (5.22) computes the delay time for each flight leg. Constraint (5.23) limits the total delay time for each flight leg. Constraint (5.24) satisfies that the crew pairing should begin and end at the same station. Constraint (5.25) satisfies that same flight leg will not be included in a pairing. Constraint (5.26) ensures that one particular flight (one of the flight legs scheduled to be flown in the recovery horizon) is included in the pairing for each crew. Constraint (5.27) limits the total flight time in a pairing. Constraint (5.28) limits the total duty time in a pairing.

5.2.2 Second Stage - Integer Programming Problem for Model C2

This integer program selects the best set of crew pairings for the recovery period, satisfying certain constraints.

After the completion of crew pairings generation during recovery period, OPL script calls the integer programming model. This model selects the optimum set of pairings during the recovery period while:

- minimizing the changes in original schedule (minimizing the deviation between original schedule and recovered schedule)
- minimizing the total cancellation costs
- minimizing the total delay costs
- minimizing the total crew reassignment costs
- satisfying that before and after recovery periods crews will return to their original schedule

The notation used in this problem is as follows:

S : set of crews

A : set of available crews in the recovery period

F : set of all flight legs

F_r^c : set of flight legs assigned to crew c in the recovery period

c : index representing the crews

i : index representing the flight legs

P_{ci} : the original crew– flight assignment. It is equal to 1, if the crew c is assigned to flights i in the original schedule, 0 otherwise.

C_{ci} : the cost of reassigning crew c to flight i.

D_i : the cost of cancelling flight i.

DC_i : the cost of delaying flight i.

E_i : the cost of assigning flight i to reserve crew.

W : the coefficient for minimizing the deviation between original schedule and recovered schedule.

X_{ci} : the binary decision variable, which takes value 1 if the crew c is assigned to flight i and 0 otherwise.

Y_i : the binary decision variable, which takes value 1 if the flight i is cancelled and 0 otherwise.

Z_i : the binary decision variable, which takes value 1 if reserve crew and 0 fly the flight i otherwise.

$recStart$: Recovery period start time

$recFinish$: Recovery period finish time

The mathematical formulation of the model is as follows:

$$Minimize \quad \sum_{c\in S}\sum_{i\in F_r^c} C_{ci}X_{ci} + \sum_{c\in S}\sum_{i\in F} W[X_{ci} - P_{ci}] + \sum_{i\in F} D_iY_i + \sum_{i\in F} E_iZ_i + \sum_{c\in S}\sum_{i\in F_r^c} DC_iX_{ci} \tag{5.29}$$

Subject to:

$$\sum_{c \in A} X_{ci} + Y_i + Z_i = 1 \qquad \forall i \in F \tag{5.30}$$

$$\sum_{i \in F_r^c} X_{ci} = 1 \qquad \forall c \in S \tag{5.31}$$

$$X_{ci} = P_{ki} \qquad \forall c \in S - \{A\} \qquad \forall i \in F : dept_i \leq recStart \,\&\, arr_i \geq recFinish \tag{5.32}$$

$$X_{ci} = 0 \qquad \forall c \in S \qquad \forall i \in F : dept_i \geq recStart \,\&\, arr_i \leq recFinish \tag{5.33}$$

$$X_{ci} \in \{0,1\} \qquad \forall c \in S \qquad \forall i \in F \tag{5.34}$$

$$Y_i \in \{0,1\} \qquad \forall i \in F \tag{5.35}$$

$$Z_i \in \{0,1\} \qquad \forall i \in F \tag{5.36}$$

The objective function (5.29) minimizes the total cost of reassigning crews to flight legs, cancelling flights, delaying flights and assigning flights to reserve crew and deviation between original schedule and recovered schedule. The constraint (5.30) ensures that all flight legs will be either covered by a good crew, reserve crew or cancelled. The constraint (5.31) ensures that all available crews, which are assigned to the flights in the initial crew schedule, are assigned to the flights during recovery period. The constraint (5.32) ensures that before and after the recovery period the crew-flight assignment will be same as the original schedule. Constraint (5.33) ensures that no flight will be assigned to shortage crew during recovery period. Constraints (5.34 - 5.36) ensure that all the decision variables are binary.

5.2.3 Numerical Example for Model C2

The developed model is solved using OPL Studio 3.7. The OPL script of the problem can be seen in Appendix H1. The OPL model of the first-stage constraint programming model can be seen in Appendix H2. The OPL model of the second-stage integer programming model can be seen in Appendix H3. The problem is solved using the domestic flight schedule of OnurAir as seen in Appendix A in Table A.3.

Several cases for different recovery period horizons chosen randomly have been tested in order to evaluate the efficiency of the approach. However, only two of the solutions are presented as example cases.

The first example considers the recovery period between 06:40 and 12:00 and the shortage on Crew 11. Thus Crew 11 can not fly flights OHY034 (which is supposed to leave station IST at 6:45 and arrive at station DIY at 08:30) and OHY035 (which is supposed to leave station DIY at 9:30 and arrive at station IST at 11:15).

The second example considers the recovery period between 12:00 and 20:00 and the shortage on Crew 08. Thus Crew 08 can not fly flights OHY038 (which is supposed to leave station IST at 12:30 and arrive at station ERZ at 14:20) and OHY039 (which is supposed to leave station ERZ at 15:20 and arrive at station IST at 17:10).

The solutions of the example problems can be seen in Table 5.2.

- In example problem 1, flights OHY034 and OHY035 are assigned to reserve/standby crew without changing any other crew's planned flights.

- In example problem 2, flights OHY038 and OHY039 are assigned to reserve/standby crew without changing any other crew's planned flights.

Reserve/standby crew has to be used, because of the duty and flight time restrictions of crew has to be respected.

Table 5.2 Recovered crew schedule obtained from the solution of Model C2

Original Schedule						Shortage occurs on Crew 11 at 06:40		Shortage occurs on Crew 8 at 12:00	
Flight No	Origin	Destination	Departure	Arrival	Crew	Crew	Recovery strategy	Crew	Recovery strategy
OHY010	IST	ADA	6:45	8:15	C12	C12	-	C12	-
OHY011	ADA	IST	9:10	10:40	C12	C12	-	C12	-
OHY016	IST	ADA	15:00	16:30	C03	C03	-	C03	-
OHY017	ADA	IST	17:30	19:00	C03	C03	-	C03	-
OHY018	IST	ADA	19:00	20:30	C04	C04	-	C04	-
OHY019	ADA	IST	21:30	23:00	C04	C04	-	C04	-
OHY022	IST	AYT	8:15	9:15	C02	C02	-	C02	-
OHY023	AYT	IST	10:45	11:45	C02	C02	-	C02	-
OHY024	IST	AYT	16:45	17:45	C07	C07	-	C07	-
OHY025	AYT	IST	18:45	19:45	C06	C06	-	C06	-
OHY026	IST	AYT	20:30	21:30	C06	C06	-	C06	-
OHY027	AYT	IST	7:30	8:30	C07	C07	-	C07	-
OHY028	IST	BJV	11:55	12:55	C05	C05	-	C05	-
OHY029	BJV	IST	13:45	14:45	C05	C05	-	C05	-
OHY034	IST	DIY	6:45	8:30	C11	Reserve	R	C011	-
OHY035	DIY	IST	9:30	11:15	C11	Reserve	R	C011	-
OHY036	IST	DIY	18:45	20:30	C02	C02	-	C02	-
OHY037	DIY	IST	21:30	23:15	C02	C02	-	C02	-
OHY038	IST	ERZ	12:30	14:20	C08	C08	-	Reserve	R
OHY039	ERZ	IST	15:20	17:10	C08	C08	-	Reserve	R
OHY042	IST	GZT	9:30	11:15	C07	C07	-	C07	-
OHY043	GZT	IST	12:15	14:00	C07	C07	-	C07	-
OHY050	IST	ADB	7:45	8:45	C08	C08	-	C08	-
OHY051	ADB	IST	10:50	11:45	C08	C08	-	C08	-
OHY054	IST	ADB	16:30	17:30	C01	C01	-	C01	-
OHY055	ADB	IST	18:50	19:45	C09	C09	-	C09	-
OHY058	IST	ADB	20:45	21:45	C09	C09	-	C09	-
OHY059	ADB	IST	7:30	8:25	C01	C01	-	C01	-
OHY062	IST	MLX	12:15	13:45	C12	C12	-	C12	-
OHY063	MLX	IST	14:30	16:00	C12	C12	-	C12	-
OHY066	IST	KSY	7:15	8:30	C03	C03	-	C03	-
OHY067	KSY	IST	9:15	10:30	C03	C03	-	C03	-
OHY074	IST	SZF	9:30	10:45	C01	C01	-	C01	-
OHY075	SZF	IST	11:45	13:00	C01	C01	-	C01	-
OHY080	IST	TZX	6:50	8:25	C05	C05	-	C05	-
OHY081	TZX	IST	9:20	10:55	C05	C05	-	C05	-
OHY086	IST	TZX	18:45	20:20	C10	C10	-	C10	-
OHY087	TZX	IST	21:15	22:50	C10	C10	-	C10	-

* Recovery Strategies :
S: swapping crews, R: using reserve/standby crew D: delaying flights, C: cancelling flights

5.3 Evaluation and Comparison of Developed Crew Recovery Models

In this chapter two models developed for crew recovery problems have been presented.

Model C1 solves crew recovery problem using multi-commodity network flow models, including four recovery strategies, which are cancelling flights, delaying flights, using reserve/standby crews and swapping crews between flight legs.

Model C2 solves crew recovery problem using constraint programming embedded solution approach, including four recovery strategies, which are cancelling flights, delaying flights, using reserve/standby crews and swapping crews between flight legs.

In order to evaluate these two models' efficiency, several cases have been solved with each of the models and the solution times and the problem sizes have been recorded.

Table 5.3 The size of the crew recovery problem and solution times (Model C1)

Crew Recovery Solutions with Multi-Commodity Network Flow Models Crew Recovery Problem Model C1				
Recovery Period	# of flight legs in recovery period	#of variables	#of constraints	Solution time (in sec)
07:00 - 10:30	6	2888	3661	1.04
06:00 - 12:00	15	2888	3395	1.35
19:00 - 23:00	5	2888	3528	0.70
06:50 - 11:00	10	2888	3437	0.78
18:45 - 00:00	10	2888	3607	0.84
06:45 - 11:15	13	2888	3462	0.75
06:00 - 10:40	9	2888	3613	0.85
15:00 - 19:00	5	2888	3325	0.82
12:30 - 17:30	6	2888	3169	0.72
16:00 - 00:00	13	2888	3241	0.92

The solution statistics of the Model C1, which uses multi-commodity network flow models, can be seen in Table 5.3. As it can be seen from the table the models developed with multi-commodity network flow models generates crew recovery solution in real time, in less than 1 second. However, this solution times do not include the network construction time, which is very time consuming and important part of the problem.

The solution statistics of the Model C2, which uses constraint programming embedded approach, can be seen in Table 5.4. The results show that the crew recovery problem solved with constraint programming embedded solution approach generates the real time solutions, in about 2.5 seconds. The results also show that the constraint programming technique efficiently generates large number of alternative feasible crew pairings in very short time.

The solution times listed in Table 5.4 include crew pairings generation times plus best set of pairing selection times. Most of the time is dedicated to pairing generation. Therefore, the best set of pairing selection times in the constraint programming involved approach is even shorter, less than a 0.1 second.

Table 5.4 The size of the crew recovery problems and solution times (Model C2)

Crew Recovery Solutions with Constraint Programming Embedded Solution Approach Crew Recovery Problem Model C2					
Recovery Period	# of flight legs in recovery period	# of generated pairings	pairings generation time (constraint prog.) (in sec)	reschedule optimization time (integer prog.) (in sec)	total solution time (in sec)
18:00 - 00:00	10	1860	2.045	0.078	2.123
06:40 - 11:40	13	1920	2.327	0.078	2.405
12:00 - 20:00	12	2544	2.168	0.109	2.277
18:50 - 00:00	7	1620	2.059	0.063	2.122
06:00 - 12:00	15	2160	2.218	0.078	2.296
09:30 - 15:00	11	2172	2.372	0.094	2.466
08:00 - 12:00	9	1608	2.060	0.063	2.123
15:00 - 19:00	5	1680	2.011	0.063	2.074
11:30 - 15:00	6	1656	2.293	0.078	2.371
07:15 - 11:30	11	1812	2.295	0.078	2.373

The efficiency of the proposed models are evaluated also in terms of number of impacted crews. The number of impacted crews in the recovery solutions obtained from Model C1 and Model C2 can be seen in Tables 5.1 and 5.2, respectively. In the solution of Model C1, the number of swapping options is much more than which is obtained from the solution of Model C2. Model C2, which involves the constraint programming technique, is capable of finding a recovered schedule without affecting all the crew members worked in the recovery period.

As mentioned earlier in Section 4.5 of Chapter 4, on pages 128-131 , the multi-commodity network flow solutions do not include the connection network construction times, which is very time consuming and the solution times seen in Table 5.3 involves only finding optimum reschedule. The connection networks of the crew recovery problem have been developed manually.

However in the constraint programming embedded model C2, no manual computation is required. Because the constraint programming part of the problem generate feasible pairings, which correspond to network structure in the multi-commodity network flow problem, with reference to the given constraints without any manual intervention to the problem. The solution times recorded in Table 5.4 include both feasible pairings generation times and obtaining optimum schedule times.

Constraint programming involved approach is also better than multi-commodity network flow models in terms modeling approach. The constraint programming part of the problem finds the feasible pairings satisfying all crew related complex constraints. Since the problem related constraints are satisfied in the constraint programming part of the problem, the integer programming problem only selects the best set of pairings, which makes the integer programming model less complex than the traditional network flow models. Since the complexity of the integer programming model is reduced, obtaining real time solutions become easier.

CHAPTER SIX
ITERATIVE SOLUTION APPROACH FOR INTEGRATED AIRCAFT AND CREW RECOVERY PROBLEMS USING MULTI-COMMODITY NETWORK FLOW MODELS

The integrated crew and aircraft recovery problems are complex in terms of modeling. Because, the integrated problem includes both crew and aircraft related constraints and variables. Thus, the number of constraints and variables become very huge compared to the individual recovery problems. Because of the increased complexity and number of variables, the integrated problem is NP-hard and it is much harder to be solved. They require special attention and complex techniques in order to be solved to optimality in real time.

Because of the above mentioned challenges of the integrated recovery problems, only a few studies have been performed on the integrated aircraft and crew recovery problems in the disruption management literature. As explained in Chapter 2, although the proposed models of the integrated problems seem efficient, they have not been fully tested with real data. Therefore this area of the airline recovery problems is still subject to further studies and improvements.

On the other hand, the aircraft and crew recovery problems have mostly been modeled and solved individually in the disruptions management literature. The general assumption in the aircraft recovery problem literature is that the crew members are available at any time. Most of the studies do not include the crew availability constraints in the aircraft recovery problems. As seen in Table 2.1, only a few studies involve the crew related considerations in the problems. However, these problems, which involve crew considerations, do not attempt to recover disrupted crew. Similarly crew recovery problems do not consider the aircraft availability. They only focus on recovering disrupted crew without considering other airline resources. So far, in this dissertation the same consideration has been taken into account. The crew and aircraft recovery problems developed in Chapters 4 and 5 of this dissertation have been considered individually.

While solving airline resource scheduling problems, the crew and aircraft related decisions affect each other. In order to operate any flight leg there must be at least one available crew and one available aircraft to be assigned to the flight leg. The same consideration exists also for recovery problems. In case of schedule disruption even if there is an available aircraft for the flight leg, subject flight leg can not be operated until an available crew is found or wise versa.

When the aircraft and crew recovery problems are solved separately, some feasibility problems occur due to lack of congruency of the solutions. For example, the solution obtained from the crew recovery problem indicates that one of the flight legs should be cancelled. On the other hand, the solution obtained from aircraft recovery problem indicates that an available aircraft covers the subject flight leg. In this case, there is a flight leg that is covered by an aircraft but not covered by a crew. That causes incongruence between the solutions of two problems; aircraft recovery problem should take into account the crew availabilities and the crew recovery problem should consider the aircraft availabilities. If one of the resources is not available for a flight leg, the subject flight leg should be cancelled or delayed until available aircraft and/or crew is found. By integrating the two problems subject incongruence can be dealt.

This chapter focuses on the solution approach developed for integrated aircraft and crew recovery problems. By taking into account the dependency and correlation between the two problems a new sequential and iterative solution algorithm is developed in order to solve aircraft and crew recovery problems in real time without integrating them together into a huge problem. Thus, the complexity resulted from the integration will be reduced and both aircraft and crew disruptions will be recovered at one time.

The solution algorithm presented in this chapter involves multi-commodity network flow problem technique. In this solution algorithm both aircraft and crew recovery problems are developed as multi-commodity network flow models as described in Model A3 of aircraft recovery problem in Section 4.3 of Chapter 4 on

pages 111-115 and Model C1 of crew recovery problem in Section 5.1 of Chapter 5 on pages 134-139, respectively.

Another solution is developed for the integrated aircraft and crew recovery problems, which involves constraint programming embedded solution approach. This second solution algorithm will be presented in Chapter 7. The aircraft recovery problem is developed as explained in Model A4 of the aircraft recovery problem in Section 4.4. Similarly, the crew recovery problem is developed as explained in Model C2 of crew recovery problem in Section 5.2.

6.1 Solution Algorithm S1

Sequential and Iterative Solution Approach for Integrated Problem Using Multi-Commodity Network Flow Models

This solution approach involves multi-commodity network flow problems. In this solution algorithm, both the aircraft and crew recovery problems are developed as multi-commodity network flow models. The crew related and aircraft related constraints and variables are not emerged in one integrated model. Instead, two different problems for each recovery problem are developed and in order to involve the dependency and correlation between the resources, linking constraints are used.

As aircraft recovery problem, Model A3 of aircraft recovery problem, which has been explained in Chapter 4, in Section 4.3 on pages 111-114 is used. However, the crew related constraints of the Model A3 are needed to be modified. The details will be explained in Section 6.1.1.

As crew recovery problem, Model C1 of the crew recovery problem, which has been explained in Chapter 5, in Section 5.1 on pages 134-138 is used. However, a new constraint that computes the number of available crews for each flight leg needs to be added to the model. The details will be explained in Section 6.1.2.

The proposed solution algorithm is as follow:

1. Solve the airline crew recovery problem.
2. Obtain a new crew schedule for recovery period.
3. According to the obtained crew re-schedule compute the number of available crews.
4. Send the computed number of available crews to the aircraft recovery problem.
5. Use the number of available crews as a resource constraint in the aircraft recovery problem.
6. Solve the aircraft recovery problem by using number of available crews.
7. Obtain new aircraft-flight assignments for recovery period.
8. Compare the congruency of the solutions of the aircraft and crew recovery problems.
    - Are all the flight legs covered by a aircraft in both problems?
    - Are all the flight legs covered by crew in both problems?
    - If any flight leg is cancelled in one problem, is it also cancelled in the other problem?
    - If any flight leg is delayed in one problem, it is also delayed in the other problem?
9. If they are congruent Stop.
10. If not Go to Step 2 and send the solution of aircraft recovery problem to the crew recovery problem.

Below Algorithm 6.1 represents the solution algorithm of the integrated aircraft and crew recovery problem which will later be converted in ILOG script language.

The Figure 6.1 represents the flow of the proposed solution process described above.

Begin
Models : Crew (Crew recovery model), Aircraft (Aircraft recovery model).
Data : DF1 = 1, DF2 = 1.
Repeat
If (DF1≠ 0 & DF2≠ 0)
Solve model Crew.
For all flight legs:
Aircraft. NOAC[i] = Crew. NOAC[i] (number of available crew)
End {**for**}
Solve model Aircraft.
For all flight legs:
DF1= (Crew.cancel[i] - Aircraft.cancel[i])
DF2= (Crew.X[i,j].dept[i] - Aircraft. X[i,j].dept[i])
End {**for**}
End {**if**}
End {**repeat**}

End {algorithm}

Algorithm 6.1 The iterative solution algorithm for integrated aircraft and crew recovery algorithm using multi-commodity network flow models

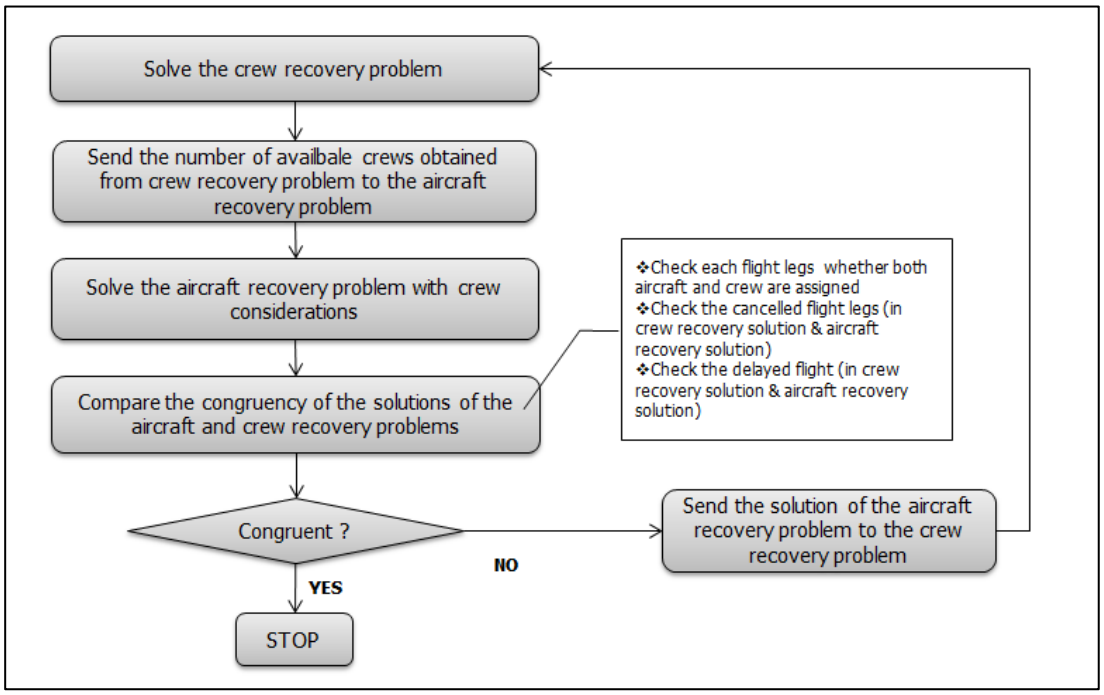

Figure 6.1 The sequential and iterative algorithm for integrated aircraft and crew recovery problem (Solution Approach S1)

6.2 The Aircraft Recovery Problem for the Solution Algorithm S1

The aircraft recovery problem used in the solution algorithm S1 is modeled based on the aircraft recovery problem Model A3 explained in Chapter 4, Sections 4.2.1 and 4.3.1. However, the crew related constraints of the Model A3 are needed to be modified.

In Model A3 presented in Chapter 4, in Sections 4.2.2 on pages 105-108 and 4.3.1 on pages 113-114 , there are 3 crew related constraints (4.25 - 4.27).

The first two ones (4.25 and 4.26) are used for calculating the number of available crews for each flight leg, at the departure station of flight leg and at the departure time of the flight leg. However, in solution algorithm S1, the number of available crew for each flight is computed in the crew recovery phase, during the recovered schedule is found. Therefore these two constraints (4.25 and 4.26) are not used in the aircraft recovery model of the solution algorithm S1. Therefore these two constraints are cancelled from the model.

$$Xac_i^k = ac_i^k \qquad \forall k \in K, \quad \forall\, i:(i,j) \in L^k \cup M^k \qquad (4.25)$$

$$Xac_i^k = ac_i^k - 1 \qquad \forall k \in K - \{R\}, \quad \forall\, i:(i,j) \in N^k \qquad (4.26)$$

The constraint (4.27) ensures that if there is not sufficient number of crews for flying any flight legs, this flight leg can not be flown. Thus, the decision variables for the aircraft- flight leg assignment will take value 0. The right hand side of this constraint (Xac_i^k) directly takes value from the solution of the crew recovery problem with the automatic calculation of number of available crew, which will be explained, in the next Section 6.1.2. This constraint may be considered as the linking constraint between two recovery problems.

$$\sum_{k \epsilon K-\{R\}} \sum_{j:(i,j) \in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k \leq Xac_i^k \qquad \forall i \in F^k \qquad (4.27)$$

After the above modification, the resulting mathematical formulation becomes as follows:

$$Minimize \quad \sum_{k\in K} \sum_{(i,j)\in N^k \cup M^k \cup L^k} c_{ij}^k \, X_{ij}^k + \sum_{k\in K} \sum_{(i,j)\in N_{30}^k \cup N_{60}^k} d_{ij}^k \, X_{ij}^k + \sum_{i\epsilon F} b_i \; Y_i \qquad (6.1)$$

Subject to:

$$\sum_{k\epsilon K} \sum_{j:(i,j)\in N^k \cup M^k \cup L^k} X_{ij}^k + \sum_{k\epsilon K} \sum_{j:(i,j)\in N_{30}^k \cup N_{60}^k} X_{ij}^k + Y_i = 1 \qquad \forall i \in F^k \qquad (6.2)$$

$$X_{ij}^k = 0 \qquad \forall k \in K - \{R\} \quad \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \qquad (6.3)$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K \;\; \forall (i,j) \in L^k \qquad (6.4)$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K \;\; \forall (i,j) \in M^k \qquad (6.5)$$

$$\sum_{j:(i,j)\epsilon N^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k - \sum_{j:(j,i)\epsilon N^k \cup N_{30}^k \cup N_{60}^k} X_{ji}^k = 0 \qquad \forall k \in R \;\; \forall i \in F^k \qquad (6.6)$$

$$X_{ij}^k \left(arr_i + turaround - dept_j\right) \geq 0 \qquad \forall k \in R \;\; \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \qquad (6.7)$$

$$X_{ij}^k \left(dest_i - org_j\right) = 0 \qquad \forall k \in R \;\; \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \qquad (6.8)$$

$$\sum_{(i,j)\in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k \leq lower \qquad \forall k \in K \qquad (6.9)$$

$$\sum_{(i,j)\in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k \geq upper \qquad \forall k \in K \qquad (6.10)$$

$$X_{ij}^k \epsilon \{0,1\} \qquad \forall k \in K \quad \forall (i,j) \in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k \qquad (6.11)$$

$$Y_i \epsilon \{0,1\} \qquad \forall i \in F \qquad (6.12)$$

$$\sum_{k\in K-\{R\}} \sum_{j:(i,j)\in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k} X_{ij}^k \leq Xac_i^k \qquad \forall i \in F^k \qquad (6.13)$$

where the objective function (6.1) minimizes the total cost of rerouting aircrafts, delaying flights and cancelling flights. The constraint (6.2) is the flight cover constraint and ensures that all flight legs will be either covered by an aircraft (delayed or on time) or cancelled. This constraint also ensures that flight legs can be either a member of rerouting connection network or delay connection network. With these constraints also logical connection between non-delayed and delayed flights is developed. For example, if the flight *j* is delayed 30 minutes in route (i, j), flight *j* should also be found 30 minutes delayed in its next connection. The constraint (6.3) ensures that during recovery period no flights will be assigned to the shortage aircrafts. The constraints (6.4) and (6.5) ensure that before and after the recovery period the aircraft assignment will be same as the original schedule. In other words, it will be returned to the original flight aircraft assignment after the shortage aircrafts is recovered. The constraint (6.6) is the flow conservation constraint, which ensures that the total arc flow, which enters the node, *i* must be equal to the total arc flow, which leaves the node *i* for the available aircrafts during recovery period. The constraint (6.7) guarantees the time requirements between consecutive flight legs: there should be at least the minimum required turnaround time between the arrival of the flight *i* and the departure of the flight *j* for the aircrafts available during recovery period. The constraint (6.8) ensures that the destination station of the first flight and the origin of the second flight are the same for the aircrafts available during recovery period. Constraints (6.9) and (6.10) guarantees the usage balance for the aircrafts ensuring the upper and lower bounds on the number of flight legs that an aircraft can fly. Constraints (6.11) and (6.12) ensure that all the decision variables are binary. The constraint (6.13) ensures that if there is not sufficient number of crews for flying any flight legs, this flight leg can not be flown. Thus, the decision variables for the aircraft- flight leg assignment will take value 0.

The notation regarding above mathematical formulation can be found in Chapter 4, in Section 4.2.2, on pages 105 and 106.

6.3 The Crew Recovery Problem for the Solution Algorithm S1

The crew recovery problem used in the solution algorithm S1 is modeled based on the crew recovery problem Model C1 explained in Chapter 5, in Section 5.1.1 on pages 135-139. However, a new constraint which computes the number of available crew at certain time at certain station is included in the crew recovery solution.

$$NC_h = 1 - Y_h + \sum_{k \in K} \sum_{\substack{(i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \\ arr[i] < dept[h] \\ dept[j] > arr[h]}} X_{ij}^k \qquad \forall h \in F$$

The above constraint is used for computing the number of available crews at the departure station of each flight leg, at the departure time of each flight leg. The result of this computation is used by the aircraft recovery problem in constraint (6.13) explained in Section 6.1.1 on page 161, the total number of available crews is computed as follows:

- The crew originally assigned to flight h is taken into consideration.
- If the flight h is cancelled the sum will be affected accordingly
- For flight h, the total number of crews who are stationed on station $org[h]$ waiting for their next flight leg between the departure and arrival time of the flight h is computed.

where;

h : index for flight legs

NC_h : total number of available crews that can fly flight leg h

The rest of the notation regarding this model has been explained in Chapter 5, in Section 5.1.1 on page 135.

After the addition of this new constraint, the resulting mathematical formulation of the problem becomes as follows:

$$Minimize \sum_{k\in K}\sum_{(i,j)\in N^k\cup M^k\cup L^k} c_{ij}^k X_{ij}^k + \sum_{k\in K}\sum_{(i,j)\in N_{30}^k\cup N_{60}^k} d_{ij}^k X_{ij}^k + \sum_{i\epsilon F} b_i\ Y_i\ + \sum_{i\epsilon F} r_i\ Z_i \quad (6.14)$$

Subject to:

$$\sum_{k\epsilon K}\sum_{j:(i,j)\in N^k\cup M^k\cup L^k} X_{ij}^k\ + \sum_{k\in K}\sum_{(i,j)\in N_{30}^k\cup N_{60}^k} X_{ij}^k + Y_i + \ Z_i = 1 \qquad \forall i \in F^k \quad (6.15)$$

$$X_{ij}^k = 0 \qquad \forall k \in K - \{R\}\ \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \quad (6.16)$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K\ \ \forall (i,j) \in L^k \quad (6.17)$$

$$X_{ij}^k = sch_{ij}^k \qquad \forall k \in K\ \ \forall (i,j) \in M^k \quad (6.18)$$

$$X_{ij}^k\left(arr_i + sittime - dept_j\right) \geq 0 \qquad \forall k \in R\ \ \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \quad (6.19)$$

$$X_{ij}^k\left(dest_i - org_j\right) = 0 \qquad \forall k \in R\ \ \forall (i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \quad (6.20)$$

$$\sum_{j:(i,j)\epsilon N^k\cup N_{30}^k\cup N_{60}^k} X_{ij}^k - \sum_{j:(j,i)\epsilon N^k\cup N_{30}^k\cup N_{60}^k} X_{ji}^k = 0 \qquad \forall k \in R\ \ \forall i \in F^k \quad (6.21)$$

$$\sum_{(i,j)\in N^k\cup M^k\cup L^k\cup N_{30}^k\cup N_{60}^k} X_{ij}^k \leq lower \qquad \forall k \in K \quad (6.22)$$

$$\sum_{(i,j)\in N^k\cup M^k\cup L^k\cup N_{30}^k\cup N_{60}^k} X_{ij}^k \geq upper \qquad \forall k \in K \quad (6.23)$$

$$\sum_{i:(i,j)\in N^k\cup M^k\cup L^k\cup N_{30}^k\cup N_{60}^k} X_{ij}^k\,(arr_i - dept_i) \leq flytime \qquad \forall k \in K \quad (6.24)$$

$$\sum_{i:(i,j)\in N^k\cup M^k\cup L^k\cup N_{30}^k\cup N_{60}^k} X_{ij}^k\,(dept_i - arr_j) \leq dutytime \qquad \forall k \in K \quad (6.25)$$

$$\sum_{i \in F} Z_i \, (dept_j - arr_i) \leq flytime \tag{6.26}$$

$$\sum_{i \in F} Z_i \, (dept_i - arr_j) \leq dutytime \tag{6.27}$$

$$X_{ij}^k \epsilon \{0,1\} \qquad \forall k \in K \; \forall (i,j) \in N^k \cup M^k \cup L^k \cup N_{30}^k \cup N_{60}^k \tag{6.28}$$

$$Y_i \epsilon \{0,1\} \qquad \forall i \in F \tag{6.29}$$

$$Z_i \epsilon \{0,1\} \qquad \forall i \in F \tag{6.30}$$

$$NC_h = 1 - Y_h + \sum_{k \in K} \sum_{\substack{(i,j) \in N^k \cup N_{30}^k \cup N_{60}^k \\ arr[i] < dept[h] \\ dept[j] > arr[h]}} X_{ij}^k \qquad \forall h \in F \tag{6.31}$$

where the objective function (6.14) minimizes the total cost of reassigning crews to the flight legs, using reserve crew, delaying flight legs and cancelling flights. The constraint (6.15) is the flight cover constraint and ensures that all flight legs will be either covered by a good crew (delayed or on time) or by a reserve crew or cancelled. The constraint (6.16) ensures that during recovery period no flights will be assigned to the shortage crew. The constraints (6.17) and (6.18) ensure that before and after the recovery period the crew assignment will be same as the original schedule. In other words, it will be returned to the original crew flight assignment after the shortage crew becomes available again. The constraint (6.19) guarantees the time requirements between consecutive flight legs: there should be at least the minimum required sit time between the arrival of the flight *i* and the departure of the flight *j* for the available crews during recovery period. The constraint (6.20) ensures that the destination station of the first flight and the origin of the second flight are the same for the available crews during recovery period. The constraint (6.21) ensures that the total arc flow, which enters the node, *i* must be equal to the total arc flow, which leaves the node *i* for the available crews during recovery period. Constraints (6.22) and (6.23) limit the number of flight legs to be flown by a crew. Constraint (6.24) limits the total flying time for a crew. A crew can not fly more than predefined total

flying time in one pairing. Constraint (6.25) limits the duty time for a crew. Total time spent on duty by a crew can not be more than predefined total duty time. Constraint (6.26) limits the total flying time for a reserve/standby crew. Constraint (6.27) limits the total duty time for a reserve/standby crew. The constraints (6.28) - (6.30) ensure that all the decision variables are binary. Finally, the constraint (6.31) computes the total number of available crew at certain time at certain station.

6.4 Numerical Example for Solution Algorithm S1

The aircraft and crew recovery models in the proposed solution algorithm S1 are modeled in OPL Studio 3.7. The OPL Model of the aircraft recovery problem can be seen in Appendix E and the OPL Model of the crew recovery problems can be seen in Appendix G.

In order to provide interaction between the steps explained previously in Section 6.1 and the involved recovery problems explained in Sections 6.1.1 and 6.1.2 (crew recovery and aircraft recovery problems), a script code is written using OPL Script tool. The script code of the problem can be seen in Appendix I. The subject OPL script code is written with reference to the algorithm represented in Algorithm 6.1.

The problem is solved using the domestic flight schedule of OnurAir as seen in Table A.2. In order to solve the problem, following input data is used;

- The flight schedule of the domestic flight legs
- The initial aircraft-flight leg assignment of OnurAir
- The initial crew assignment of OnurAir
- The connection networks of the aircrafts (before recovery period, after recovery period, during recovery period - on time, 30 minutes of delay and 60 minutes of delay-)
- The connection network of the crews (before recovery period, after recovery period, during recovery period - on time, 30 minutes of delay and 60 minutes of delay-)
- The reassignment cost, delay cost and cancellation costs

Several cases are tested in order to evaluate the efficiency of the approach: Multiple aircraft disruptions, both aircraft and crew disruption of the same flight leg, and also aircraft and crew disruptions for different flight legs. Only an example case's results are presented in this section.

Table 6.1 Recovered schedules obtained from the solution of Solution Algorithm S1

Original Schedule							AIRCRAFT RECOVERY Shortage occurs on Aircraft MD88e at 06:00		CREW RECOVERY Shortage occurs on Crew 3 at 06:00	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Crew	Aircraft	Recovery strategy	Crew	Recovery strategy
OHY010	IST	ADA	6:45	8:15	MD83c	C010	MD83d	S	C10	-
OHY011	ADA	IST	9:10	10:40	MD83c	C010	MD83d	S	C10	-
OHY016	IST	ADA	15:00	16:30	MD83c	C04	MD83c	-	C04	-
OHY017	ADA	IST	17:30	19:00	MD83c	C04	MD83c	-	C04	-
OHY018	IST	ADA	19:00	20:30	MD88a	C02	MD88a	-	C02	-
OHY019	ADA	IST	21:30	23:00	MD88a	C02	MD88a	-	C02	-
OHY022	IST	AYT	8:15	9:15	MD88c	C02	MD88d	S	C02	-
OHY023	AYT	IST	10:45	11:45	MD88c	C02	MD88d	S	C02	-
OHY024	IST	AYT	16:45	17:45	MD83b	C09	MD83b	-	C09	-
OHY025	AYT	IST	18:45	19:45	MD83b	C09	MD83b	-	C09	-
OHY026	IST	AYT	20:30	21:30	MD83d	C01	MD83d	-	C01	-
OHY027	AYT	IST	7:30	8:30	MD83b	C01	MD83a	S	C01	-
OHY028	IST	BJV	11:55	12:55	MD83d	C10	MD83d	-	C10	-
OHY029	BJV	IST	13:45	14:45	MD83d	C10	MD83d	-	C10	-
OHY034	IST	DIY	6:45	8:30	MD88a	C11	MD88c	S	Reserve	R
OHY035	DIY	IST	9:30	11:15	MD88a	C11	MD88c	S	Reserve	R
OHY036	IST	DIY	18:45	20:30	MD88b	C06	MD88b	-	C06	-
OHY037	DIY	IST	21:30	23:15	MD88b	C06	MD88b	-	C06	-
OHY038	IST	ERZ	12:30	14:20	MD88c	C08	MD88c	-	C08	-
OHY039	ERZ	IST	15:20	17:10	MD88c	C08	MD88c	-	C08	-
OHY042	IST	GZT	9:30	11:15	MD83a	C05	MD83a	-	C05	-
OHY043	GZT	IST	12:15	14:00	MD83a	C05	MD83a	-	C05	-
OHY050	IST	ADB	7:45	8:45	MD88d	C04	MD83b	S	C04	-
OHY051	ADB	IST	10:50	11:45	MD88d	C04	MD83b	S	C04	-
OHY054	IST	ADB	16:30	17:30	MD88d	C03	MD88d	-	C03	-
OHY055	ADB	IST	18:50	19:45	MD88d	C03	MD88d	-	C03	-
OHY058	IST	ADB	20:45	21:45	MD88e	C11	MD88e	-	C11	-
OHY059	ADB	IST	7:30	8:25	MD83d	C06	MD83c	S	C01	S
OHY062	IST	MLX	12:15	13:45	MD88e	C01	MD88e	-	C01	-
OHY063	MLX	IST	14:30	16:00	MD88e	C01	MD88e	-	C01	-
OHY066	IST	KSY	7:15	8:30	MD88e	C03	MD88a	S	C09	S
OHY067	KSY	IST	9:15	10:30	MD88e	C03	MD88a	S	C09	S
OHY074	IST	SZF	9:30	10:45	MD83b	C09	MD83c	S	C11	S
OHY075	SZF	IST	11:45	13:00	MD83b	C09	MD83c	S	C11	S
OHY080	IST	TZX	6:50	8:25	MD88b	C07	MD88b	-	C07	-
OHY081	TZX	IST	9:20	10:55	MD88b	C07	MD88b	-	C07	-
OHY086	IST	TZX	18:45	20:20	MD83a	C12	MD83a	-	C12	-
OHY087	TZX	IST	21:15	22:50	MD83a	C12	MD83a	-	C12	-

* Recovery Strategies
R : using reserve/standby crew, S : swapping crews/aircrafts, D : delaying flights , C : cancelling flights

As an example case, there occurs a shortage on crew "Crew 3" and on aircraft "MD88e" between 06:00 and 12:00. During the recovery period crew "Crew3" and aircraft "MD88e" which is originally assigned to flight legs "OHY066" and

“OHY067” can not perform their corresponding flight legs. The problem is to find a feasible aircraft re-assignment and crew re-schedule during recovery period with limited available resources by minimizing the total operating and disruption costs.

6.5 Evaluation of the Proposed Solution Algorithm S1

The solution approach presented in this chapter finds solutions for both aircraft recovery and crew recovery problems. The aircraft recovery and crew recovery problems are modeled as multi-commodity network flow problems where the underlying network is connection network. The solution algorithm solves both aircraft and crew recovery problems by taking into account the dependency of two problems, representing the correlation between them without integrating the two recovery problems.

With this solution algorithm, both aircraft and crew disruptions are recovered at one time. The overall algorithm does not include any integration of aircraft and crew recovery problems. Thus, the problem complexity and huge number of constraints and variables are prevented; the real time solutions are obtained in easier and more practical way. Furthermore, it is possible to consider several combinations of dependent or independent aircraft and crew unavailabilities through the algorithm.

The performance of the proposed solution algorithm is also satisfactory. The Table 6.2 summarizes the solution time and problem size in terms of number of constraints and variables.

The Table 6.2 summarizes the solution times when the aircraft recovery problem is solved independently; when crew recovery problem is solved independently and when interactive solution approach is applied.

Table 6.2 The solution times and the problem size obtained from Solution Algorithm S1

Crew Recovery Solutions					Aircraft Recovery Solutions					Integrated Recovery
Recovery period	# of flight legs in recovery period	#of variables	#of constraints	Solution time (in sec)	Recovery period	# of flight legs in recovery period	#of variables	#of constraints	Solution time (in sec)	Solution time (in sec)
06:00 - 12:00	15	2888	3395	1.530	06:00 -12:00	15	6428	18719	1.720	3.385
06:00 - 12:00	15	2888	3395	1.530	06:00 -12:00	15	6428	18719	1.680	3.420
06:00 - 12:00	15	2888	3395	1.530	12:00- 18:00	9	6428	18719	1.370	2.528
06:50 -11:00	10	2888	3437	0.780	06:50 -11:00	10	8102	18719	1.470	3.323
18:45 - 00:00	10	2888	3607	0.840	16:45 -19:45	4	6860	18719	1.280	1.544
18:45 - 00:00	10	2888	3607	0.840	06:00 -11:30	13	8017	18719	0.980	1.653
06:45 - 11:15	13	2888	3462	0.750	06:00 -11:30	13	8017	18719	0.980	1.670
06:45 - 11:15	13	2888	3462	0.750	08:00-12:00	9	7492	18719	0.950	1.669
08:00 - 12:00	9	2888	3254	0.590	08:00-12:00	9	7492	18719	0.950	1.653
19:00 - 23:00	5	2888	3528	0.700	08:00-12:00	9	7492	18719	0.950	2.746

As it can be seen from the Table 6.2, the proposed solution algorithm can produce both aircraft and crew recovery solutions in real time, in about 2.4 seconds. In order to satisfy the congruency of the crew and aircraft recovery solutions, the algorithm iterates between two problems. The solution times more than 3 seconds indicate that the number of iterations for these cases is greater than the other cases.

The advantages of the proposed solution algorithm can be summarized as follows:

The proposed solution algorithm finds recovery solutions for both aircraft and crew disruptions at one time, connecting aircraft and crew recovery problems together via linking constraints without combining these problems' constraints into a huge model.

Our way of integration is performed by connecting two problems via linking constraints rather than developing fully integrated problem, which includes both aircraft and crew related constraints into a huge model, as done in the classical way of integration in the literature. Therefore, the problem complexity and huge number of constraints and variables, coming from the classical way of integration, are prevented. Thus, the overall problem is less complex than the integrated problems developed in classical way, in terms of modeling and the real time solutions can be obtained in easier and more practical way.

Several combinations of aircraft and crew availabilities can be considered. The algorithm can find solutions for multiple dependent or independent crew and aircraft unavailabilities: In some cases only aircraft disruptions can be recovered. In some cases aircraft and crew disruptions for same flight leg can be considered or aircraft unavailability for flight *a* and crew unavailability for flight *b* can be solved.

CHAPTER SEVEN
ITERATIVE SOLUTION APPROACH FOR INTEGRATED AIRCAFT AND CREW RECOVERY PROBLEMS USING CONSTRAINT PROGRAMMING

The challenges of the integrated aircraft and crew recovery problems and the importance of solving aircraft and crew recovery problems together have already been mentioned several times in the previous chapters of this dissertation. Furthermore, in Chapter 6, the first solution algorithm, which is developed for integrated aircraft and crew recovery problems have been presented. The solution algorithm presented in Chapter 6 finds recovery solutions for both aircraft and crew disruptions, connecting aircraft and crew recovery problems via linking constraints, without combining aircraft and crew related constraints in a one huge and unmanageable sized model, using multi-commodity network flow technique. Such integration prevents the challenges coming from the classical way of integration (combining both aircraft and crew related constraint into one model), becomes more manageable and easier to be solved. Therefore, in this chapter, a second solution algorithm is proposed in order to solve aircraft and crew recovery problems integrated in our way.

The second solution algorithm, S2 for integrated aircraft and crew recovery problem involves constraint programming embedded solution approach. The concept and the steps of this solution algorithm is similar to the solution algorithm S1 which has been explained in Section 6.1. However, the modeling technique, which is used for developing aircraft and crew recovery problems, is not multi-commodity flow models. Instead, the constraint programming embedded solution approach is used in order to develop aircraft and crew recovery problems.

The solution algorithm S2 which will be proposed in this chapter is the most improved solution approach among all models and solution approaches explained in this dissertation. The solution algorithm S2 has following contributions:

The first contribution is developing a system, which can solve integrated aircraft and crew recovery using iterative and sequential solution approach.

- As mentioned previously, the most of the studies in the literature focuses on aircraft recovery and crew recovery problems separately. While solving airline resource scheduling problems, the crew related and aircraft related decisions affect each other. In order to operate any flight leg there must be at least 1 available crew and 1 available aircraft to be assigned to the flight leg. By solving crew recovery and aircraft recovery problems iteratively, both aircraft assignment and crew reschedule are integrated and their availabilities are considered together.

- Rather than fully integration, such iterative solution approach is more efficient. Because integrated problem is very challenging and complex in terms of modeling. The integrated problem includes both crew related and aircraft related constraints and variables. Thus, the number of constraints and variables become very huge compared to the individual recovery problems. Because of the increased complexity and number of variables, the integrated problem is NP-hard. By solving crew recovery and aircraft recovery problems iteratively, the problem complexity and the solution time will decrease.

The second contribution is including constraint programming technique into the solution process of aircraft and crew recovery problems.

- Although Constraint Programming approach is widely applied to scheduling problems and NP-hard problems, there is not a published work in the literature, which solves the airline recovery problems with constraint programming technique. However, the constraint programming is a powerful tool that can be used as an alternative method to mathematical programming techniques and offers a more flexible modeling framework than mathematical programming. Aircraft and crew recovery problems need such flexible modeling framework because of the complexity of their constraints.

Since the solution algorithm S2 is the most improved solution approach in this dissertation, the efficiency and performance of this method is tested for further recovery scenarios using different data sets. For this purpose, not only the data of OnurAir but also the data obtained from other 2 different airlines operating Turkish domestic flights is used.

The proposed solution approach which includes both constraint programming and integer programming in the solution procedure is tested for different recovery scenarios using different data sets. For this purpose, the data obtained from 3 different airlines operating daily domestic flights in Turkey is used:

1. OnurAir Airlines' daily domestic flights (see Appendix A in Table A.1 for fleet information of OnurAir & Table A.3. for initial schedule of OnurAir)

- The OnurAir operates 38 flight legs in a day.
- All the flights are flown within the same day over a network of 12 stations.
- There are 2 different fleet types and total of 9 aircrafts: Boeing MD-88 and Boeing MD-83. There are 5 aircrafts of Boeing MD-88 type and 4 aircrafts of Boeing MD-83 type. Both types of aircrafts have the same technical specifications and are capable of flying all the domestic flights of the airline.
- The original crew schedule of OnurAir composes of 12 crews.

2. AtlasJet Airlines' daily flights (see Appendix A, Table A.5 for fleet information of AtlasJet & Table A.6 for initial schedule of AtlasJet)

- The AtlasJet operates 38 flight legs in a day.
- All the flights are flown within the same day over a network of 10 stations.
- There are total of 9 aircrafts: 6 Airbus and 3 Boeing type. All aircrafts are capable of flying all the flights of the airline.
- The original crew schedule of OnurAir composes of 12 crews.

3. Pegasus Airlines' daily domestic flights (see Appendix A, Table A.7 for fleet information of Pegasus Airlines & Table A.8 for initial schedule of Pegasus Airlines)

- The Pegasus operates 164 domestic flight legs in a day.
- All the flights are flown within the same day over a network of 20 stations.
- There are total of 36 aircrafts: 5 Airbus and 31 Boeing type. All aircrafts are capable of flying all the flights of the airline.
- The original crew schedule of OnurAir composes of 42 crews.

7.1 Solution Algorithm S2

Sequential and Iterative Solution Approach for Integrated Problem using Constraint Programming Embedded Solution Approach

In solution algorithm S2, both the aircraft and crew recovery problems are developed using constraint programming embedded solution approach. The crew related and aircraft related constraints and variables are not emerged in one integrated model. Instead, two different problems for each recovery problem are developed and in order to involve the dependency and correlation between the resources, linking constraints are used as in Solution algorithm S1 which has been explained in Chapter 6, on pages 154-169.

As aircraft recovery problem, Model A4 of aircraft recovery problem, which has been explained in Chapter 4, in Section 4.4 on pages 118-126 is used. Therefore, the aircraft recovery problem is modeled as two-stage problem: the first one is the constraint programming which generates feasible routes for each available aircraft during the recovery period; the second one is the integer programming which selects the best set of routes minimizing the total recovery costs and the changes in the initial schedule.

As crew recovery problem, Model C2 of the crew recovery problem, which has been explained in Chapter 5, in Section 5.2 on pages 141-148 is used. Therefore, the crew recovery problem is modeled as two-stage problem: the first one is the constraint programming which generates feasible crew pairings for each available

crew during the recovery period; the second one is the integer programming which selects the best set of crew pairings minimizing the total recovery costs and the changes in the initial schedule.

The solution procedure of the proposed algorithm S2 is as follows:

1. Providing initial conditions:

    1.1. Crew scheduling problem is solved using constraint programming in order to provide initial conditions of the problem.
    1.2. The initial crew-flights assignments are defined as input data for crew recovery problem.
    1.3. Aircraft routing problem is solved using constraint programming in order to provide initial conditions of the problem.
    1.4. The initial aircraft-flights assignments are defined as input data for aircraft recovery problem.

2. Generating aircraft and crew routes for recovery period:

    2.1. Constraint programming model is solved for generating feasible crew pairings for each crew including each flight leg scheduled to be flown during recovery period.
    2.2. Constraint programming model is solved for generating routes for each aircraft including each flight leg scheduled to be flown during recovery period.

3. Solving crew recovery optimization problem:

    3.1. Newly generated crew pairings (in step 2.1.) are sent to integer programming problem, which selects the optimum set of crew pairings while satisfying recovery period start/ end conditions and minimizing the recovery costs and the changes in the original schedule

3.2. The integer programming problem is solved and the best set of pairings which covers all the flight legs and minimizes the recovery costs, is obtained.

4. Determining number of available crews according to the solution obtained from crew recovery problem: The number of available crew is computed for each flight leg, according to following parameters:

- The crew already assigned to the subject flight leg
- The crews positioned on floor at departure time of the subject flight leg, where the positioned station is the origin station of the subject flight leg.

5. Sending the computed number of available crews to the aircraft recovery problem: The computed number of available crews for each flight leg is input data for aircraft recovery problem and used in the crew related constraint of the problem.

6. Solving aircraft recovery optimization problem:

6.1. Newly generated aircraft routes (in Step 2.2.) is sent to integer programming problem, which selects the optimum set of aircrafts routes while satisfying recovery period start/ end conditions and minimizing the recovery costs and the changes in the original schedule.

6.2. The integer programming problem is solved and the best set of aircraft routes which covers all the flight legs and minimizes the recovery costs, is obtained.

7. Comparison of the congruency of the solutions of the aircraft and crew recovery problems:

- Are all the flight legs covered by a aircraft in both problems?
- Are all the flight legs covered by crew in both problems?

- If any flight leg is cancelled in one problem, is it also cancelled in the other problem?
- If any flight leg is delayed in one problem, it is also delayed in the other problem?

8. Are they congruent?

- Yes → Stop
- No → Go to Step 3. Send the solution of aircraft recovery integer programming problem to crew recovery integer programming problem

A new algorithm is developed and with reference to such algorithm an OPL Script code is written which will solve aircraft and crew recovery problems in iterative and sequential manner, taking into account the both crew and aircraft availabilities and correlation between them. Such algorithm includes both constraint programming and integer programming techniques. The Algorithm 7.1 represents the solution algorithm of the integrated aircraft and crew recovery problem which will later be converted in ILOG script language.

In order to provide interaction and link between the stages listed above, a code has been written using OPL Script tool. The overall solution procedure is controlled by OPL Script which can be seen in Appendix J.

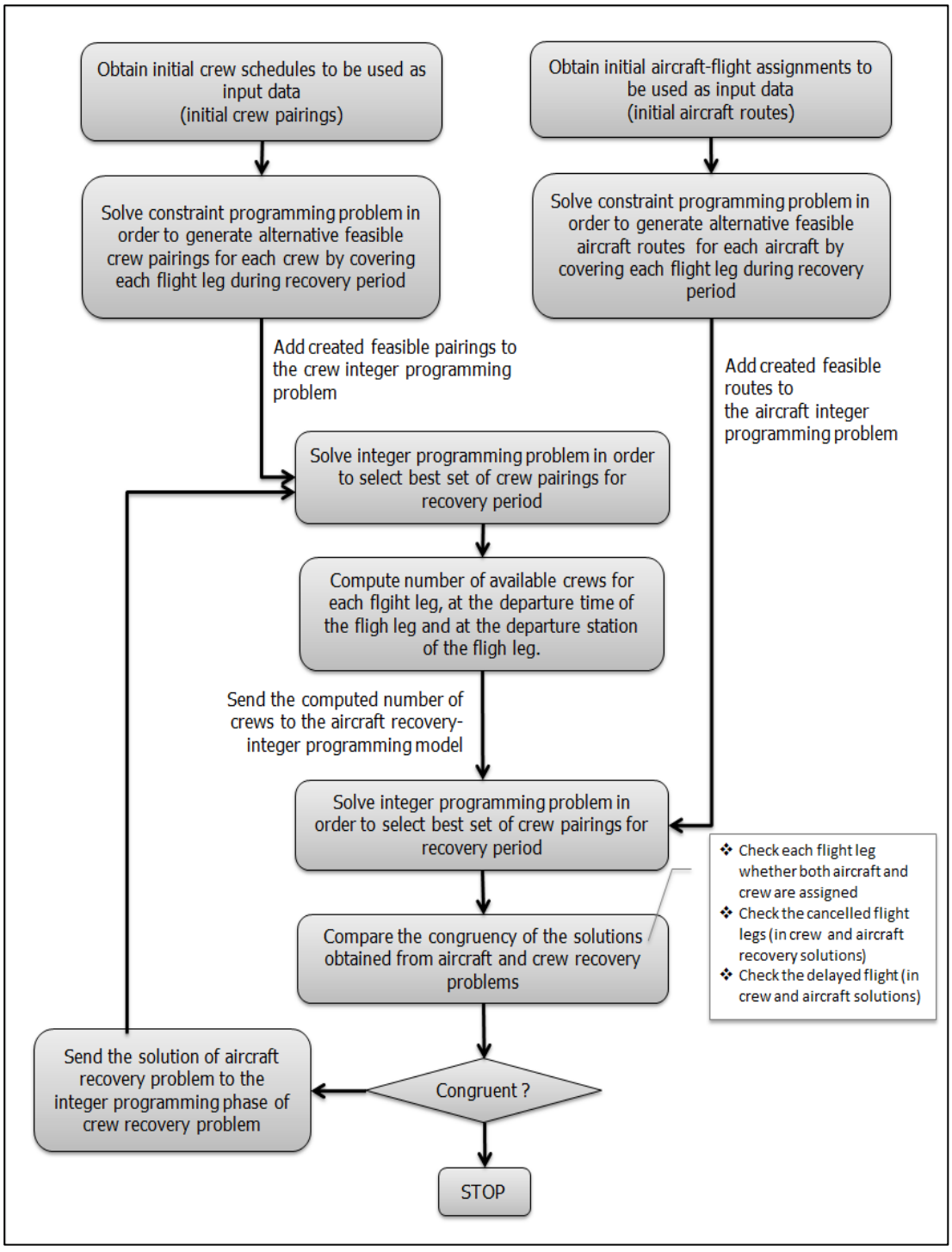

Figure 7.1 The solution procedure of the Solution Algorithm S2

```
Begin
Models :  Crew_initial (Crew scheduling model for obtaining initial crew schedule),
          Crew_pairings (Constraint programming model for generating crew pairings),
          Crew_int (Integer programming model for finding optimized crew reschedule),
          Aircraft_initial (Aircraft routing model for obtaining initial aircraft routes),
          Aircraft_routes (Constraint programming model for generating aircraft routes),
          Aircraft_int (Integer programming model for obtaining optimized reassignments)

Data : DF1 = 1, DF2 = 1.
Data : RP (Recovery period)
Data : Pairings[Crews, int+]
Data: Routes [Aircrafts, int+]

Solve Crew_initial.
Solve Aircraft_initial.

For all flight legs in RP.
   Solve Crew_pairings.
   Add Solution (Crew_pairings) to Pairings[Crews, int+]
End{for}

For all flight legs in RP.
   Solve Aircraft_routes
   Add Solution (Aircraft_routes) to Routes[Aircrafts, int+]
End{for}

Repeat
        If (DF1≠ 0 & DF2≠ 0)
        Solve model Crew_int using Pairings [Crews, int+].
                For all flight legs:
                Aircraft. NOAC[i] = Crew. NOAC[i] (number of available crew)
                End {for}
        Solve model Aircraft_int using Routes[Aircrafts, int+].
                For all flight legs:
                DF1= (Crew.cancel[i] - Aircraft.cancel[i] )
                DF2= (Crew.X[i,j].dept[i] - Aircraft. X[i,j].dept[i] )
                End {for}
        End {if}
End {repeat}
End {algorithm}
```

Algorithm 7.1 The iterative solution algorithm for integrated aircraft and crew recovery algorithm using constraint programming

The OPL script code seen in Appendix J, first creates input data of initial aircraft-flight and crew-flight assignments. Then it calls the two constraint programming models sequentially for each flight leg in the recovery period.

- CP model for crew pairings: Constraint programming model for generating feasible crew pairings sequentially for each crew and for each flight leg in the recovery

- CP model for aircraft routes: Constraint programming model for generating feasible aircraft routes sequentially for each aircraft for each flight leg in the recovery period.

When the routes and pairings are generated for each aircraft/crew and for all selected flight legs, the script calls the second-stage two integer programming models sequentially, providing link between them:

- IP model for crew recovery: Integer programming model which selects the optimum set of crew pairings while minimizing the recovery costs and changes in original schedule.

- IP model for aircraft recovery: Integer programming model which selects the optimum set of aircraft routes while minimizing the recovery costs and changes in original schedule.

7.2 Constraint Programming Model for Generating Crew Pairings (Step 2.1)

This constraint programming model generates feasible crew pairings sequentially for each crew and for every selected flight legs (flight legs in the recovery period). For each crew the pairings including 4 flight legs and 2 flight legs will be developed.

The constraint programming model is called for each flight leg in the recovery period. The variable “cover” representing flight legs during recovery period is

imported from OPL Script to the constraint programming model for each flight leg in the recovery period.

The constraint programming models for generating crew pairings used in the Solution algorithm S2 is the same as the constraint programming model used in the crew recovery model C2, which has been explained in Section 5.2.1 on pages 143-146. In this section, for the sake of convenience, we are presenting the mathematical formulation of the model again:

$$CitySeq_{c,i-1} = dest_{seq_{c,i}} \qquad \forall c \in S, \ \forall i \in P \qquad (5.18)$$

$$CitySeq_{c,i} = org_{seq_{c,i}} \qquad \forall c \in S, \ \forall i \in P \qquad (5.19)$$

$$CitySeq_{c,i-1} = CitySeq_{c,i} \qquad \forall c \in S, \ \forall i \in P \qquad (5.20)$$

$$AcArr_{seq_{c,i}} + MinStop \leq AcDept_{seq_{c,i+1}} \qquad \forall c \in S, \ \forall i \in R \qquad (5.21)$$

$$Delay_{seq_{c,i}} = AcDept_{seq_{c,i+1}} - dept_{seq_{c,i+1}} \qquad \forall c \in S, \ \forall i \in R \qquad (5.22)$$

$$Delay_{seq_{c,i}} \leq 90 \qquad \forall c \in S, \ \forall i \in R \qquad (5.23)$$

$$CitySeq_{c,0} = CitySeq_{c,n} \qquad \forall c \in S, \ n = 2 \ or \ n = 4 \qquad (5.24)$$

$$\left(\sum_{i \in P} seq_{c,i} = f \right) \leq 1 \qquad \forall c \in S, \ \forall f \in F \qquad (5.25)$$

$$\left(\sum_{i \in P} seq_{c,i} = cover \right) = 1 \qquad \forall c \in S \qquad (5.26)$$

$$\sum_{i \in P} \left(arr_{seq_{c,i}} - dept_{seq_{c,i}} \right) \leq MaxFlight \qquad \forall c \in S \qquad (5.27)$$

$$arr_{seq_{c,n}} - dept_{seq_{c,0}} \leq MaxDuty \qquad \forall c \in S, \ n = 2 \ or \ n = 4 \qquad (5.28)$$

The notation used in the problem and the definition of the constraints have been explained in Section 5.2.1 on pages 144 -146.

7.3 Constraint Programming Model for Generating Aircraft Routes (Step 2.2)

Similar to the problem in Step 2.1, this constraint programming model generates feasible routes sequentially for each aircraft and for every selected flight legs (flight legs in the recovery period). For each aircraft the routes including 4 flight legs and 6 flight legs have been developed.

The constraint programming model is called for each flight leg in the recovery period. The variable "cover" representing flight legs during recovery period is imported from OPL Script to the constraint programming model for each flight leg in the recovery period.

The constraint programming models for generating aircraft routes, used in the Solution algorithm S2 is the same as the constraint programming model used in the aircraft recovery model A4, which has been explained in Section 4.4.1 on pages 121-123. In this section, for the sake of convenience, we are presenting the mathematical formulation of the model again:

$$CitySeq_{a,i-1} = dest_{seq_{a,i}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.28)$$

$$CitySeq_{a,i} = org_{seq_{a,i}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.29)$$

$$CitySeq_{a,i-1} = CitySeq_{a,i} \qquad \forall a \in A, \ \forall i \in R \qquad (4.30)$$

$$AcArr_{seq_{a,i}} + MinStop \leq AcDept_{seq_{a,i+1}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.31)$$

$$Delay_{seq_{a,i}} = AcDept_{seq_{a,i+1}} - dept_{seq_{a,i+1}} \qquad \forall a \in A, \ \forall i \in R \qquad (4.32)$$

$$Delay_{seq_{a,i}} \leq 90 \qquad \forall a \in A, \ \forall i \in R \qquad (4.33)$$

$$CitySeq_{a,0} = CitySeq_{a,n} \qquad \forall a \in A, \quad n = 4 \; or \; n = 6 \qquad (4.34)$$

$$\left(\sum_{i \in R} seq_{a,i} = f \right) \leq 1 \qquad \forall a \in A, \quad \forall f \in F \qquad (4.35)$$

$$\left(\sum_{i \in R} seq_{a,i} = cover \right) = 1 \qquad \forall a \in A \qquad (4.36)$$

The notation used in the problem and the definition of the constraints have been explained in Section 4.4.1 on pages 121 - 123.

7.4 Integer Programming Model for Crew Recovery Problem (Step 3.2)

This integer program selects the best set of crew pairings for the recovery period, satisfying certain constraints.

After the completion of crew pairings generation during recovery period in Step 2.1, OPL script calls the integer programming model. This model selects the optimum set of pairings during the recovery period while:

- minimizing the changes in original schedule (minimizing the deviation between original schedule and recovered schedule)
- minimizing the total cancellation costs
- minimizing the total crew reassignment costs
- satisfying that before and after recovery periods crews will return to their original schedule

The integer programming models for crew recovery problem, used in the Solution algorithm S2 is the same as the integer programming model used in the crew recovery model C2, which has been explained in Section 5.2.2 on pages 146-148. In this section, for the sake of convenience, we are presenting the mathematical formulation of the model again:

$$Minimize \quad \sum_{c\in S}\sum_{i\in F_r^c} C_{ci}X_{ci} + \sum_{c\in S}\sum_{i\in F} W[X_{ci} - P_{ci}] + \sum_{i\in F} D_i Y_i + \sum_{i\in F} E_i Z_i + \sum_{c\in S}\sum_{i\in F_r^c} DC_i X_{ci} \tag{5.29}$$

Subject to:

$$\sum_{c\in A} X_{ci} + Y_i + Z_i = 1 \qquad \forall i \in F \tag{5.30}$$

$$\sum_{i\in F_r^c} X_{ci} = 1 \qquad \forall c \in S \tag{5.31}$$

$$X_{ci} = P_{ki} \qquad \forall c \in S - \{A\} \qquad \forall i \in F : dept_i \leq recStart \;\&\; arr_i \geq recFinish \tag{5.32}$$

$$X_{ci} = 0 \qquad \forall c \in S \qquad \forall i \in F : dept_i \geq recStart \;\&\; arr_i \leq recFinish \tag{5.33}$$

$$X_{ci} \in \{0,1\} \qquad \forall c \in S \qquad \forall i \in F \tag{5.34}$$

$$Y_i \in \{0,1\} \qquad \forall i \in F \tag{5.35}$$

$$Z_i \in \{0,1\} \qquad \forall i \in F \tag{5.36}$$

The notation used in the problem and the definition of the constraints have been explained in Section 5.2.2 on pages 146 - 148.

7.5 Integer Programming Model for Aircraft Recovery Problem (Step 6.2)

As in the integer programming model of crew recovery problem in Step 3.2, this integer program selects the best set of aircraft routes satisfying certain constraints.

After the completion of aircraft route generation during recovery period in Step 2.2 and computation of available crews for each flight leg in the recovery period in Step 4, the computed number of available crews is sent to this integer programming

model in step 5. And then, OPL script calls the integer programming model. This model selects the optimum set of routes while:

- minimizing the changes in original schedule (minimizing the deviation between original schedule and recovered schedule)
- minimizing the total cancellation costs
- minimizing the total delay costs
- minimizing the total aircraft swapping costs
- satisfying that before and after recovery periods aircrafts will return their original schedule

The integer programming models for aircraft recovery problem, used in the Solution algorithm S2 is the same as the integer programming model used in the aircraft recovery model A4, which has been explained in Section 4.4.2 on pages 123-126. In this section, for the sake of convenience, we are presenting the mathematical formulation of the model again:

$$Minimize \quad \sum_{k\in R}\sum_{i\in F_r^k} C_{ki}X_{ki} + \sum_{k\in K}\sum_{i\in F_r^k} DC_i X_{ki} + \sum_{k\in R}\sum_{i\in F} W[X_{ki} - A_{ki}] + \sum_{i\in F} D_i Y_i \tag{4.37}$$

Subject to:

$$\sum_{k\in K} X_{ki} + Y_i = 1 \qquad \forall i \in F \tag{4.38}$$

$$\sum_{i\in F_r^k} X_{ki} = 1 \qquad \forall k \in R \tag{4.39}$$

$$\sum_{i\in F_r^k} X_{ki} \leq AC_{ki} \qquad \forall k \in R \tag{4.40}$$

$$X_{ki} = A_{ki} \qquad \forall k \in K - \{R\} \qquad \forall i \in F : dept_i \leq recStart \,\&\, arr_i \geq recFinish \tag{4.41}$$

$$X_{ki} = 0 \qquad \forall k \in R \qquad \forall i \in F : dept_i \geq recStart \,\&\, arr_i \leq recFinish \tag{4.42}$$

$$X_{ik} \in \{0,1\} \qquad \forall k \in K \qquad \forall i \in F \tag{4.43}$$

$$Y_i \in \{0,1\} \qquad \forall i \in F \tag{4.44}$$

The notation used in the problem and the definition of the constraints have been explained in Section 4.4.2 on pages 123 - 126.

7.6 Numerical Example for Solution Algorithm S2

The proposed solution approach which includes both constraint programming and integer programming in the solution procedure is tested for different recovery scenarios using different data sets. For this purpose, the data obtained from 3 different airlines operating daily domestic flights in Turkey is used:

1. OnurAir Airlines' daily domestic flights (see Table A.1 for fleet information of OnurAir *&* Table A.3. for initial schedule of OnurAir)

2. AtlasJet Airlines' daily flights (see Table A.5 for fleet information of AtlasJet & Table A.6 for initial schedule of AtlasJet)

3. Pegasus Airlines' daily domestic flights (see Table A.7 for fleet information of Pegasus Airlines & Table A.8 for initial schedule of Pegasus Airlines)

The solution algorithm S2 is applied to real data obtained from the above mentioned 3 different airlines' daily domestic flight schedule. For each airline, only one example case is presented in this section:

The first example case is solved with OnurAir data. There occurs a shortage on crew “C03” between 15:00 and 20:00 and on aircraft “MD88a” between 06:40 and 12:00. During the recovery period crew “C03” can not fly the flights OHY016 and OHY017. The aircraft “MD88a” can not fly the flights OHY034 and OHY035. The solution obtained for OnurAir using solution algorithm S2 can be seen in Table 7.1. In the recovered schedule, no flight legs have to be cancelled or delayed. The shortage of aircraft MD88a is covered with another aircraft MD88c and the shortage crew C03 is substituted by a reserve crew.

The second example case is solved with AtlasJet airline’s data. There occurs a shortage on crew “C01” between 12:00 and 16:00 and on aircraft “B737c200b” between 07:00 and 11:00. During the recovery period crew “C01” can not fly the flights KK1011 and KK1012. The aircraft “B737c200b” can not fly the flights KK20 and KK21. The solution obtained for AtlasJet using solution algorithm S2 can be seen in Table 7.2. In the recovered solution, the flight legs KK20 and KK21 are covered by aircraft A321c204a, just swapping the aircrafts among the flight legs and the flight legs KK1011 and KK1012 are assigned to the reserved/standby crew.

The third example case is solved with Pegasus Airline’s data. There occurs a shortage on crew “C01” at 13:00 and on aircraft “B800a12” at 18:00. During the recovery period crew “C01” can not fly the flights PC108 and PC109, the aircraft “B800a12” can not fly the flights PC112 and PC113. The solution obtained for Pegasus Airlines using solution algorithm S2 can be seen in Table 7.3. In the recovered solution, the flight legs PC112 and PC113 are covered by aircraft B500a2, just swapping the aircrafts among the flight legs and the flight legs PC108 and PC109 are assigned to the reserved/standby crew.

Table 7.1 Recovered schedules obtained from the solution of Solution Algorithm S2 for OnurAir

Original Schedule							AIRCRAFT RECOVERY Shortage occurs on MD88a at 06:40		CREW RECOVERY Shortage occurs on Crew C03 at 15:00	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Crew	Aircraft	Recovery strategy	Crew	Recovery strategy
OHY010	IST	ADA	6:45	8:15	MD88d	C12	MD88d	-	C12	-
OHY011	ADA	IST	9:10	10:40	MD88d	C12	MD88d	-	C12	-
OHY016	IST	ADA	15:00	16:30	MD88a	C03	MD88a	-	Reserve	R
OHY017	ADA	IST	17:30	19:00	MD88a	C03	MD88a	-	Reserve	R
OHY018	IST	ADA	19:00	20:30	MD83c	C04	MD83c	-	C04	-
OHY019	ADA	IST	21:30	23:00	MD83c	C04	MD83c	-	C04	-
OHY022	IST	AYT	8:15	9:15	MD88b	C02	MD88b	-	C02	-
OHY023	AYT	IST	10:45	11:45	MD88b	C02	MD88b	-	C02	-
OHY024	IST	AYT	16:45	17:45	MD83b	C07	MD83b	-	C07	-
OHY025	AYT	IST	18:45	19:45	MD83b	C06	MD83b	-	C06	-
OHY026	IST	AYT	20:30	21:30	MD88e	C06	MD88e	-	C06	-
OHY027	AYT	IST	7:30	8:30	MD88e	C07	MD88e	-	C07	-
OHY028	IST	BJV	11:55	12:55	MD88d	C05	MD88d	-	C05	-
OHY029	BJV	IST	13:45	14:45	MD88d	C05	MD88d	-	C05	-
OHY034	IST	DIY	6:45	8:30	MD88a	C11	MD88c	S	C11	-
OHY035	DIY	IST	9:30	11:15	MD88a	C11	MD88c	S	C11	-
OHY036	IST	DIY	18:45	20:30	MD83d	C02	MD83d	-	C02	-
OHY037	DIY	IST	21:30	23:15	MD83d	C02	MD83d	-	C02	-
OHY038	IST	ERZ	12:30	14:20	MD88c	C08	MD88c	-	C08	-
OHY039	ERZ	IST	15:20	17:10	MD88c	C08	MD88c	-	C08	-
OHY042	IST	GZT	9:30	11:15	MD83b	C07	MD83b	-	C07	-
OHY043	GZT	IST	12:15	14:00	MD83b	C07	MD83b	-	C07	-
OHY050	IST	ADB	7:45	8:45	MD83d	C08	MD83d	-	C08	-
OHY051	ADB	IST	10:50	11:45	MD83d	C08	MD83d	-	C08	-
OHY054	IST	ADB	16:30	17:30	MD88b	C01	MD88b	-	C01	-
OHY055	ADB	IST	18:50	19:45	MD88b	C09	MD88b	-	C09	-
OHY058	IST	ADB	20:45	21:45	MD83b	C09	MD83b	-	C09	-
OHY059	ADB	IST	7:30	8:25	MD83b	C01	MD83b	-	C01	-
OHY062	IST	MLX	12:15	13:45	MD83a	C12	MD83a	-	C12	-
OHY063	MLX	IST	14:30	16:00	MD83a	C12	MD83a	-	C12	-
OHY066	IST	KSY	7:15	8:30	MD83a	C03	MD83a	-	C03	-
OHY067	KSY	IST	9:15	10:30	MD83a	C03	MD83a	-	C03	-
OHY074	IST	SZF	9:30	10:45	MD88e	C01	MD88e	-	C01	-
OHY075	SZF	IST	11:45	13:00	MD88e	C01	MD88e	-	C01	-
OHY080	IST	TZX	6:50	8:25	MD83c	C05	MD83c	-	C05	-
OHY081	TZX	IST	9:20	10:55	MD83c	C05	MD83c	-	C05	-
OHY086	IST	TZX	18:45	20:20	MD88c	C10	MD88c	-	C10	-
OHY087	TZX	IST	21:15	22:50	MD88c	C10	MD88c	-	C10	-

* Recovery Strategies
R : using reserve/standby crew, S : swapping crews/aircrafts, D : delaying flights , C : cancelling flights

Table 7.2 Recovered schedules obtained from the solution of Solution Algorithm S2 for AtlasJet

Original Schedule							AIRCRAFT RECOVERY Shortage occurs on Aircraft B737c200b at 07:00		CREW RECOVERY Shortage occurs on Crew C01 at 07:00	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Crew	Aircraft	Recovery strategy	Crew	Recovery strategy
KK10	IST	AYT	07:30	08:35	A321c199a	C07	A321c199a	--	C07	--
KK11	AYT	IST	09:30	10:35	A321c199a	C07	A321c199a	--	C07	--
KK14	IST	AYT	15:00	16:05	A321c210a	C04	A321c210a	--	C04	--
KK15	AYT	IST	17:00	18:05	A321c210a	C04	A321c210a	--	C04	--
KK18	IST	AYT	19:00	20:05	A321c210a	C04	A321c210a	--	C04	--
KK19	AYT	IST	21:00	22:10	A321c210a	C04	A321c210a	--	C04	--
KK20	IST	ADB	07:00	07:55	B737c200b	C10	A321c204a	S	C10	--
KK21	ADB	IST	08:45	09:40	B737c200b	C10	A321c204a	S	C10	--
KK24	IST	ADB	16:00	16:55	B737c200a	C06	B737c200a	--	C06	--
KK25	ADB	IST	18:00	18:55	B737c200a	C06	B737c200a	--	C06	--
KK28	IST	ADB	11:30	12:25	A321c199a	C07	A321c199a	--	C07	--
KK29	ADB	IST	14:00	14:55	A321c199a	C07	A321c199a	--	C07	--
KK40	IST	BJV	07:00	08:05	A321c204b	C02	A321c204b	--	C02	--
KK41	BJV	IST	09:00	10:05	A321c204b	C02	A321c204b	--	C02	--
KK42	IST	BJV	11:30	12:35	B737c200b	C11	B737c200b	--	C11	--
KK43	BJV	IST	13:50	14:55	B737c200b	C11	B737c200b	--	C11	--
KK44	IST	BJV	18:00	19:05	A321c204a	C09	A321c204a	--	C09	--
KK45	BJV	IST	20:00	21:05	A321c204a	C09	A321c204a	--	C09	--
KK1011	ECN	IST	07:00	08:30	A321c210c	C01	A321c210c	--	Reserve	R
KK1012	IST	ECN	09:30	11:00	A321c210c	C01	A321c210c	--	Reserve	R
KK1014	IST	ECN	16:00	17:30	A321c210b	C03	A321c210b	--	C03	--
KK1015	ECN	IST	19:00	20:30	B737c210a	C05	B737c210a	--	C05	--
KK1016	IST	ECN	21:30	23:00	B737c210a	C05	B737c210a	--	C05	--
KK1017	ECN	IST	21:30	23:00	A321c210b	C03	A321c210b	--	C03	--
KK1030	ADB	ECN	17:00	18:15	B737c210a	C05	B737c210a	--	C05	--
KK1031	ECN	ADB	15:00	16:15	B737c210a	C05	B737c210a	--	C05	--
KK1053	ECN	ADA	12:00	12:45	A321c210c	C01	A321c210c	--	C01	--
KK1054	ADA	ECN	13:30	14:15	A321c210c	C01	A321c210c	--	C01	--
KK1055	ECN	ADA	18:30	19:15	A321c210b	C03	A321c210b	--	C03	--
KK1056	ADA	ECN	20:00	20:45	A321c210b	C03	A321c210b	--	C03	--
KK4012	IST	AYT	11:00	12:05	A321c204b	C02	A321c204b	--	C02	--
KK4013	AYT	IST	13:00	14:05	A321c204b	C02	A321c204b	--	C02	--
KK4022	IST	ADB	19:45	20:40	B737c200a	C06	B737c200a	--	C06	--
KK4023	ADB	IST	21:30	22:25	B737c200a	C06	B737c200a	--	C06	--
KK6200	IST	EBL	10:30	12:45	A321c204a	C12	A321c204a	--	C12	--
KK6201	EBL	IST	14:00	16:30	A321c204a	C12	A321c204a	--	C12	--
KK625	IST	PRN	09:50	10:10	A321c210a	C08	A321c210a	--	C08	--
KK626	PRN	IST	11:30	14:00	A321c210a	C08	A321c210a	--	C08	--

* Recovery Strategies

R : using reserve/standby crew, S : swapping crews/aircrafts, D : delaying flights , C : cancelling flights

Table 7.3 Recovered schedules obtained from the solution of Solution Algorithm S2 for Pegasus

Original Schedule							AIRCRAFT RECOVERY Shortage occurs on Aircraft B800a12 at 18:00		CREW RECOVERY Shortage occurs on Crew C01 at 13:00	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Crew	Aircraft	Recovery strategy	Crew	Recovery strategy
PC100	SAW	ESB	8:30	9:25	A319b2	C16	A319b2	--	C16	--
PC101	ESB	SAW	9:50	10:45	A319b2	C16	A319b2	--	C16	--
PC102	SAW	ESB	6:50	7:45	B800a3	C23	B800a3	--	C23	--
PC103	ESB	SAW	8:10	9:05	B800a3	C23	B800a3	--	C23	--
PC104	SAW	ESB	9:50	10:45	B800a1	C07	B800a1	--	C07	--
PC105	ESB	SAW	11:10	12:05	B800a1	C07	B800a1	--	C07	--
PC106	SAW	ESB	17:10	18:05	B800a6	C37	B800a6	--	C37	--
PC107	ESB	SAW	18:30	19:25	B800a6	C37	B800a6	--	C37	--
PC108	SAW	ESB	13:15	14:10	B800b1	C01	B800b1	--	Reserve	R
PC109	ESB	SAW	14:35	15:30	B800b1	C01	B800b1	--	Reserve	R
PC110	SAW	ESB	19:55	20:50	A319b2	C35	A319b2	--	C35	--
PC111	ESB	SAW	21:15	22:10	A319b2	C35	A319b2	--	C35	--
PC112	SAW	ESB	18:50	19:45	B800a12	C02	B500a2	S	C02	--
PC113	ESB	SAW	20:10	21:05	B800a12	C02	B500a2	S	C02	--
PC114	SAW	ADB	6:40	7:45	B400a2	C21	B400a2	--	C21	--
PC115	ADB	SAW	8:10	9:10	B400a2	C21	B400a2	--	C21	--
PC116	SAW	ADB	9:55	10:55	B800a19	C09	B800a19	--	C09	--
PC117	ADB	SAW	11:20	12:20	B800a19	C09	B800a19	--	C09	--
PC118	SAW	ADB	12:50	13:50	B800a4	C24	B800a4	--	C24	--
PC119	ADB	SAW	14:15	15:15	B800a4	C24	B800a4	--	C24	--
PC120	SAW	ADB	18:30	19:30	B800a11	C29	B800a11	--	C29	--
PC121	ADB	SAW	19:55	20:55	B500a1	C29	B500a1	--	C29	--
PC122	SAW	ADB	19:55	21:00	B800a15	C18	B800a15	--	C18	--
PC123	ADB	SAW	21:25	22:25	B800a15	C18	B800a15	--	C18	--
PC124	SAW	ADA	6:05	7:25	B800a17	C29	B800a17	--	C29	--
PC125	ADA	SAW	7:50	9:15	B800a17	C29	B800a17	--	C29	--
PC126	SAW	ADA	20:35	21:55	B800a25	C34	B800a25	--	C34	--
PC127	ADA	SAW	22:20	23:45	B800a25	C34	B800a25	--	C34	--
PC128	SAW	ADA	15:40	17:00	B800a16	C42	B800a16	--	C42	--
PC129	ADA	SAW	17:25	18:50	B800a16	C42	B800a16	--	C42	--
PC130	SAW	SZF	20:10	21:30	B800a23	C42	B800a23	--	C42	--
PC131	SZF	SAW	21:55	23:15	B800a23	C42	B800a23	--	C42	--
PC132	SAW	TZX	15:30	17:10	B800a18	C05	B800a18	--	C05	--
PC133	TZX	SAW	17:35	19:20	B800a18	C05	B800a18	--	C05	--
PC134	SAW	TZX	8:20	10:00	B800a13	C15	B800a13	--	C15	--
PC135	TZX	SAW	10:25	12:10	A319b1	C15	A319b1	--	C15	--
PC136	SAW	TZX	22:45	0:25	A319b1	C18	A319b1	--	C18	--
PC137	TZX	SAW	6:00	7:45	B800a7	C19	B800a7	--	C19	--
PC138	SAW	TZX	20:25	22:05	B800a3	C05	B800a3	--	C05	--
PC139	TZX	SAW	22:30	0:15	B800a3	C05	B800a3	--	C05	--
PC140	SAW	AYT	6:30	7:40	B800a15	C26	B800a15	--	C26	--
PC141	AYT	SAW	8:05	9:15	B800a15	C26	B800a15	--	C26	--
PC142	SAW	AYT	17:05	18:15	B400a1	C25	B400a1	--	C25	--
PC143	AYT	SAW	18:40	19:50	B400a1	C25	B400a1	--	C25	--
PC144	SAW	AYT	12:15	13:25	A319b3	C23	A319b3	--	C23	--
PC145	AYT	SAW	13:50	15:00	A320b2	C23	A320b2	--	C23	--
PC146	SAW	AYT	20:05	21:15	B800a20	C11	B800a20	--	C11	--
PC147	AYT	SAW	21:40	22:50	B800a14	C11	B800a14	--	C11	--
PC148	SAW	AYT	8:00	9:10	B500a2	C17	B500a2	--	C17	--
PC149	AYT	SAW	9:35	10:45	A319b3	C17	A319b3	--	C17	--
PC152	SAW	SZF	6:10	7:30	B800a5	C01	B800a5	--	C01	--
PC153	SZF	SAW	7:55	9:15	B800a5	C01	B800a5	--	C01	--
PC156	SAW	GZT	5:45	7:15	B800a1	C07	B800a1	--	C07	--
PC157	GZT	SAW	7:40	9:15	B800a1	C07	B800a1	--	C07	--
PC158	SAW	GZT	20:15	21:45	B800a22	C37	B800a22	--	C37	--
PC159	GZT	SAW	22:30	0:00	B800a22	C37	B800a22	--	C37	--

* Recovery Strategies
R : using reserve/standby crew, S : swapping crews/aircrafts, D : delaying flights , C : cancelling flights

Table 7.3 Recovered schedules obtained from the solution of Solution Algorithm S2 for Pegasus (cont)

Original Schedule							AIRCRAFT RECOVERY Shortage occurs on Aircraft B800a12 at 18:00		CREW RECOVERY Shortage occurs on Crew C01 at 13:00	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Crew	Aircraft	Recovery strategy	Crew	Recovery strategy
PC162	SAW	ASR	6:20	7:35	B800b1	C09	B800b1	--	C09	--
PC163	ASR	SAW	8:00	9:15	B800b1	C09	B800b1	--	C09	--
PC164	SAW	ASR	20:10	21:25	A320b1	C16	A320b1	--	C16	--
PC165	ASR	SAW	21:50	23:05	A320b1	C16	A320b1	--	C16	--
PC166	SAW	ASR	8:40	10:00	B800a7	C30	B800a7	--	C30	--
PC167	ASR	SAW	10:25	11:45	B800a7	C30	B800a7	--	C30	--
PC168	SAW	MLX	18:30	20:00	B800a5	C27	B800a5	--	C27	--
PC169	MLX	SAW	20:45	22:20	B800a5	C27	B800a5	--	C27	--
PC172	SAW	BJV	6:35	7:45	B500a1	C24	B500a1	--	C24	--
PC173	BJV	SAW	8:10	9:20	B500a1	C24	B500a1	--	C24	--
PC174	SAW	BJV	18:35	19:45	B800a8	C36	B800a8	--	C36	--
PC175	BJV	SAW	20:10	21:20	B800a8	C36	B800a8	--	C36	--
PC176	SAW	BJV	20:00	21:10	B800a16	C17	B800a16	--	C17	--
PC177	BJV	SAW	21:35	22:45	B800a16	C17	B800a16	--	C17	--
PC178	SAW	DLM	6:00	7:15	B800a4	C13	B800a4	--	C13	--
PC179	DLM	SAW	8:10	9:20	B800a4	C13	B800a4	--	C13	--
PC180	SAW	HTY	20:20	21:55	B800a18	C25	B800a18	--	C25	--
PC181	HTY	SAW	22:40	0:15	B800a18	C25	B800a18	--	C25	--
PC182	SAW	KYA	20:10	21:20	B800a6	C10	B800a6	--	C10	--
PC183	KYA	SAW	21:40	22:50	B800a6	C10	B800a6	--	C10	--
PC188	SAW	KYA	6:00	7:15	B800a19	C06	B800a19	--	C06	--
PC189	KYA	SAW	8:10	9:20	B800a19	C06	B800a19	--	C06	--
PC190	SAW	VAN	10:00	12:05	B800a3	C13	B800a3	--	C13	--
PC191	VAN	SAW	12:35	14:45	B800a3	C13	B800a3	--	C13	--
PC194	SAW	EZS	11:50	13:25	B800a11	C06	B800a11	--	C06	--
PC195	EZS	SAW	13:45	15:25	B800a11	C06	B800a11	--	C06	--
PC196	SAW	DIY	12:40	14:20	B800a19	C03	B800a19	--	C03	--
PC197	DIY	SAW	15:15	17:00	B800a19	C03	B800a19	--	C03	--
PC202	AYT	DIY	23:00	0:35	B800a20	C14	B800a20	--	C14	--
PC203	DIY	AYT	1:30	3:00	B800a20	C14	B800a20	--	C14	--
PC2130	ADB	ESB	7:00	8:10	B800a9	C08	B800a9	--	C08	--
PC2131	ESB	ADB	8:35	9:50	B800a9	C08	B800a9	--	C08	--
PC2132	ADB	ESB	14:55	16:05	B800a2	C22	B800a2	--	C22	--
PC2133	ESB	ADB	16:30	17:45	B800a10	C38	B800a10	--	C38	--
PC2136	ADB	ESB	18:10	19:20	B800a10	C38	B800a10	--	C38	--
PC2137	ESB	ADB	19:45	21:00	B800a2	C22	B800a2	--	C22	--
PC2182	ADB	EZS	21:30	23:15	B800a2	C22	B800a2	--	C22	--
PC2183	EZS	ADB	23:45	1:45	B800a2	C22	B800a2	--	C22	--
PC2194	ADB	MQM	10:00	11:50	B800a2	C33	B800a2	--	C33	--
PC2195	MQM	ADB	12:40	14:35	B800a2	C33	B800a2	--	C33	--
PC2402	ADA	AYT	8:25	9:25	A320b2	C41	A320b2	--	C41	--
PC2403	AYT	ADA	9:55	10:55	B500a2	C41	B500a2	--	C41	--
PC2404	ADA	AYT	19:50	20:50	B800a14	C28	B800a14	--	C28	--
PC2405	AYT	ADA	21:15	22:15	A320b2	C28	A320b2	--	C28	--
PC2418	ADB	ASR	15:20	16:45	B800a21	C40	B800a21	--	C40	--
PC2419	ASR	ADB	17:30	19:05	B800a21	C40	B800a21	--	C40	--
PC2426	TZX	ADA	13:50	15:15	B500a1	C41	B500a1	--	C41	--
PC2427	ADA	TZX	12:00	13:25	B500a2	C41	B500a2	--	C41	--
PC2466	ESB	BJV	19:45	21:00	B800a10	C38	B800a10	--	C38	--
PC2467	BJV	ESB	21:20	22:30	B800a10	C38	B800a10	--	C38	--
PC250	SAW	VAS	6:20	7:35	B800a23	C04	B800a23	--	C04	--
PC251	VAS	SAW	8:30	9:50	B800a23	C04	B800a23	--	C04	--
PC2810	ADB	IST	7:15	8:15	B800a21	C20	B800a21	--	C20	--
PC2811	IST	ADB	9:10	10:10	B800a21	C20	B800a21	--	C20	--
PC2812	ADB	IST	10:40	11:40	B800a9	C20	B800a9	--	C20	--
PC2813	IST	ADB	12:10	13:15	B800a9	C20	B800a9	--	C20	--

* Recovery Strategies
R : using reserve/standby crew, S : swapping crews/aircrafts, D : delaying flights , C : cancelling flights

Table 7.3 Recovered schedules obtained from the solution of Solution Algorithm S2 for Pegasus (cont.)

Original Schedule							AIRCRAFT RECOVERY Shortage occurs on Aircraft B800a12 at 18:00		CREW RECOVERY Shortage occurs on Crew C01 at 13:00	
Flight No	Origin	Destination	Departure	Arrival	Aircraft	Crew	Aircraft	Recovery strategy	Crew	Recovery strategy
PC2814	ADB	IST	14:55	15:55	B800a9	C08	B800a9	--	C08	--
PC2815	IST	ADB	16:40	17:40	B800a9	C08	B800a9	--	C08	--
PC2816	ADB	IST	18:15	19:15	B800a24	C12	B800a24	--	C12	--
PC2817	IST	ADB	20:00	21:00	B800a24	C12	B800a24	--	C12	--
PC2818	ADB	IST	21:40	22:40	B800a24	C40	B800a24	--	C40	--
PC2819	IST	ADB	0:15	1:20	B800a24	C40	B800a24	--	C40	--
PC2892	ADB	ADA	6:35	8:00	A320b2	C12	A320b2	--	C12	--
PC2893	ADA	ADB	15:45	17:15	B500a1	C12	B500a1	--	C12	--
PC2894	ADB	ADA	17:55	19:20	B800a14	C28	B800a14	--	C28	--
PC2899	ADA	ADB	22:45	0:15	A320b2	C28	A320b2	--	C28	--
PC4114	SAW	ADB	21:35	22:40	B800a12	C02	B800a12	--	C02	--
PC4115	ADB	SAW	23:05	0:05	B800a12	C02	B800a12	--	C02	--
PC4119	ADB	SAW	6:45	7:45	B800a11	C33	B800a11	--	C33	--
PC4120	SAW	ADB	8:15	9:15	A320b1	C33	A320b1	--	C33	--
PC4122	SAW	ADB	16:00	17:05	B800a14	C10	B800a14	--	C10	--
PC4123	ADB	SAW	17:30	18:30	A320b1	C10	A320b1	--	C10	--
PC4126	SAW	ADA	8:15	9:35	B500a3	C31	B500a3	--	C31	--
PC4127	ADA	SAW	9:55	11:20	B500a3	C31	B500a3	--	C31	--
PC4130	SAW	SZF	8:10	9:30	B800a11	C19	B800a11	--	C19	--
PC4131	SZF	SAW	9:55	11:15	B800a11	C19	B800a11	--	C19	--
PC4136	SAW	TZX	12:35	14:15	B800a7	C19	B800a7	--	C19	--
PC4137	TZX	SAW	14:40	16:25	B500a2	C18	B500a2	--	C18	--
PC4138	SAW	TZX	10:30	12:10	B500a1	C04	B500a1	--	C04	--
PC4139	TZX	SAW	12:40	14:25	B800a13	C04	B800a13	--	C04	--
PC4140	SAW	AYT	15:20	16:30	A320b2	C15	A320b2	--	C15	--
PC4141	AYT	SAW	16:55	18:05	B800a20	C27	B800a20	--	C27	--
PC4146	SAW	AYT	22:40	23:50	A319b3	C27	A319b3	--	C27	--
PC4147	AYT	SAW	21:00	22:10	A319b3	C15	A319b3	--	C15	--
PC4152	SAW	SZF	15:15	16:35	B800a13	C39	B800a13	--	C39	--
PC4153	SZF	SAW	17:00	18:20	B800a13	C39	B800a13	--	C39	--
PC4158	SAW	GZT	7:45	9:15	B800a25	C35	B800a25	--	C35	--
PC4159	GZT	SAW	9:55	11:30	B800a25	C35	B800a25	--	C35	--
PC4162	SAW	ASR	18:45	20:00	B400a2	C21	B400a2	--	C21	--
PC4163	ASR	SAW	20:20	21:35	B400a2	C21	B400a2	--	C21	--
PC4170	SAW	BJV	16:00	17:10	B800a22	C11	B800a22	--	C11	--
PC4171	BJV	SAW	17:30	18:40	B800a22	C11	B800a22	--	C11	--
PC4172	SAW	BJV	21:45	22:55	B800a8	C36	B800a8	--	C36	--
PC4173	BJV	SAW	23:15	0:25	B800a8	C36	B800a8	--	C36	--
PC4174	SAW	BJV	12:00	13:10	A319b2	C26	A319b2	--	C26	--
PC4175	BJV	SAW	13:30	14:40	A319b2	C26	A319b2	--	C26	--
PC4178	SAW	DLM	20:05	21:20	B800a17	C32	B800a17	--	C32	--
PC4179	DLM	SAW	21:45	22:55	B800a17	C32	B800a17	--	C32	--
PC4180	SAW	HTY	5:40	7:15	B800a6	C03	B800a6	--	C03	--
PC4181	HTY	SAW	8:00	9:35	B800a6	C03	B800a6	--	C03	--
PC4182	SAW	DLM	18:30	19:45	B500a3	C30	B500a3	--	C30	--
PC4183	DLM	SAW	20:45	21:55	B500a3	C30	B500a3	--	C30	--
PC4188	SAW	KYA	18:55	20:05	A319b1	C39	A319b1	--	C39	--
PC4189	KYA	SAW	20:45	21:55	A319b1	C39	A319b1	--	C39	--
PC4194	SAW	EZS	14:00	15:35	A319b1	C32	A319b1	--	C32	--
PC4195	EZS	SAW	16:30	18:10	A319b1	C32	A319b1	--	C32	--
PC4477	DLM	SAW	22:00	23:10	B400a1	C31	B400a1	--	C31	--
PC4478	SAW	DLM	20:15	21:30	B400a1	C31	B400a1	--	C31	--

* Recovery Strategies

R : using reserve/standby crew, S : swapping crews/aircrafts, D : delaying flights , C : cancelling flights

7.7 Evaluation of the Solution Algorithm S2

The Table 7.4 summarizes the results obtained from the selected examples solved with using the data obtained from 3 Turkish airline's daily flight schedule, in terms of computational times and the problem sizes. In this table, both aircraft and crew related problems' solution times and the integrated problem's overall solution times can be seen.

As it can be seen from Table 7.4 the proposed solution algorithm are generating real time solutions. However, solution times directly depend on the problem size. Thus, when the number of flight legs, number of crews and number of aircrafts are increased, the solution time is increased as well.

The number of feasible routes also depends on the problem size. When the number of aircrafts, number of crews and number of flight legs are increased, the number of possible feasible routes/ pairings increases as well. Route and pairing generation problem is solved using constraint programming. Figure 7.2 displays the interaction between the number of routes generated and the route generation time using constraint programming for the example cases solved with the data obtained from 3 Turkish airline's daily flight schedule.

We can conclude that the constraint programming can be efficiently used for this purpose, since the routes are generated in real time. However, when the number of routes increases, the solution time increases as well. For example, when the solution time for generating 5000 routes is 2 or 2.5 seconds, the solution time for generating almost 100,000 routes, is 2 -2.5 minutes.

Table 7.4 The solution times and the problem size obtained from Solution Algorithm S1

			Crew Recovery						Aircraft Recovery							Integrated
total #of flights	total #of aircraft	total #of crew	Recovery period	# of flight legs in recovery period	# of generated pairings	pairings generation time (in sec)	reschedule optimization time (in sec)	total solution time (in sec)	Recovery period	# of flight legs in recovery period	Shortage aircraft	# of generated routes	route generation time (in sec)	reschedule optimization time (in sec)	total solution time (in sec)	total solution time (in sec)
38	9	12	6:45 - 12:00	16	2196	2.2560	0.0780	2.3340	12:30 - 18:00	7	MD88c	2520	1.7140	0.2030	1.9170	4.2510
38	9	12	6:45 - 12:00	16	2196	2.3710	0.0940	2.4650	06:40 - 12:00	16	MD83a	3096	1.9180	0.2500	2.1680	4.6330
38	9	12	6:45 - 12:00	16	2196	2.3060	0.0940	2.4000	12:00 - 20:00	12	MD88a	3465	1.8720	0.2650	2.1370	4.5370
38	9	12	12:00 - 20:00	12	2544	2.2320	0.0930	2.3250	12:00 - 20:00	12	MD88a	3465	1.9190	0.2810	2.2000	4.5250
38	9	12	12:00 - 20:00	12	2544	2.0920	2.2790	4.3710	09:00 - 15:00	14	MD88e	3375	2.0730	2.6040	4.6770	9.0480
38	9	12	09:00 - 15:00	14	2316	2.4000	2.5880	4.9880	09:00 - 15:00	14	MD88a	3375	2.0740	2.5890	4.6630	9.6510
38	9	12	09:00 - 15:00	14	2316	2.2640	0.0940	2.3580	18:00 - 00:00	10	MD83d	2745	1.6220	0.2340	1.8560	4.2140
38	9	12	18:00 - 00:00	10	1860	2.0600	0.0780	2.1380	18:00 - 00:00	10	MD83d	2745	1.6690	0.2180	1.8870	4.0250
38	9	12	18:00 - 00:00	10	1860	2.0450	0.0780	2.1230	07:00 - 15:00	19	MD88b	4059	2.2470	0.2800	2.5270	4.6500
38	9	12	07:00 - 15:00	19	2844	2.5870	0.1250	2.7120	07:00 - 15:00	19	MD88b	4059	2.2290	0.3280	2.5570	5.2690
38	9	12	07:00 - 15:00	19	2844	2.5730	0.1250	2.6980	15:00 - 20:00	7	MD88a	2205	1.6840	0.2340	1.9180	4.6160
38	9	12	15:00 - 20:00	7	2028	2.0310	0.0780	2.1090	15:00 - 20:00	7	MD88a	2205	1.7020	0.1870	1.8890	3.9980
38	9	12	06:00 - 12:00	9	4428	2.8270	0.1560	2.9830	06:00 - 12:00	9	A321c199a	4365	2.2400	0.3740	2.6140	5.5970
38	9	12	06:00 - 12:00	9	4428	2.5480	0.1560	2.7040	06:00 - 10:00	5	B737c200b	3348	2.1540	0.3430	2.4970	5.2010
38	9	12	06:00 - 10:00	5	3252	2.6360	0.1250	2.7610	06:00 - 10:00	5	B737c200b	3348	2.2330	0.2970	2.5300	5.2910
38	9	12	06:00 - 10:00	5	3252	2.5760	0.1090	2.6850	12:00 - 17:00	11	A321c210c	3555	2.5110	0.2650	2.7760	5.4610
38	9	12	12:00 - 15:00	5	3108	2.6990	0.1250	2.8240	12:00 - 17:00	11	A321c210c	3555	2.5140	0.2800	2.7940	5.6180
38	9	12	12:00 - 15:00	5	3108	2.6950	2.9130	5.6080	16:00 - 22:00	12	A321c210b	4671	2.0870	4.2130	6.3000	11.9080
38	9	12	16:00 - 22:00	12	5376	3.0150	3.3890	6.4040	16:00 - 22:00	12	A321c210b	4671	2.8870	3.7600	6.6470	13.0510
38	9	12	16:00 - 22:00	12	5376	2.9930	0.2190	3.2120	11:00 - 14:30	7	A321c204b	3213	2.2120	0.3900	2.6020	5.8140
38	9	12	11:00 - 14:30	7	3888	2.7600	0.1400	2.9000	11:00 - 14:30	7	A321c204b	3213	2.1670	0.3120	2.4790	5.3790
38	9	12	11:00 - 14:30	7	3888	2.7260	0.1400	2.8660	19:30 - 22:30	5	B737c200a	3294	2.3090	0.3120	2.6210	5.4870
38	9	12	19:30 - 22:30	5	3228	2.5670	0.1250	2.6920	19:30 - 22:30	5	B737c200a	3294	2.1520	0.2650	2.4170	5.1090
164	36	42	13:00 - 15:30	15	32185	28.7390	31.3440	60.0830	18:40 - 21:10	21	B800a12	50160	64.4940	71.5450	136.0390	196.1220
164	36	42	18:40 - 21:10	21	38525	38.3560	41.4760	79.8320	18:40 - 21:10	21	B800a12	50160	64.4230	72.0360	136.4590	216.2910
164	36	42	18:40 - 21:10	21	38525	39.9230	1.5600	41.4830	08:30 - 10:00	12	B800a3	32952	35.8970	2.9480	38.8450	80.3280
164	36	42	08:30 - 10:00	12	26579	25.5860	1.0610	26.6470	08:30 - 10:00	12	B800a3	32952	37.9110	2.4180	40.3290	66.9760
164	36	42	08:30 - 10:00	12	26579	25.9570	28.1880	54.1450	06:30 - 09:30	32	B800a15	71388	119.9730	129.0520	249.0250	303.1700
164	36	42	06:30 - 09:30	32	53955	69.9690	74.4160	144.3850	06:30 - 09:30	32	B800a15	71388	118.6600	130.1570	248.8170	393.2020
164	36	42	06:30 - 09:30	32	53955	67.5220	72.0780	139.6000	20:00 - 00:00	36	B800a17	97608	146.2620	160.9410	307.2030	446.8030
164	36	42	20:00 - 00:00	36	69452	84.0580	90.9170	174.9750	20:00 - 00:00	36	B800a17	97608	144.7520	160.9910	305.7430	479.8180
164	36	42	20:00 - 00:00	36	69452	83.9400	2.9180	86.8580	08:00 - 10:00	26	B800a7	60408	98.1000	5.9590	104.0590	190.9170

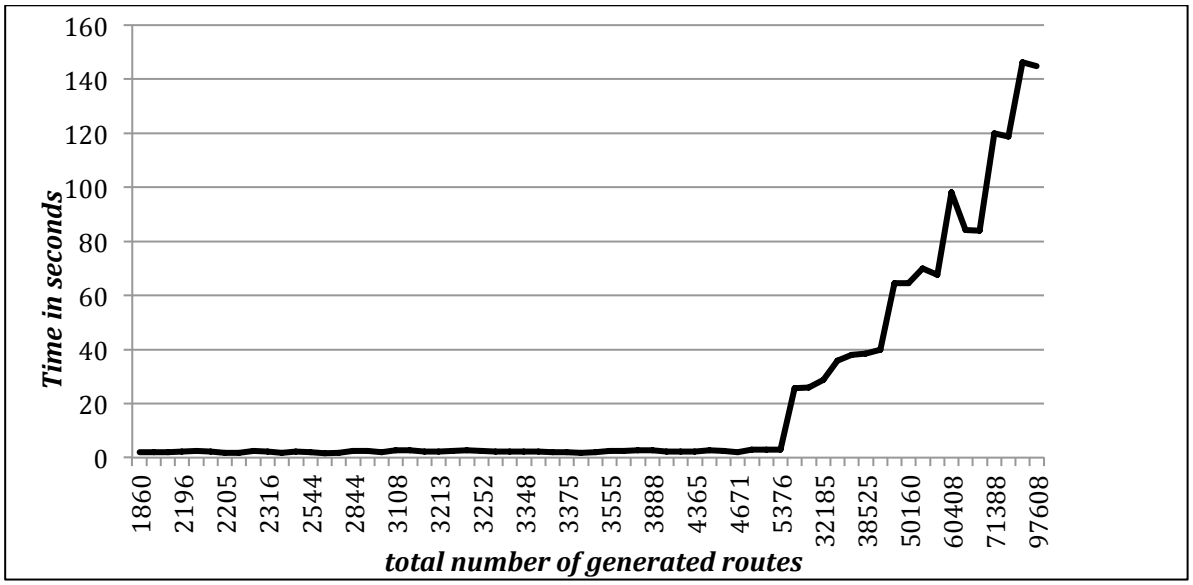

Figure 7.2 The relation between the number of generated routes and the route generation time.

As a summary, constraint programming models are capable of generating large number of routes and pairings, satisfying all the aircraft and crew related constraints, in almost no time.

- The average route generation time for relatively small/medium size problem (number of flight legs are 38) is 2.3 seconds.
- The average route generation time for larger problems (number of flights are 164) is 70 seconds.

Not only the route generation times, but also the overall problem solution times are satisfactory:

- The average solution time for relatively small/medium size problem (number of flight legs are 38) is 6 seconds.
- The average route generation time for larger problems (number of flights are 164) is 264 seconds (4,4 minutes).

As we can see from the solutions, the proposed solution algorithm perfectly effective when the number of flight legs is relatively small. However, for the airlines which operates large number of flight legs in a day, problem solution time increases.

The problem may be further improved in order to solve the larger problems in shorter time.

The advantage of using constraint programming technique in the integrated problem can be summarized as follows:

The problem related constraints, which makes the problem more complex are now satisfied in the constraint programming phase in generating crew pairings and aircraft routes.

Since problem related constraints are satisfied in the constraint programming part of the problem, the integer programming problem only selects the best set of routes, which makes the integer programming model less complex than the traditional network flow models. Since the complexity of the integer programming model is reduced, obtaining real time solutions become easier.

The advantages of the proposed iterative solution algorithm can be summarized as follows:

Both aircraft and crew availabilities are considered in the algorithm. Without integration, both aircraft and crew disruptions are recovered at one time.

Since the solution obtained for the aircraft and crew recovery problems without integration, the problem complexity and huge number of constraints and variables are prevented. Thus, the overall problem is less complex than the integrated problem in terms of modeling and the real time solutions can be obtained in easier and more practical way.

Several combinations of aircraft and crew availabilities can be considered. The algorithm can find solutions for multiple dependent or independent crew and aircraft unavailabilities: In some cases only aircraft disruptions can be recovered. In some cases aircraft and crew disruptions for same flight leg can be considered or aircraft unavailability for flight a and crew unavailability for flight b can be solved.

CHAPTER EIGHT
CONCLUSION

8.1 Summary and Conclusion

The primary objective of this research was to develop efficient models and solution algorithms for both aircraft and crew recovery problems, which are very highly challenging in terms of modeling techniques, complex constraints and problem size. Then we realized that these problems need to be solved by connecting the interacting constraints. Therefore, we also felt the need to focus on this objective. Within these purposes, the aircraft and crew recovery problems have been considered both individually and integrated in this dissertation.

The planning and scheduling problems in the airline industry have always taken significant attention, since operating airlines and their resources are very costly. Therefore, the researchers have been developing several operations research techniques and optimization tools which can be applied to airlines' planning and operations problems since 1950s. The airlines are now capable of managing their planning process and schedules using such optimization tools. However, they still try to find a way to recover their planned schedules as soon as possible when they face with a disruption which happens very often.

Although there have been significant studies on the airline recovery problems during the last decades in operations research field, these problems are still challenging and subject to further improvements. Therefore, this dissertation has focused on developing efficient models and solution algorithms for the aircraft and crew recovery problems and for their integration, considering following challenges:

1. The aircraft and crew recovery problems have mostly considered individually in the literature. The developed solution algorithms for the airline recovery problems generally involve only one resource's shortage, either aircraft or crew. However, the crew related and aircraft related decisions affect each other. In order to operate any flight leg there must be at least one available

crew and one available aircraft to be assigned to the flight leg.

2. The integration of the airline resources causes complex recovery models and requires special attention to be solved to optimality. Because the integrated crew and aircraft recovery problems includes both crew related and aircraft related constraints and variables. Thus, the number of constraints and variables become very large compared to the individual recovery problems. Because of the increased complexity and number of variables, the integrated problem is NP-hard.

3. The airline recovery problems in the literature have been modeled as multi-commodity network flow problems in order to obtain feasible and optimum (or near-optimum) solutions. Therefore, the resulting mathematical programs are more or less identical. With the improvement of optimization technology and computer systems, new solution approaches may be developed in order to solve airline recovery problems in real-time with less effort using more computer based technology.

By taking into account above listed challenges, new models and solution algorithms have been developed for both aircraft and crew recovery problems and for their integration.

8.1.1 Summary of the Individual Aircraft and Crew Recovery Problems

First the aircraft and crew recovery problems have been considered individually and different models have been developed for them.

Four different models have been developed for the aircraft recovery problems, three of those have used multi-commodity network flow technique where the underlying network is connection network and the fourth one have used constraint programming involved solution approach, as we described in Chapter 4, on pages 94-132:

1. Model A1: Cancellation model for single aircraft unavailability using multi-commodity network flow problem where the underlying network is connection network
2. Model A2: Cancellation and delay model for multiple aircraft unavailabilities using multi-commodity network flow problem where the underlying network is connection network
3. Model A3: Cancellation and delay model for multiple aircraft unavailabilities including crew considerations using multi-commodity network flow problem where the underlying network is connection network
4. Model A4: Cancellation and delay model for multiple aircraft unavailabilities including crew considerations using constraint programming embedded solution approach

Two different models have been developed for the crew recovery problems, as can be found in Chapter 5, on pages 133-153:

1. Model C1: crew recovery model using multi-commodity network flow problem where the underlying network is the connection network.
2. Model C2: crew recovery model using constraint programming embedded solution approach.

For both aircraft and crew recovery problems, the developed models have been tested on the real data obtained from the daily domestic flight schedules of OnurAir.

The proposed models and obtained solutions for both aircraft and crew recovery problems have been evaluated in terms of problem modeling technique, the computational times and the number of impacted aircrafts/crews.

The solution times obtained from both multi-commodity flow models and constraint programming embedded approach are very satisfactory for medium sized airline (38 flight legs a day, among 12 stations). However, the solution times obtained from the multi-commodity flow models have not included the network development times, which are very time consuming and challenging.

The constraint programming embedded solution approach has included both routes/pairing generation times, which corresponds to the network development in the multi-commodity network models, and optimized schedule generation times.

The quality of the obtained solution has also been evaluated in terms of the number of impacted aircraft and crews. One of the objectives in the recovery problems is that finding a recovered schedule with minimum change in the original schedule. However, in the solutions obtained from multi-commodity network flow models, there have been several aircraft/crew swapping options, causing that the original schedules have been changed a lot. On the other hand, constraint programming embedded solution approach has been capable of generating optimum reschedules without changing the all schedules in the recovery period.

Constraint programming embedded approach is more efficient and stronger method compared to the traditional multi-commodity network flow models. The main reasons can be summarized as follows:

- While solving aircraft recovery problems modeled as multi-commodity network flow problem, network structure of the problem needs to be prepared manually and given as input data to the problem. This process is very time consuming and it is hard to adjust in case of flight schedule changes. However, constraint programming approach generate feasible routes, which correspond to network structure in the multi-commodity network flow problem, with reference to the given constraints without any manual intervention to the problem. The constraint programming generates large number of routes in very short time.

- The other strength of the constraint programming involved approach is that constraint programming part of the problem finds the feasible routes satisfying all the constraints. Since the problem related constraints are satisfied in the constraint programming part of the problem, the integer programming problem only selects the best set of routes, which makes the integer programming model less complex than the traditional network flow models. Since the complexity of

the integer programming model is reduced, obtaining real time solutions become easier. On the other hand, in the traditional multi-commodity network flow problems, all the complex constraints have to be modeled in the network flow model formulation. Because of the complex constraints, the problem does not always give the feasible solution. Therefore in order to obtain feasible solutions, special attention is required and constraint relaxation techniques should be used.

- Modeling constraint programming is less complex than developing network flow models. While in network flow problems, it is difficult to satisfy all constraints and to find feasible solutions without using complex solution techniques, the constraint programming approach can generate feasible routes with respect to all constraints and does not require complex solution techniques.

- The constraint programming approach is very adaptable in case of schedule changes. The developed constraint programming model is not airline dependent and it can be easily adapted to different airlines' schedule without much effort.

8.1.2 Summary of the Integrated Aircraft and Crew Recovery Problems

This dissertation has also considered the correlation and dependency of the aircraft and crew recovery problems. Two new solution algorithms have been presented which solve aircraft recovery and crew recovery problems sequentially and in iterative manner, using different modeling techniques. These solution algorithms have taken into account the dependency of two problems, representing the correlation between them without integrating the two recovery problems:

1. Solution Algorithm S1, which the detailed description can be found in Chapter 6, on pages 154-169, has used multi-commodity network flow technique in order to model aircraft and crew recovery problems. This solution algorithm has used following individual problems and formulated an algorithm, which relates the two problems using linking constraints and iteration.

- Multi-commodity network flow model for aircraft recovery problem (cancellation and delay model for multiple aircraft unavailabilities including crew considerations)
- Multi-commodity network flow model for crew recovery problem

2. Solution Algorithm S2, which the detailed description can be found in Chapter 7, on pages 170-195, has used constraint programming embedded solution approach in order to model aircraft and crew recovery problems. This solution algorithm has used following individual problems and formulated an algorithm, which relates the two problems using linking constraints and iteration.

   - Constraint programming model for generating feasible crew pairings in the recovery period for crew recovery problem
   - Constraint programming model for generating feasible aircraft routes in the recovery period for aircraft recovery problem
   - Integer programming model for obtaining best set of pairings for crew recovery problem, covering all the flight legs and minimizing the deviation from original schedule
   - Integer programming model for obtaining best set of routes for aircraft recovery problem, covering all the flight legs and minimizing the deviation from original schedule.

The both solution algorithms have been tested using the real data. The solution algorithm S1 has been tested using daily domestic schedule of OnurAir.

Since the solution algorithm S2 is the most improved solution approach in this dissertation, the efficiency and performance of this method has been tested better using the real data obtained from the three different airlines' daily domestic flight schedule.

The solution times obtained from the both solution approaches have been very satisfactory. Both solution algorithms generate aircraft and crew recovery solutions in real time especially for the medium sized airlines.

The solution algorithm S2, which uses constraint programming in order to generate feasible routes and pairings in the recovery period, perfectly effective when the number of flight legs is relatively small. However, for the airlines which operates large number of flight legs in a day, problem solution time increases. The problem may be further improved in order to solve the larger problems in shorter time.

The advantages of the both solution algorithms can be summarized as follows:

- The proposed solution algorithms find recovery solutions for both aircraft and crew disruptions at one time, connecting aircraft and crew recovery problems together via linking constraints without combining these problems' constraints into a huge model.

- Our way of integration is performed by connecting two problems via linking constraints rather than developing fully integrated problem, which includes both aircraft and crew related constraints into a huge model, as done in the classical way of integration in the literature. Therefore, the problem complexity and huge number of constraints and variables, coming from the classical way of integration, are prevented. Thus, the overall problem is less complex than the integrated problems developed in classical way, in terms of modeling and the real time solutions can be obtained in easier and more practical way.

- Several combinations of aircraft and crew availabilities can be considered. The algorithm can find solutions for multiple dependent or independent crew and aircraft unavailabilities: In some cases only aircraft disruptions can be recovered. In some cases aircraft and crew disruptions for same flight leg can be considered or aircraft unavailability for flight *a* and crew unavailability for flight *b* can be solved.

8.1.3 Contributions

This dissertation has developed several models and solution algorithms in order to deal with the challenges mentioned at the beginning of the section 8.1. Therefore, the contributions of the research performed in this dissertation can be summarized as follows:

1. A detailed literature review for aircraft and crew recovery problems has been provided:

   - By explaining the each work performed on airline recovery problems in detail, a time framework of the problem have been obtained and the impacts and challenges of the airline recovery problems have been discussed.

2. New solution algorithms for integrated aircraft and crew recovery problems have been developed using different techniques. Newly developed algorithms are capable of finding recovery solutions considering both aircraft and crew availabilities and their correlations, without integrating the aircraft and crew related constraints in one big problem:

   - As mentioned previously, the most of the studies in the literature focuses on aircraft recovery and crew recovery problems separately. While solving airline resource scheduling problems, the crew related and aircraft related decisions affect each other. In order to operate any flight leg there must be at least one available crew and one available aircraft to be assigned to the flight leg. By solving crew recovery and aircraft recovery problems iteratively, both aircraft assignment and crew reschedule are integrated and their availabilities are considered together.

   - Rather than fully integration, such iterative solution algorithms are more efficient. Because the integrated crew and aircraft recovery problems are complex in terms of modeling. The integrated problem includes both crew related and aircraft related constraints and variables. Thus, the number of

constraints and variables become very huge compared to the individual recovery problems. Because of the increased complexity and number of variables, the integrated problem is NP-hard. By solving crew recovery and aircraft recovery problems iteratively, the problem complexity and the solution time decreases.

3. Constraint programming embedded solution algorithm has been developed for aircraft recovery, crew recovery and also integrated recovery problems:

- In the literature most of the work on airline recovery problems has used multi-commodity network flow technique in order to model aircraft and crew recovery problems. Although this traditional technique is still effective, new solution approaches become available with the improvement of the optimization technology and computer systems. Being one of computer technology based technique constraint programming is widely applied to scheduling and planning problems in the literature. However, to best of our knowledge there has not been any published work on applying it to the airline recovery problems. Therefore, constraint programming involved solution approach has been applied to aircraft and crew recovery problems providing more flexible modeling opportunity with less effort and obtaining real-time solutions.

8.2 Directions for Future Research

The field of aircraft and crew recovery problems has increasing interest over the last decade and still very promising for further research opportunities.

In this dissertation, the interaction and dependency between aircraft and crew recovery problems have been considered. However, in airline industry there are not only aircraft and crews but also other resources, which are affected from the inconvenience resulting from the disruption. Another important resource for the airlines, after aircraft and crew, is the passenger. The passengers are affected form the disruptions as much as aircraft and crews. Furthermore, the airlines have to pay not

only the cancellation and delay costs but also the passenger hospitality costs, for the passengers affected by a disruption. Therefore while developing integrated recovery models, the passenger recovery should be considered as well. The proposed solution algorithms in this dissertation may be extended taking into account the recovering passengers with the objective of minimizing the passengers' inconvenience and the passenger delay costs.

In this dissertation while modeling aircraft, crew and integrated aircraft and crew problems, it was assumed that the maintenance service of the aircrafts is made after the midnight, when no flights are scheduled to be flown. However, in real world, this is not the case. The proposed models may be further amended satisfying also aircraft maintenance requirements.

Another interesting research topic closely related to disruption management is robust planning. The central idea in the creating robust schedules is to incorporate the possibility to absorb disruptions and remain feasible into the schedule, even if any disruption occurs. Advances in mathematical programming techniques and the speed of computers allowed the airlines to remove slacks in the aircraft and crew schedules in order to decrease their associated costs. However, this kind of schedules became more sensitive to irregularities and disruptions in the airline operations causing a gap between the actual schedules and planned schedules causing the great difference also in their corresponding costs. Developing new efficient models and solution algorithms for robust planning and scheduling problems, including computer based techniques which are available for recent decades with the improvement of the optimization technology and computer systems, will be an interesting research.

REFERENCES

Abara, J. (1989). Applying integer linear programming to the fleet assignment problem. *Interfaces, 19*(4), 20-28.

Abdelghany, K.F., Shah, S.S., Raina, S., & Abdelghany, A.F. (2004). A Model for projecting flight delays during irregular operation conditions. *Journal of Air Transport Management, 10*, 385-394.

Abdelghany, A.F., Ekollu, G., Narasimhan, R., & Abdelghany, K.F. (2004). A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research*, *127*, 309-331.

Abdelghany, K.F., Abdelghany, A.F., & Ekollu, G. (2008). An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, *185*, 825 - 848.

AhmadBeygi, S., Cohn, A., & Weir, M. (2009). An integer programming approach to generating airline crew pairings. *Computers and Operations Research, 36*(4), 1284 -1298.

Andersson, T., & Varbrand, P. (2004). The flight perturbation problem. *Transportation Planning and Technology, 27*(2), 91-118.

Arabeyre, J.P., Fearnley, J., Steiger, F.C., & Theather ,W.A. (1969). Airline crew scheduling problem: A Survey. *Transportation Science, 3*, 140-163.

Argüello, M.F., Bard, J.F., & Yu, G. (1997). A GRASP for aircraft routing in response to grounding and delays. *Journal of Combinatorial Optimization*, *5*, 211-228.

Barnhart, C., Boland, N.L., Clarke, L.W., Johnson, E.L, Nemhauser, G.L., & Shenoi, R.G. (1998). Flight string models for aircraft fleeting and routing. *Transportation Science, 32*(3), 208-220.

Barnhart, C., & Shenoi, R.G. (1998). An approximate model and solution approach for the long-haul crew pairing problem. *Transportation Science*, *32*(3), 221-231.

Barnhart, C., Kniker, T., & Lohatepanont, M. (2002). Itinerary-based airline fleet assignment. *Transportation Science*, *36*, 199-217.

Barnhart, C., Belobaba, P., & Odoni, A.R. (2003). Applications of operations research in the air transport industry. *Transportation Science, 37* (4), 368-391.

Barnhart, C., & Cohn, A. (2004). Airline schedule planning: Accomplishments and opportunities. *Informs, 6*(1), 3-22.

Bard, J.F., Yu, G., & Argüello, M.F. (2000). Optimizing aircraft routings in response to groundings and delays. *IEE Transactions*, *33*, 931-947.

Bratu, S., & Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, *9* (3), 279 – 298.

Beasley, J.A., & Cao, B. (1998). A Dynamic programming based algorithm for the crew scheduling problem. *Computers and Operations Research*, *25* (7/8), 567-587.

Bélanger, N., Desaulniers, G., Soumis, F., Desrosiers, J., & Lavigne, J. (2006). Weekly airline fleet assignment with homogeneity. *Transportation Research B, 40*(4), 306- 318.

Bélanger, N., Desaulniers, G., Soumis, F., & Desrosiers, J. (2006). Periodic airline fleet assignment with time windows spacing constraints and time dependent revenues. *European Journal of Operational Research*, *3*, 1754-1766.

Berge, M. and Hopperstad, C. A. (1993). Demand driven dispatch: a method for dynamic aircraft capacity assignment: Models and algorithms. *Operations Research*, *41*, 153-168.

Berge, M. (1994). Timetable optimization: Formulation solution approaches and computational issues. *AGIFORS Proc*, 341– 357.

Biggs, M.C.B., Parkhurst, S.C., & Wilson, S.P. (2003). Global optimization approaches to an aircraft routing problem. *European Journal of Operational Research*, *146*, 417-431.

Butchers, E.R., Day, P.R., Goldie, A.P., Miller, S., Meyer, J.A., Ryan, D. M., Scott, A.C., & Wallace, C.A. (2001). Optimized crew scheduling at Air New Zealand. *Interfaces*, *31*, 30-56.

Cao, J.M., & Kanafi, A. (1997). Real-time decision support for integration of airline flight cancellations and delays. part i: Mathematical formulation. *Transportation Planning and Technology*, *20*, 183 - 199.

Cao, J.M., & Kanafi, A. (1997). Real-time decision support for integration of airline flight cancellations and delays. part ii: Algorithm and computational experiments. *Transportation Planning and Technology*, *20*, 201 - 217.

Chan, Y. (1972). Route network improvement in air transportation schedule planning. *Technical Report R72-3 MIT Flight Transportation Laboratory*. Cambridge. MA.

Chu, H.D., Gelman, E, & Johnson, E.L. (1997). Solving large scale crew scheduling problems. *European Journal of Operational Research*, *97*, 260-268.

Chu, S.C.K. (2007). Generating, scheduling and rostering of shift crew-duties: Applications at the Hong Kong International Airport. *European Journal of Operational Research*, *177*(3), 1764 -1778.

Clarke, L., Hane, C., Johnson, E., & Nemhauser, G. (1996). Maintenance and crew considerations in fleet assignment. *Transportation Science*, *30*, 249-260.

Clarke, L., Johnson, E., Nemhauser, G., & Zhu, Z. (1996). The aircraft rotation problem. *Annuals of Operations Research*, *69,* 33-46.

Cohn, A., & Barnhart, C. (2003). Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, *51*, 387–396.

Cordeau, J.F., Stojkoviç, G., Soumis, F., & Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, *35*(4), 375-388.

Daskin, M. S., & Panayotopoulos, N. A. (1989). Lagrangian relaxation approach to assigning aircraft to routes in hub-and-spoke networks. *Transportation Science*, *23*, 91-99.

Dawid, H., König, J., & Strauss, C. (2001). An enhanced rostering model for airline crews. *Computers and Operations Research*, *28*, 671-688.

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

Desrochers, M., & Soumis, F. A. (1989). Column generation approach to the urban transit crew scheduling problem. *Transportation Science*, *23* (1), 1-13.

Dias, T.G., Sousa, J.P., & Cunha, J.F. (2002). Genetic algorithms for the bus driver scheduling problem: A case study. *Journal of the Operational Research Society*, *53*, 324-335.

Dobson, G., & Lederer, P.J. (1993). Airline scheduling and routing in a hub-and-spoke system. *Transportation Science*, *27*(3), 281-197.

Dowling, D., Krishnamoorthy, M., Mackenzie, H., & Sier, D. (1997). Staff rostering at a large international airport. *Annals of Operations Research*, *72*, 125-147.

Eggenberg, N., Salani, M., & Bierlaire, M. (2010). Constraint-specific recovery network for solving airline recovery problems. *Computers and Operations Research*, *37*, 1014-1026.

Etschmaier, M.M., & Mathaisel, D.F.X. (1984). Airline scheduling: The state of the art. *AGIFORS Sympos*. Strasbourg, France.

Etschamier, M.M., & Mathaisel, D.F.X. (1985). Airline scheduling: An overview. *Transportation Science, 19*(2), 127-138.

Feo, T. A., & Bard, J. F. (1989). Flight scheduling and maintenance base planning. *Management Science*, *35*, 1415-1432.

Ftulis, S.G., Giardono, M., Plüss, J.J., & Vota R.J. (1998). Rule based constraint programming: Application to crew assignment. *Expert Systems with Applications*, *15*, 77-85.

Gabteni, S., & Grönkvist, M. (2009). Combining column generation and constraint programming to solve the tail assignment problem. *Annuals of Operations Research*, *171*, 61-76.

Gamache, M., Soumis, F., Villeneuve, D., Desrosiers, J., & Gelinas, E. (1998). The preferential bidding system at Air Canada. *Transportation Science*, *32*, 246-255.

Gopalakrishan, B., & Johnson, E.L. (2005). Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, *140*, 305-337.

Gopalan, R., & Talluri, K. (1998a). Mathematical models in airline schedule planning: A survey. *Annals of Operations Research*, *76*(1), 155-185.

Gopalan, R., & Talluri, K. (1998b). The aircraft maintenance routing problem. *Operations Research*, *46*, 260–271.

Gershkoff, I. (1989). Optimizing flight crew schedules. *Interfaces*, *19*, 29-43.

Guo, Y., Mellouli, T., Suhl, L., & Thiel, M.P. (2006). A partially integrated airline crew scheduling approach with time-dependent crew capacities and multiple home bases. *European Journal of Operational Research*, *171*, 1169-1181.

Graves, G.W., McBride, R.D., Gershkoff, I., Anderson, D., & Mahidhara, D. (1993). Flight crew scheduling. *Management Science, 39*(6), 736-745.

Hane, C.A, Barnhart, C., Johnson, E.L., Marsten, R.E., Nemhauser, G.L., & Sigismondi, G. (1995). The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming* , *70,* 211-232.

Haouari, M., Aissaoui, N., & Mansour, F.Z. (2009). Network flow-based approaches for integrated aircraft fleeting and routing. *European Journal of Operational Research*, *193*, 591-599.

Hentenryck, V.P. (1999). *The OPL Optimization Programming Language*, Cambridge, Massachusetts : MIT Press.

Hoffman, K., & Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, *39*(6), 657-682.

Ioachim, I., Gélina, S., Desrosiers, J., & Soumis. F. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks, 31,* 193-204.

Ioachim, I., Desrosiers, J., Soumis, F., & Bélanger, N. (1999). Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research, 119*(1), 75-90.

Jafari, N., & Zegordi, S.H. (in press). Simultaneous recovery model for aircraft and passengers. *Journal of the Franklin Institute*.

Jarrah, A.I.Z., Yu, G., Krishnamurthy, N., & Rakshit, A. (1993). A Decision support framework for airline flight cancellations and delays. *Transportation Science*, *27*(3), 266-280.

Kim, D., & Barnhart, C. (2007). Flight schedule design for a charter airline. *Computers and Operations Research*, *34,* 1516-1531.

Klabjan, D., Johnson, E.L., Nemhauser, G.L., Gelman, E., & Ramaswamy, S. (2001). Solving large airline crew scheduling problems: Random pairing generation and strong branching. *Computational Optimization and Applications*, *20,* 73-91.

Klabjan, D., Johnson, E.L., Nemhauser, G.L, Gelman, E., & Ramaswamy, S. (2001). Airline crew scheduling with regularity. *Transportation Science, 35*(4), 359-374.

Klabjan, D., Johnson, E.J., Nemhauser, G.L., Gelman E. & Ramaswamy, S. (2002). Airline crew scheduling with time windows and plane-count constraints. *Transportation Science, 36*(3), 337-348.

Kohl, N., & Karisch, S.E. (2004). Airline crew rostering: Problem types, modeling and optimization. *Annals of Operations Research*, *12,* 223-257.

Kornilakis, H. & Stamatopoulos, P. (2002). Crew pairing optimization with genetic algorithms. *I.P. Vlahavas and C.D. Spyropoulos (Eds.), 2308,* 109-120.

Lagerholm, M., Peterson, C., & Söderberg, B. (2000). Theory and methodology: Airline crew scheduling using potts mean field techniques. *European Journal of Operational Research*, *120,* 81-96.

Lavoie, S., Minoux, M., & Odier, E. (1988). A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, *35,* 45–58.

Lettovsky, L. (1997). Airline Operations Recovery: An Optimization Approach. PhD thesis, Georgia Institute of Technology.

Lettovsky, L., Johnson, E.L., & Nemhauser, G.L. (2000). Airline crew recovery. *Transportation Science*, *34*(4), 337-348.

Levin, A. (1971). Scheduling and fleet routing models for transportation systems. *Transportation Science*, *5,* 232-255.

Lohatepanont, M., & Barnhart, C. (2004). Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *Transportation Science, 8*(1), 19-32.

Lou, S., & Yu, G. (1997). On the airline schedule perturbation problem caused by the ground delay program. *Transportation Science*, *31*(4), 298-311.

Lucic, P., & Teodorovic, D. (1999). Simulated annealing for the multi-objective aircrew rostering problem. *Transportation Research Part A, 33,* 19-45.

Marsten, R., & Berge, M. (1994). Timetable optimization: Formulation solution approaches and computational issues. *AGIFORS Proc*. 341–35.

Mathaisel, D.F.X. (1996). Decision support for airline system operations control and irregular operations. *Computers and Operations Research*, *23*(11), 1083 - 1098.

Medard, C.P., & Sawhney, N. (2003). Airline crew scheduling from planning to operations. *Carmen Systems, Research and Technology Reports*.

Mercier, A., Cordeau, J.F., & Soumis, F. (2005). A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers and Operations Research, 32*(6), 1451-1476.

Mercier, A., & Soumis, F. (2007). An integrated aircraft routing, crew scheduling and flight retiming model. *Computers and Operations Research, 34*(8), 2251-2265.

Mingozzi, A., Boschetti, M.A., Ricciardelli, S., & Bianco, L. (1999). A set partitioning approach to the crew scheduling problem. *Operations Research., 47*(6), 873-888.

Moudani, W.E., & Camino, F.M. (2000). A dynamic approach for aircraft assignment and maintenance scheduling by airlines. *Journal of Air Transport Management, 6,* 233-237.

Nissen, R., & Haase, K. (2006). Duty-period-based network model for crew rescheduling in European Airlines, *Journal of Scheduling, 9,* 255-278.

Pinedo, M.L. (2004). *Planning and Scheduling in Manufacturing and Services*, New York : Springer

Rexing, B., Barnhart, C., Kniker, T., Jarrah, A., & Krishnamurthy, N. (2000). Airline fleet assignment with time windows. *Transportation Science, 34,* 1-20.

Rosenberger, J.M., Johnson, E.L., & Nemhauser, G.L. (2003). Rerouting aircraft for airline recovery. *Transportation Science*, *37*(4), 408-421.

Rubin, J. (1973). A technique for the solution of massive set covering problems with application to airline crew scheduling. *Transportation Science*, *7,* 34-48.

Rushmeier, R., & Kontogiorgis, S. (1997). Advances in the optimization of airline fleet assignment. *Transportation Science*, *31,* 159-169.

Özdemir, H.T., & Mohan, C.K. (2001). Flight graph based genetic algorithm for crew scheduling in airlines. *Information Science, 133,* 165-173.

Sarac, A., Rajan, B., & Rump, C. M. (2006). A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research, 175,* 1850-1869.

Sandhu, R., & Klabjan, D. (2004). Integrated airline planning. *AGIFORMS Symposium*.

Sherali, H.D., Bish, E.K., & Zhu, X. (2006). Airline fleet assignment concepts models and algorithms. *European Journal of Operational Research*, *172,* 1-30.

Simpson, R.W. (1996). Computerized schedule construction for an airline transportation system. *MIT Flight Transportation Laboratory Report FT-66-3,* Cambridge, MA.

Stefan, N.K., & Karisch E. (2004). Airline crew rostering: Problem types, modeling and optimization. *Annals of Operations Research*, *127,* 223-257.

Souai, N., & Teghem, J. (2009). Genetic algorithm based approach for the integrated airline crew-pairing and rostering problem. *European Journal of Operational Research, 199*(3), 674-683.

Soumis, F., Ferland ,J.A., & Rousseau J.M. (1980). A model for large scale aircraft routing and scheduling problems. *Transportation Research, 14B,* 191-201.

Stojkovic, M., Soumis, F., & Desrosiers, J. (1998). The operational crew scheduling problem. *Transportation Science*, *32*(3), 232- 245.

Stojkovic, G., Soumis, F., Desrosiers, J., & Solomon, M.M. (2002). An optimization model for a real-time flight scheduling problem. *Transportation Research Part A*, *36,* 779–788.

Subramanian, R., Scheff, R.P., Quillinan, J.D., Wiper, D.S., & Marsten, R.E. (1994). Coldstart: Fleet assignment at Delta Airlines. *Interfaces, 24*(1), 104-120.

Talluri, K.T. (1996). Swapping applications in a daily airline fleet assignment. *Transportation Science*, *30*(3), 237-248.

Talluri, K.T. (1998). The four-day aircraft maintenance routing problem. *Transportation Science*, *32* (1), 43-53.

Tang, C.H., Yan, S., & Chen, Y.H. (2008). An integrated model and solution algorithms for passenger cargo and combi flight scheduling. *Transportation Research Part E*, *44,* 1004-1024.

Teodorovic, D., & Guberinic, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research, 15,* 178-182.

Teodorovic, D. (1985). A model for designing the meteorologically most reliable airline schedule. *European Journal of Operations Research*, *21,* 156-164.

Teodorovic, D., & Lucic P. (1998). A fuzzy set theory approach to the aircrew rostering problem. *Fuzzy Sets and Systems*, *95,* 261-271.

Teodorovic, D., & Stojkovic, G. (1990). Model for operational daily airline scheduling. *Transportation Planning and Technology*, *14,* 273- 285.

Teodorovic, D., & Stojkovic, G. (1995). Model to reduce airline schedule disturbances. *Journal of Transportation Engineering*, *121*(4), 324-331.

Thengvall, B.G., Bard J.F., & Yu, G. (2000). Balancing user preferences for aircraft schedule recovery during irregular operations. *IIE Transactions*, *32,* 181-193.

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

Wei, G., Yu, G., & Song, M. (1997). Optimization model and algorithm for crew management during airline irregular operations. *Journal of Combinatorial Optimization*, *1,* 305-321.

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

Yan, S., & Yang, D-H. (1996). A decision support framework for handling schedule perturbation. *Transportation Research B*, *30*(6), 405 - 419.

Yan, S., & Young, H.F. (1996). A decision support framework for multi-fleet routing and multi-stop flight scheduling. *Transportation Research*, *30A,* 379-398.

Yan, S., & Tu, Y.P. (1996). Multi-fleet routing and multi-stop flight scheduling for schedule perturbation. *European Journal of Operational Research*, *103,* 155-169.

Yan, S., & Tu, Y.P. (2002). A network model for airline cabin crew scheduling. *European Journal of Operational Research*, *140,* 531-540.

Yan, S., & Chang, J.C. (2002). Discrete Optimization: Airline cockpit crew scheduling. *European Journal of Operations Research*, *136,* 501-511.

Yan, S., Tung, T.T., & Tu, Y.P. (2002). Optimal construction of airline individual crew pairings. *Computers and Operations Research*, *29,* 341-363.

Yan, S., & Tseng, C.H. (2002). A passenger demand model for airline flight scheduling and fleet routing. *Computers and Operations Research*, *2,* 1559-1581.

Yan, S., Chen, S.C., & Chen C.H. (2006). Air cargo fleet routing and timetable setting with multiple on-time demands. *Transportation Research Part E*, *42,* 409-430.

Yan, S., Tang, C.H., & Lee, M.C. (2007). A flight scheduling model for Taiwan Airlines under market competitions. *Omega, 35,* 61-74.

Yan, S., Tang, C.H., & Fu T.C. (2008). An airline scheduling model and solution algorithms under stochastic demands. *European Journal of Operational Research*, *190,* 22-39.

Yen, J.W., & Birge J.R. (2006). A Stochastic programming approach to the airline crew scheduling problem. *Transportation Science, 40*(1), 3-14.

APPENDICES

APPENDIX A

FLIGHT SCHEDULES AND INITIAL AIRCRAFT AND CREW SCHEDULES OF EXAMPLE AIRLINES

Table A.1 Fleet Information of OnurAir

OnurAir Fleet Information total # of aircraft = 9	BOEING MD-88	BOEING MD-83
Number of Aircraft	5	4
Passengers capacity	172	172
Cargo	1,253 cu ft (35.5 cu m)	1,103 cu ft (28.7 cu m)
Maximum Fuel Capacity	5,840 U.S. gal (22,106 L)	7,000 U.S. Gal (26,4951 L)
Maximum Takeoff Weight	149,500 lb (67,812 kg)	160,000 lb (72,575 kg)
Maximum Range	2,052 nautical miles (3,798 km)	2,504 nautical miles (4,635 km)
Typical Cruise Speed at 35,000 feet	504 mph (811 km/h)	504 mph (811 km/h)
Wing Span	107 ft 8 in (32.8 m)	107 ft 8 in (32.8 m)
Overall Length	147 ft 8 in (45.1 m)	147 ft 8 in (45.1 m)
Tail Height	29 ft 6 in (9.05 m)	29 ft 6 in (9.05 m)

Table A.2 Domestic flight schedule, initial crew schedule and initial aircraft-flight assignments of OnurAir

Flight No	Origin		Destination		Departure	Arrival	Aircraft	Crew
OHY010	IST	İstanbul-Atatürk	ADA	Adana-ADANA	06:45	08:15	MD83c	C10
OHY011	ADA	Adana-ADANA	IST	İstanbul-Atatürk	09:10	10:40	MD83c	C10
OHY016	IST	İstanbul-Atatürk	ADA	Adana-ADANA	15:00	16:30	MD83c	C04
OHY017	ADA	Adana-ADANA	IST	İstanbul-Atatürk	17:30	19:00	MD83c	C04
OHY018	IST	İstanbul-Atatürk	ADA	Adana-ADANA	19:00	20:30	MD88a	C02
OHY019	ADA	Adana-ADANA	IST	İstanbul-Atatürk	21:30	23:00	MD88a	C02
OHY022	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	08:15	09:15	MD88c	C02
OHY023	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	10:45	11:45	MD88c	C02
OHY024	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	16:45	17:45	MD83b	C09
OHY025	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	18:45	19:45	MD83b	C09
OHY026	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	20:30	21:30	MD83d	C01
OHY027	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	07:30	08:30	MD83b	C01
OHY028	IST	İstanbul-Atatürk	BJV	Bodrum-BODRUM MİLAS	11:55	12:55	MD83d	C10
OHY029	BJV	Bodrum-BODRUM MİLAS	IST	İstanbul-Atatürk	13:45	14:45	MD83d	C10
OHY034	IST	İstanbul-Atatürk	DIY	Diyarbakır-DIYARBAKIR	06:45	08:30	MD88a	C11
OHY035	DIY	Diyarbakır-DIYARBAKIR	IST	İstanbul-Atatürk	09:30	11:15	MD88a	C11
OHY036	IST	İstanbul-Atatürk	DIY	Diyarbakır-DIYARBAKIR	18:45	20:30	MD88b	C06
OHY037	DIY	Diyarbakır-DIYARBAKIR	IST	İstanbul-Atatürk	21:30	23:15	MD88b	C06
OHY038	IST	İstanbul-Atatürk	ERZ	Erzurum-ERZURUM	12:30	14:20	MD88c	C08
OHY039	ERZ	Erzurum-ERZURUM	IST	İstanbul-Atatürk	15:20	17:10	MD88c	C08
OHY042	IST	İstanbul-Atatürk	GZT	Gaziantep-GAZIANTEP	09:30	11:15	MD83a	C05
OHY043	GZT	Gaziantep-GAZIANTEP	IST	İstanbul-Atatürk	12:15	14:00	MD83a	C05
OHY050	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	07:45	08:45	MD88d	C04
OHY051	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	10:50	11:45	MD88d	C04
OHY054	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	16:30	17:30	MD88d	C03
OHY055	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	18:50	19:45	MD88d	C03
OHY058	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	20:45	21:45	MD88e	C11
OHY059	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	07:30	08:25	MD83d	C06
OHY062	IST	İstanbul-Atatürk	MLX	Malatya-MALATYA	12:15	13:45	MD88e	C01
OHY063	MLX	Malatya-MALATYA	IST	İstanbul-Atatürk	14:30	16:00	MD88e	C01
OHY066	IST	İstanbul-Atatürk	KSY	Kayseri-ERKİLET	07:15	08:30	MD88e	C03
OHY067	KSY	Kayseri-ERKİLET	IST	İstanbul-Atatürk	09:15	10:30	MD88e	C03
OHY074	IST	İstanbul-Atatürk	SZF	Samsun-CARSAMBA	09:30	10:45	MD83b	C09
OHY075	SZF	Samsun-CARSAMBA	IST	İstanbul-Atatürk	11:45	13:00	MD83b	C09
OHY080	IST	İstanbul-Atatürk	TZX	Trabzon-TRABZON	06:50	08:25	MD88b	C07
OHY081	TZX	Trabzon-TRABZON	IST	İstanbul-Atatürk	09:20	10:55	MD88b	C07
OHY086	IST	İstanbul-Atatürk	TZX	Trabzon-TRABZON	18:45	20:20	MD83a	C12
OHY087	TZX	Trabzon-TRABZON	IST	İstanbul-Atatürk	21:15	22:50	MD83a	C12

Table A.3 Domestic flight schedule, original crew schedule and original aircraft-flight assignments of OnurAir (amended for constraint programming)

Flight No	Origin		Destination		Departure	Arrival	Aircraft	Crew
OHY010	IST	İstanbul-Atatürk	ADA	Adana-ADANA	06:45	08:15	MD88d	C12
OHY011	ADA	Adana-ADANA	IST	İstanbul-Atatürk	09:10	10:40	MD88d	C12
OHY016	IST	İstanbul-Atatürk	ADA	Adana-ADANA	15:00	16:30	MD88a	C03
OHY017	ADA	Adana-ADANA	IST	İstanbul-Atatürk	17:30	19:00	MD88a	C03
OHY018	IST	İstanbul-Atatürk	ADA	Adana-ADANA	19:00	20:30	MD83c	C04
OHY019	ADA	Adana-ADANA	IST	İstanbul-Atatürk	21:30	23:00	MD83c	C04
OHY022	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	08:15	09:15	MD88b	C02
OHY023	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	10:45	11:45	MD88b	C02
OHY024	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	16:45	17:45	MD83b	C07
OHY025	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	18:45	19:45	MD83b	C06
OHY026	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	20:30	21:30	MD88e	C06
OHY027	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	07:30	08:30	MD88e	C07
OHY028	IST	İstanbul-Atatürk	BJV	Bodrum-BODRUM MİLAS	11:55	12:55	MD88d	C05
OHY029	BJV	Bodrum-BODRUM MİLAS	IST	İstanbul-Atatürk	13:45	14:45	MD88d	C05
OHY034	IST	İstanbul-Atatürk	DIY	Diyarbakır-DIYARBAKIR	06:45	08:30	MD88a	C11
OHY035	DIY	Diyarbakır-DIYARBAKIR	IST	İstanbul-Atatürk	09:30	11:15	MD88a	C11
OHY036	IST	İstanbul-Atatürk	DIY	Diyarbakır-DIYARBAKIR	18:45	20:30	MD83d	C02
OHY037	DIY	Diyarbakır-DIYARBAKIR	IST	İstanbul-Atatürk	21:30	23:15	MD83d	C02
OHY038	IST	İstanbul-Atatürk	ERZ	Erzurum-ERZURUM	12:30	14:20	MD88c	C08
OHY039	ERZ	Erzurum-ERZURUM	IST	İstanbul-Atatürk	15:20	17:10	MD88c	C08
OHY042	IST	İstanbul-Atatürk	GZT	Gaziantep-GAZIANTEP	09:30	11:15	MD83b	C07
OHY043	GZT	Gaziantep-GAZIANTEP	IST	İstanbul-Atatürk	12:15	14:00	MD83b	C07
OHY050	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	07:45	08:45	MD83d	C08
OHY051	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	10:50	11:45	MD83d	C08
OHY054	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	16:30	17:30	MD88b	C01
OHY055	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	18:50	19:45	MD88b	C09
OHY058	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	20:45	21:45	MD83b	C09
OHY059	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	07:30	08:25	MD83b	C01
OHY062	IST	İstanbul-Atatürk	MLX	Malatya-MALATYA	12:15	13:45	MD83a	C12
OHY063	MLX	Malatya-MALATYA	IST	İstanbul-Atatürk	14:30	16:00	MD83a	C12
OHY066	IST	İstanbul-Atatürk	KSY	Kayseri-ERKİLET	07:15	08:30	MD83a	C03
OHY067	KSY	Kayseri-ERKİLET	IST	İstanbul-Atatürk	09:15	10:30	MD83a	C03
OHY074	IST	İstanbul-Atatürk	SZF	Samsun-CARSAMBA	09:30	10:45	MD88e	C01
OHY075	SZF	Samsun-CARSAMBA	IST	İstanbul-Atatürk	11:45	13:00	MD88e	C01
OHY080	IST	İstanbul-Atatürk	TZX	Trabzon-TRABZON	06:50	08:25	MD83c	C05
OHY081	TZX	Trabzon-TRABZON	IST	İstanbul-Atatürk	09:20	10:55	MD83c	C05
OHY086	IST	İstanbul-Atatürk	TZX	Trabzon-TRABZON	18:45	20:20	MD88c	C10
OHY087	TZX	Trabzon-TRABZON	IST	İstanbul-Atatürk	21:15	22:50	MD88c	C10

Table A.4 The number of available crews for each flight leg

Flight No	Origin	Destination	Departure	Arrival	# of available crew
OHY010	IST	ADA	06:45	08:15	1
OHY011	ADA	IST	09:10	10:40	1
OHY016	IST	ADA	15:00	16:30	3
OHY017	ADA	IST	17:30	19:00	3
OHY018	IST	ADA	19:00	20:30	2
OHY019	ADA	IST	21:30	23:00	2
OHY022	IST	AYT	08:15	09:15	1
OHY023	AYT	IST	10:45	11:45	1
OHY024	IST	AYT	16:45	17:45	3
OHY025	AYT	IST	18:45	19:45	3
OHY026	IST	AYT	20:30	21:30	2
OHY027	AYT	IST	07:30	08:30	1
OHY028	IST	BJV	11:55	12:55	4
OHY029	BJV	IST	13:45	14:45	4
OHY034	IST	DIY	06:45	08:30	1
OHY035	DIY	IST	09:30	11:15	1
OHY036	IST	DIY	18:45	20:30	2
OHY037	DIY	IST	21:30	23:15	2
OHY038	IST	ERZ	12:30	14:20	5
OHY039	ERZ	IST	15:20	17:10	5
OHY042	IST	GZT	09:30	11:15	2
OHY043	GZT	IST	12:15	14:00	2
OHY050	IST	ADB	07:45	08:45	1
OHY051	ADB	IST	10:50	11:45	1
OHY054	IST	ADB	16:30	17:30	3
OHY055	ADB	IST	18:50	19:45	3
OHY058	IST	ADB	20:45	21:45	2
OHY059	ADB	IST	07:30	08:25	1
OHY062	IST	MLX	12:15	13:45	4
OHY063	MLX	IST	14:30	16:00	4
OHY066	IST	KSY	07:15	08:30	1
OHY067	KSY	IST	09:15	10:30	1
OHY074	IST	SZF	09:30	10:45	2
OHY075	SZF	IST	11:45	13:00	2
OHY080	IST	TZX	06:50	08:25	1
OHY081	TZX	IST	09:20	10:55	1
OHY086	IST	TZX	18:45	20:20	2
OHY087	TZX	IST	21:15	22:50	2

Table A.5 Fleet Information of AtlsJet

AtlasJet Fleet Information total # of aircraft = 9					
	AIRBUS 321-231	AIRBUS A321-131	AIRBUS A321-231	BOEING 757-200	BOEING 757-200
Number of Aircraft	3	2	1	1	2
Passenger Capacity	210	204	199	210	200
Cruising Speed	900 km/h max	900 km/h max	900 km/h max	900 km/h max	900 km/h max
Cruising Altitude	12.811 m	12.000 m	12.000 m	12.811 m	12.811 m
Length	47,32 m	44,5 m	44,5 m	47,32 m	47,32 m
Height	13,56 m	11,75m	11,75m	13,56 m	13,56 m
Width of the Cabin	3,95 m	3,95 m	3,95 m	3,75 m	3,75 m
Wing Span	34,1 m	34,1 m	34,1 m	38,05 m	38,05 m
Take-Off Weight	89.000 kg	85.000 kg	89.000 kg	108.862 kg	108.862 kg
Landing Weight	75.500 kg	74.500 kg	75.500 kg	89.811 kg	89.811 kg
Fuel Capacity	19.079 kg	18.960 kg	19.079 kg	34.473 kg	34.473 kg
Noise Level	ICAO Annex 16, CH3	ICAO Annex 16, CH3	ICAO Annex 16, CH3	ICAO Annex 16, CH3	ICAO Annex 16, CH3
Cargo Capacity	50,25 m3	48,87 m3	50,25 m3	50,8 m3	50,8 m3
Maximum Range	5600 km	5600 km	5600 km	6296 km	6574,6 km

Table A.6 Domestic flight schedule, initial crew schedule and initial aircraft-flight assignments of AtlasJet

Flight No	Origin		Destination		Departure	Arrival	Aircraft	Crew
KK10	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	07:30	08:35	A321c199a	C07
KK11	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	09:30	10:35	A321c199a	C07
KK14	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	15:00	16:05	A321c210a	C04
KK15	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	17:00	18:05	A321c210a	C04
KK18	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	19:00	20:05	A321c210a	C04
KK19	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	21:00	22:10	A321c210a	C04
KK20	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	07:00	07:55	B737c200b	C10
KK21	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	08:45	09:40	B737c200b	C10
KK24	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	16:00	16:55	B737c200a	C06
KK25	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	18:00	18:55	B737c200a	C06
KK28	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	11:30	12:25	A321c199a	C07
KK29	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	14:00	14:55	A321c199a	C07
KK40	IST	İstanbul-Atatürk	BJV	Bodrum-BODRUM MİLAS	07:00	08:05	A321c204b	C02
KK41	BJV	Bodrum-BODRUM MİLAS	IST	İstanbul-Atatürk	09:00	10:05	A321c204b	C02
KK42	IST	İstanbul-Atatürk	BJV	Bodrum-BODRUM MİLAS	11:30	12:35	B737c200b	C11
KK43	BJV	Bodrum-BODRUM MİLAS	IST	İstanbul-Atatürk	13:50	14:55	B737c200b	C11
KK44	IST	İstanbul-Atatürk	BJV	Bodrum-BODRUM MİLAS	18:00	19:05	A321c204a	C09
KK45	BJV	Bodrum-BODRUM MİLAS	IST	İstanbul-Atatürk	20:00	21:05	A321c204a	C09
KK1011	ECN	Kıbrıs - Ercan	IST	İstanbul-Atatürk	07:00	08:30	A321c210c	C01
KK1012	IST	İstanbul-Atatürk	ECN	Kıbrıs - Ercan	09:30	11:00	A321c210c	C01
KK1014	IST	İstanbul-Atatürk	ECN	Kıbrıs - Ercan	16:00	17:30	A321c210b	C03
KK1015	ECN	Kıbrıs - Ercan	IST	İstanbul-Atatürk	19:00	20:30	B737c210a	C05
KK1016	IST	İstanbul-Atatürk	ECN	Kıbrıs - Ercan	21:30	23:00	B737c210a	C05
KK1017	ECN	Kıbrıs - Ercan	IST	İstanbul-Atatürk	21:30	23:00	A321c210b	C03
KK1030	ADB	Izmir-ADNAN MENDERES	ECN	Kıbrıs - Ercan	17:00	18:15	B737c210a	C05
KK1031	ECN	Kıbrıs - Ercan	ADB	Izmir-ADNAN MENDERES	15:00	16:15	B737c210a	C05
KK1053	ECN	Kıbrıs - Ercan	ADA	Adana-ADANA	12:00	12:45	A321c210c	C01
KK1054	ADA	Adana-ADANA	ECN	Kıbrıs - Ercan	13:30	14:15	A321c210c	C01
KK1055	ECN	Kıbrıs - Ercan	ADA	Adana-ADANA	18:30	19:15	A321c210b	C03
KK1056	ADA	Adana-ADANA	ECN	Kıbrıs - Ercan	20:00	20:45	A321c210b	C03
KK4012	IST	İstanbul-Atatürk	AYT	Antalya-ANTALYA	11:00	12:05	A321c204b	C02
KK4013	AYT	Antalya-ANTALYA	IST	İstanbul-Atatürk	13:00	14:05	A321c204b	C02
KK4022	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	19:45	20:40	B737c200a	C06
KK4023	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	21:30	22:25	B737c200a	C06
KK6200	IST	İstanbul-Atatürk	EBL	Irak- Erbil	10:30	12:45	A321c204a	C12
KK6201	EBL	Irak- Erbil	IST	İstanbul-Atatürk	14:00	16:30	A321c204a	C12
KK625	IST	İstanbul-Atatürk	PRN	Sırbistan - Pristina	09:50	10:10	A321c210a	C08
KK626	PRN	Sırbistan - Pristina	IST	İstanbul-Atatürk	11:30	14:00	A321c210a	C08

Table A.7 Fleet Information of Pegasus Airlines

Pegasus Fleet Information total # of aircraft = 36						
	BOEING 737-800	BOEING 737-400	BOEING 737-500	BOEING 737-800	AIRBUS A319	AIRBUS A320
Number of Aircraft	25	2	3	1	3	2
Maximum Take-Off Weight	79.000 KG	68.038 KG	58.966 KG	79.000 KG	75.500 KG	77.700 KG
Wing Span	35.78 M	28.90 M	28.90 M	35.78 M	34.10 M	34.10 M
Length	39.47 M	36.40 M	31.01 M	39.47 M	33.84 M	37.50 M
Height	12.55 M	11.125 M	11.125 M	12.55 M	11.76 M	11.76 M
Cruising Speed	858 KM/H	797 KM/H	797 KM/H	858 KM/H	833 KM/H	833 KM/H
Passenger Capacity	189	170	131	189	144	180
Seat Pitch (Inches)	29-32	29-30	29-30	29-32	29-30	28-29
Maximum Cruising Altitude	41.000 FT	37.000 FT	37.000 FT	41.000 FT	39.800 FT	39.800 FT
Maximum Passenger Range	4.630 KM	3.148 KM	3250 KM	4.630 KM	3.334 KM	4.815 KM
Maximum Cargo Capacity	8.408 KG / 45.05 M^3	7.491 KG / 39.22M^3	4.349 KG / 23.3 M^3	8.408 KG / 45.05 M^3	6.786 KG / 27.6 M^3	6.786 / 37.42 M^3
Maximum Fuel Capacity	20.896 KG	15.933 KG	15.933 KG	20.896 KG	18.730 KG	19.158 KG

Table A.8 Domestic flight schedule, initial crew schedule and initial aircraft-flight assignments of Pegasus Airlines

Flight No	Origin		Destination		Departure	Arrival	Aircraft	Crew
PC100	SAW	Istanbul-SABİHA GÖKÇEN	ESB	Ankara-ESENBOĞA	8:30	9:25	A319b2	C16
PC101	ESB	Ankara-ESENBOĞA	SAW	Istanbul-SABİHA GÖKÇEN	9:50	10:45	A319b2	C16
PC102	SAW	Istanbul-SABİHA GÖKÇEN	ESB	Ankara-ESENBOĞA	6:50	7:45	B800a3	C23
PC103	ESB	Ankara-ESENBOĞA	SAW	Istanbul-SABİHA GÖKÇEN	8:10	9:05	B800a3	C23
PC104	SAW	Istanbul-SABİHA GÖKÇEN	ESB	Ankara-ESENBOĞA	9:50	10:45	B800a1	C07
PC105	ESB	Ankara-ESENBOĞA	SAW	Istanbul-SABİHA GÖKÇEN	11:10	12:05	B800a1	C07
PC106	SAW	Istanbul-SABİHA GÖKÇEN	ESB	Ankara-ESENBOĞA	17:10	18:05	B800a6	C37
PC107	ESB	Ankara-ESENBOĞA	SAW	Istanbul-SABİHA GÖKÇEN	18:30	19:25	B800a6	C37
PC108	SAW	Istanbul-SABİHA GÖKÇEN	ESB	Ankara-ESENBOĞA	13:15	14:10	B800b1	C01
PC109	ESB	Ankara-ESENBOĞA	SAW	Istanbul-SABİHA GÖKÇEN	14:35	15:30	B800b1	C01
PC110	SAW	Istanbul-SABİHA GÖKÇEN	ESB	Ankara-ESENBOĞA	19:55	20:50	A319b2	C35
PC111	ESB	Ankara-ESENBOĞA	SAW	Istanbul-SABİHA GÖKÇEN	21:15	22:10	A319b2	C35
PC112	SAW	Istanbul-SABİHA GÖKÇEN	ESB	Ankara-ESENBOĞA	18:50	19:45	B800a12	C02
PC113	ESB	Ankara-ESENBOĞA	SAW	Istanbul-SABİHA GÖKÇEN	20:10	21:05	B800a12	C02
PC114	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	6:40	7:45	B400a2	C21
PC115	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	8:10	9:10	B400a2	C21
PC116	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	9:55	10:55	B800a19	C09
PC117	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	11:20	12:20	B800a19	C09
PC118	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	12:50	13:50	B800a4	C24
PC119	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	14:15	15:15	B800a4	C24
PC120	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	18:30	19:30	B800a11	C29
PC121	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	19:55	20:55	B500a1	C29
PC122	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	19:55	21:00	B800a15	C18
PC123	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	21:25	22:25	B800a15	C18
PC124	SAW	Istanbul-SABİHA GÖKÇEN	ADA	Adana-ADANA	6:05	7:25	B800a17	C29
PC125	ADA	Adana-ADANA	SAW	Istanbul-SABİHA GÖKÇEN	7:50	9:15	B800a17	C29
PC126	SAW	Istanbul-SABİHA GÖKÇEN	ADA	Adana-ADANA	20:35	21:55	B800a25	C34
PC127	ADA	Adana-ADANA	SAW	Istanbul-SABİHA GÖKÇEN	22:20	23:45	B800a25	C34
PC128	SAW	Istanbul-SABİHA GÖKÇEN	ADA	Adana-ADANA	15:40	17:00	B800a16	C42
PC129	ADA	Adana-ADANA	SAW	Istanbul-SABİHA GÖKÇEN	17:25	18:50	B800a16	C42
PC130	SAW	Istanbul-SABİHA GÖKÇEN	SZF	Samsun-CARSAMBA	20:10	21:30	B800a23	C42
PC131	SZF	Samsun-CARSAMBA	SAW	Istanbul-SABİHA GÖKÇEN	21:55	23:15	B800a23	C42
PC132	SAW	Istanbul-SABİHA GÖKÇEN	TZX	Trabzon-TRABZON	15:30	17:10	B800a18	C05
PC133	TZX	Trabzon-TRABZON	SAW	Istanbul-SABİHA GÖKÇEN	17:35	19:20	B800a18	C05
PC134	SAW	Istanbul-SABİHA GÖKÇEN	TZX	Trabzon-TRABZON	8:20	10:00	B800a13	C15
PC135	TZX	Trabzon-TRABZON	SAW	Istanbul-SABİHA GÖKÇEN	10:25	12:10	A319b1	C15
PC136	SAW	Istanbul-SABİHA GÖKÇEN	TZX	Trabzon-TRABZON	22:45	0:25	A319b1	C18
PC137	TZX	Trabzon-TRABZON	SAW	Istanbul-SABİHA GÖKÇEN	6:00	7:45	B800a7	C19
PC138	SAW	Istanbul-SABİHA GÖKÇEN	TZX	Trabzon-TRABZON	20:25	22:05	B800a3	C05
PC139	TZX	Trabzon-TRABZON	SAW	Istanbul-SABİHA GÖKÇEN	22:30	0:15	B800a3	C05
PC140	SAW	Istanbul-SABİHA GÖKÇEN	AYT	Antalya-ANTALYA	6:30	7:40	B800a15	C26
PC141	AYT	Antalya-ANTALYA	SAW	Istanbul-SABİHA GÖKÇEN	8:05	9:15	B800a15	C26
PC142	SAW	Istanbul-SABİHA GÖKÇEN	AYT	Antalya-ANTALYA	17:05	18:15	B400a1	C25
PC143	AYT	Antalya-ANTALYA	SAW	Istanbul-SABİHA GÖKÇEN	18:40	19:50	B400a1	C25
PC144	SAW	Istanbul-SABİHA GÖKÇEN	AYT	Antalya-ANTALYA	12:15	13:25	A319b3	C23
PC145	AYT	Antalya-ANTALYA	SAW	Istanbul-SABİHA GÖKÇEN	13:50	15:00	A320b2	C23
PC146	SAW	Istanbul-SABİHA GÖKÇEN	AYT	Antalya-ANTALYA	20:05	21:15	B800a20	C11
PC147	AYT	Antalya-ANTALYA	SAW	Istanbul-SABİHA GÖKÇEN	21:40	22:50	B800a14	C11
PC148	SAW	Istanbul-SABİHA GÖKÇEN	AYT	Antalya-ANTALYA	8:00	9:10	B500a2	C17
PC149	AYT	Antalya-ANTALYA	SAW	Istanbul-SABİHA GÖKÇEN	9:35	10:45	A319b3	C17
PC152	SAW	Istanbul-SABİHA GÖKÇEN	SZF	Samsun-CARSAMBA	6:10	7:30	B800a5	C01
PC153	SZF	Samsun-CARSAMBA	SAW	Istanbul-SABİHA GÖKÇEN	7:55	9:15	B800a5	C01
PC156	SAW	Istanbul-SABİHA GÖKÇEN	GZT	Gaziantep-GAZIANTEP	5:45	7:15	B800a1	C07
PC157	GZT	Gaziantep-GAZIANTEP	SAW	Istanbul-SABİHA GÖKÇEN	7:40	9:15	B800a1	C07
PC158	SAW	Istanbul-SABİHA GÖKÇEN	GZT	Gaziantep-GAZIANTEP	20:15	21:45	B800a22	C37
PC159	GZT	Gaziantep-GAZIANTEP	SAW	Istanbul-SABİHA GÖKÇEN	22:30	0:00	B800a22	C37

Table A.8 Domestic flight schedule, initial crew schedule and initial aircraft-flight assignments of Pegasus Airlines (cont.)

Flight No	Origin		Destination		Departure	Arrival	Aircraft	Crew
PC162	SAW	Istanbul-SABİHA GÖKÇEN	ASR	Kayseri-ERKİLET	6:20	7:35	B800b1	C09
PC163	ASR	Kayseri-ERKİLET	SAW	Istanbul-SABİHA GÖKÇEN	8:00	9:15	B800b1	C09
PC164	SAW	Istanbul-SABİHA GÖKÇEN	ASR	Kayseri-ERKİLET	20:10	21:25	A320b1	C16
PC165	ASR	Kayseri-ERKİLET	SAW	Istanbul-SABİHA GÖKÇEN	21:50	23:05	A320b1	C16
PC166	SAW	Istanbul-SABİHA GÖKÇEN	ASR	Kayseri-ERKİLET	8:40	10:00	B800a7	C30
PC167	ASR	Kayseri-ERKİLET	SAW	Istanbul-SABİHA GÖKÇEN	10:25	11:45	B800a7	C30
PC168	SAW	Istanbul-SABİHA GÖKÇEN	MLX	Malatya-MALATYA	18:30	20:00	B800a5	C27
PC169	MLX	Malatya-MALATYA	SAW	Istanbul-SABİHA GÖKÇEN	20:45	22:20	B800a5	C27
PC172	SAW	Istanbul-SABİHA GÖKÇEN	BJV	Bodrum-BODRUM MİLAS	6:35	7:45	B500a1	C24
PC173	BJV	Bodrum-BODRUM MİLAS	SAW	Istanbul-SABİHA GÖKÇEN	8:10	9:20	B500a1	C24
PC174	SAW	Istanbul-SABİHA GÖKÇEN	BJV	Bodrum-BODRUM MİLAS	18:35	19:45	B800a8	C36
PC175	BJV	Bodrum-BODRUM MİLAS	SAW	Istanbul-SABİHA GÖKÇEN	20:10	21:20	B800a8	C36
PC176	SAW	Istanbul-SABİHA GÖKÇEN	BJV	Bodrum-BODRUM MİLAS	20:00	21:10	B800a16	C17
PC177	BJV	Bodrum-BODRUM MİLAS	SAW	Istanbul-SABİHA GÖKÇEN	21:35	22:45	B800a16	C17
PC178	SAW	Istanbul-SABİHA GÖKÇEN	DLM	Dalaman-DALAMAN	6:00	7:15	B800a4	C13
PC179	DLM	Dalaman-DALAMAN	SAW	Istanbul-SABİHA GÖKÇEN	8:10	9:20	B800a4	C13
PC180	SAW	Istanbul-SABİHA GÖKÇEN	HTY	Hatay-HATAY	20:20	21:55	B800a18	C25
PC181	HTY	Hatay-HATAY	SAW	Istanbul-SABİHA GÖKÇEN	22:40	0:15	B800a18	C25
PC182	SAW	Istanbul-SABİHA GÖKÇEN	KYA	Konya-KONYA_AB	20:10	21:20	B800a6	C10
PC183	KYA	Konya-KONYA_AB	SAW	Istanbul-SABİHA GÖKÇEN	21:40	22:50	B800a6	C10
PC188	SAW	Istanbul-SABİHA GÖKÇEN	KYA	Konya-KONYA_AB	6:00	7:15	B800a19	C06
PC189	KYA	Konya-KONYA_AB	SAW	Istanbul-SABİHA GÖKÇEN	8:10	9:20	B800a19	C06
PC190	SAW	Istanbul-SABİHA GÖKÇEN	VAN	Van-FERİT MELEN	10:00	12:05	B800a3	C13
PC191	VAN	Van-FERİT MELEN	SAW	Istanbul-SABİHA GÖKÇEN	12:35	14:45	B800a3	C13
PC194	SAW	Istanbul-SABİHA GÖKÇEN	EZS	Elazig-ELAZIG	11:50	13:25	B800a11	C06
PC195	EZS	Elazig-ELAZIG	SAW	Istanbul-SABİHA GÖKÇEN	13:45	15:25	B800a11	C06
PC196	SAW	Istanbul-SABİHA GÖKÇEN	DIY	Diyarbakır-DIYARBAKIR	12:40	14:20	B800a19	C03
PC197	DIY	Diyarbakır-DIYARBAKIR	SAW	Istanbul-SABİHA GÖKÇEN	15:15	17:00	B800a19	C03
PC202	AYT	Antalya-ANTALYA	DIY	Diyarbakır-DIYARBAKIR	23:00	0:35	B800a20	C14
PC203	DIY	Diyarbakır-DIYARBAKIR	AYT	Antalya-ANTALYA	1:30	3:00	B800a20	C14
PC2130	ADB	Izmir-ADNAN MENDERES	ESB	Ankara-ESENBOĞA	7:00	8:10	B800a9	C08
PC2131	ESB	Ankara-ESENBOĞA	ADB	Izmir-ADNAN MENDERES	8:35	9:50	B800a9	C08
PC2132	ADB	Izmir-ADNAN MENDERES	ESB	Ankara-ESENBOĞA	14:55	16:05	B800a2	C22
PC2133	ESB	Ankara-ESENBOĞA	ADB	Izmir-ADNAN MENDERES	16:30	17:45	B800a10	C38
PC2136	ADB	Izmir-ADNAN MENDERES	ESB	Ankara-ESENBOĞA	18:10	19:20	B800a10	C38
PC2137	ESB	Ankara-ESENBOĞA	ADB	Izmir-ADNAN MENDERES	19:45	21:00	B800a2	C22
PC2182	ADB	Izmir-ADNAN MENDERES	EZS	Elazig-ELAZIG	21:30	23:15	B800a2	C22
PC2183	EZS	Elazig-ELAZIG	ADB	Izmir-ADNAN MENDERES	23:45	1:45	B800a2	C22
PC2194	ADB	Izmir-ADNAN MENDERES	MQM	Mardin-Mardin	10:00	11:50	B800a2	C33
PC2195	MQM	Mardin-Mardin	ADB	Izmir-ADNAN MENDERES	12:40	14:35	B800a2	C33
PC2402	ADA	Adana-ADANA	AYT	Antalya-ANTALYA	8:25	9:25	A320b2	C41
PC2403	AYT	Antalya-ANTALYA	ADA	Adana-ADANA	9:55	10:55	B500a2	C41
PC2404	ADA	Adana-ADANA	AYT	Antalya-ANTALYA	19:50	20:50	B800a14	C28
PC2405	AYT	Antalya-ANTALYA	ADA	Adana-ADANA	21:15	22:15	A320b2	C28
PC2418	ADB	Izmir-ADNAN MENDERES	ASR	Kayseri-ERKİLET	15:20	16:45	B800a21	C40
PC2419	ASR	Kayseri-ERKİLET	ADB	Izmir-ADNAN MENDERES	17:30	19:05	B800a21	C40
PC2426	TZX	Trabzon-TRABZON	ADA	Adana-ADANA	13:50	15:15	B500a1	C41
PC2427	ADA	Adana-ADANA	TZX	Trabzon-TRABZON	12:00	13:25	B500a2	C41
PC2466	ESB	Ankara-ESENBOĞA	BJV	Bodrum-BODRUM MİLAS	19:45	21:00	B800a10	C38
PC2467	BJV	Bodrum-BODRUM MİLAS	ESB	Ankara-ESENBOĞA	21:20	22:30	B800a10	C38
PC250	SAW	Istanbul-SABİHA GÖKÇEN	VAS	Sivas-SIVAS	6:20	7:35	B800a23	C04
PC251	VAS	Sivas-SIVAS	SAW	Istanbul-SABİHA GÖKÇEN	8:30	9:50	B800a23	C04
PC2810	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	7:15	8:15	B800a21	C20
PC2811	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	9:10	10:10	B800a21	C20
PC2812	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	10:40	11:40	B800a9	C20
PC2813	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	12:10	13:15	B800a9	C20

Table A.8 Domestic flight schedule, initial crew schedule and initial aircraft-flight assignments of Pegasus Airlines (cont.)

Flight No	Origin		Destination		Departure	Arrival	Aircraft	Crew
PC2814	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	14:55	15:55	B800a9	C08
PC2815	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	16:40	17:40	B800a9	C08
PC2816	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	18:15	19:15	B800a24	C12
PC2817	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	20:00	21:00	B800a24	C12
PC2818	ADB	Izmir-ADNAN MENDERES	IST	İstanbul-Atatürk	21:40	22:40	B800a24	C40
PC2819	IST	İstanbul-Atatürk	ADB	Izmir-ADNAN MENDERES	0:15	1:20	B800a24	C40
PC2892	ADB	Izmir-ADNAN MENDERES	ADA	Adana-ADANA	6:35	8:00	A320b2	C12
PC2893	ADA	Adana-ADANA	ADB	Izmir-ADNAN MENDERES	15:45	17:15	B500a1	C12
PC2894	ADB	Izmir-ADNAN MENDERES	ADA	Adana-ADANA	17:55	19:20	B800a14	C28
PC2899	ADA	Adana-ADANA	ADB	Izmir-ADNAN MENDERES	22:45	0:15	A320b2	C28
PC4114	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	21:35	22:40	B800a12	C02
PC4115	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	23:05	0:05	B800a12	C02
PC4119	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	6:45	7:45	B800a11	C33
PC4120	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	8:15	9:15	A320b1	C33
PC4122	SAW	Istanbul-SABİHA GÖKÇEN	ADB	Izmir-ADNAN MENDERES	16:00	17:05	B800a14	C10
PC4123	ADB	Izmir-ADNAN MENDERES	SAW	Istanbul-SABİHA GÖKÇEN	17:30	18:30	A320b1	C10
PC4126	SAW	Istanbul-SABİHA GÖKÇEN	ADA	Adana-ADANA	8:15	9:35	B500a3	C31
PC4127	ADA	Adana-ADANA	SAW	Istanbul-SABİHA GÖKÇEN	9:55	11:20	B500a3	C31
PC4130	SAW	Istanbul-SABİHA GÖKÇEN	SZF	Samsun-CARSAMBA	8:10	9:30	B800a11	C19
PC4131	SZF	Samsun-CARSAMBA	SAW	Istanbul-SABİHA GÖKÇEN	9:55	11:15	B800a11	C19
PC4136	SAW	Istanbul-SABİHA GÖKÇEN	TZX	Trabzon-TRABZON	12:35	14:15	B800a7	C19
PC4137	TZX	Trabzon-TRABZON	SAW	Istanbul-SABİHA GÖKÇEN	14:40	16:25	B500a2	C18
PC4138	SAW	Istanbul-SABİHA GÖKÇEN	TZX	Trabzon-TRABZON	10:30	12:10	B500a1	C04
PC4139	TZX	Trabzon-TRABZON	SAW	Istanbul-SABİHA GÖKÇEN	12:40	14:25	B800a13	C04
PC4140	SAW	Istanbul-SABİHA GÖKÇEN	AYT	Antalya-ANTALYA	15:20	16:30	A320b2	C15
PC4141	AYT	Antalya-ANTALYA	SAW	Istanbul-SABİHA GÖKÇEN	16:55	18:05	B800a20	C27
PC4146	SAW	Istanbul-SABİHA GÖKÇEN	AYT	Antalya-ANTALYA	22:40	23:50	A319b3	C27
PC4147	AYT	Antalya-ANTALYA	SAW	Istanbul-SABİHA GÖKÇEN	21:00	22:10	A319b3	C15
PC4152	SAW	Istanbul-SABİHA GÖKÇEN	SZF	Samsun-CARSAMBA	15:15	16:35	B800a13	C39
PC4153	SZF	Samsun-CARSAMBA	SAW	Istanbul-SABİHA GÖKÇEN	17:00	18:20	B800a13	C39
PC4158	SAW	Istanbul-SABİHA GÖKÇEN	GZT	Gaziantep-GAZIANTEP	7:45	9:15	B800a25	C35
PC4159	GZT	Gaziantep-GAZIANTEP	SAW	Istanbul-SABİHA GÖKÇEN	9:55	11:30	B800a25	C35
PC4162	SAW	Istanbul-SABİHA GÖKÇEN	ASR	Kayseri-ERKİLET	18:45	20:00	B400a2	C21
PC4163	ASR	Kayseri-ERKİLET	SAW	Istanbul-SABİHA GÖKÇEN	20:20	21:35	B400a2	C21
PC4170	SAW	Istanbul-SABİHA GÖKÇEN	BJV	Bodrum-BODRUM MİLAS	16:00	17:10	B800a22	C11
PC4171	BJV	Bodrum-BODRUM MİLAS	SAW	Istanbul-SABİHA GÖKÇEN	17:30	18:40	B800a22	C11
PC4172	SAW	Istanbul-SABİHA GÖKÇEN	BJV	Bodrum-BODRUM MİLAS	21:45	22:55	B800a8	C36
PC4173	BJV	Bodrum-BODRUM MİLAS	SAW	Istanbul-SABİHA GÖKÇEN	23:15	0:25	B800a8	C36
PC4174	SAW	Istanbul-SABİHA GÖKÇEN	BJV	Bodrum-BODRUM MİLAS	12:00	13:10	A319b2	C26
PC4175	BJV	Bodrum-BODRUM MİLAS	SAW	Istanbul-SABİHA GÖKÇEN	13:30	14:40	A319b2	C26
PC4178	SAW	Istanbul-SABİHA GÖKÇEN	DLM	Dalaman-DALAMAN	20:05	21:20	B800a17	C32
PC4179	DLM	Dalaman-DALAMAN	SAW	Istanbul-SABİHA GÖKÇEN	21:45	22:55	B800a17	C32
PC4180	SAW	Istanbul-SABİHA GÖKÇEN	HTY	Hatay-HATAY	5:40	7:15	B800a6	C03
PC4181	HTY	Hatay-HATAY	SAW	Istanbul-SABİHA GÖKÇEN	8:00	9:35	B800a6	C03
PC4182	SAW	Istanbul-SABİHA GÖKÇEN	DLM	Dalaman-DALAMAN	18:30	19:45	B500a3	C30
PC4183	DLM	Dalaman-DALAMAN	SAW	Istanbul-SABİHA GÖKÇEN	20:45	21:55	B500a3	C30
PC4188	SAW	Istanbul-SABİHA GÖKÇEN	KYA	Konya-KONYA_AB	18:55	20:05	A319b1	C39
PC4189	KYA	Konya-KONYA_AB	SAW	Istanbul-SABİHA GÖKÇEN	20:45	21:55	A319b1	C39
PC4194	SAW	Istanbul-SABİHA GÖKÇEN	EZS	Elazig-ELAZIG	14:00	15:35	A319b1	C32
PC4195	EZS	Elazig-ELAZIG	SAW	Istanbul-SABİHA GÖKÇEN	16:30	18:10	A319b1	C32
PC4477	DLM	Dalaman-DALAMAN	SAW	Istanbul-SABİHA GÖKÇEN	22:00	23:10	B400a1	C31
PC4478	SAW	Istanbul-SABİHA GÖKÇEN	DLM	Dalaman-DALAMAN	20:15	21:30	B400a1	C31

APPENDIX B

ILOG OPTIMIZATION SUITE COMPONENTS

The ILOG Optimization Suite applies to long-term planning, scheduling, routing, configuration, as well as other resource allocation problems. The ILOG Optimization Suite product line —ILOG CPLEX, ILOG Solver, ILOG Scheduler, ILOG Dispatcher, ILOG Configurator, and ILOG OPL Studio—is fully integrated and allows the management of resource optimization problems from long-term capacity planning to short-term operations. The ILOG optimization software provides core engines for applying constraint programming and mathematical programming to a wide variety of problems. Additionally, the ILOG Optimization Suite enables the user to apply optimization through the use of the vertical engine extensions for constraint-based scheduling, technician dispatching, vehicle routing, and product and service configuration. Further, the ILOG Optimization Suite product line provides a complete modeling environment to support both constraint programming and mathematical programming applications, enabling the user to develop and deploy high level optimization applications without detailed knowledge of computer programming.

Core Engines of ILOG Optimization Suite

1. ILOG CPLEX: ILOG CPLEX provides powerful C and C++ libraries of fundamental algorithms for operations research and mathematical programming professionals. This library includes ILOG's Simplex, Barrier, and Mixed Integer solvers for linear, integer, and quadratic programming. CPLEX also provides easy to use C++ modeling objects that allow the expression of linear and integer programs in a simplified form directly related to their algebraic models.

2. ILOG Solver: The ILOG Solver engine provides the foundation of the ILOG Optimization Suite. ILOG Solver is one of the core C++ libraries of the ILOG Optimization Suite and implements the basic engine for constraint-based optimization. It can solve highly combinatorial real-world applications that are impractical to solve with traditional mathematical programming methods. This high-

performance constraint-programming engine can be used alone, with CPLEX, or as a foundation for the vertical engine extensions. ILOG Solver provides all the functions needed to model and solve a very wide range of resource allocation and decision problems.

3. ILOG Concert Technology : ILOG Solver and ILOG CPLEX are supplied with ILOG Concert Technology, a set of C++ modeling objects that can be used to represent a wide range of optimization problems. This framework facilitates the comparison as well as integration of constraint programming and mathematical programming technologies.

Vertical Engine Extensions

Besides the core engines, ILOG also has vertical engine extensions. The vertical engine extensions to ILOG Solver allow developers to take advantage of the power of constraint programming with extensions for specific problem domains. These extensions allow complete access to the underlying power of ILOG Solver. *ILOG Scheduler which* contains specialized modeling and algorithmic enhancements for scheduling applications, including constraint-based scheduling algorithms for handling the time sequencing of resource constrained activities. *ILOG Dispatcher* which is designed for building vehicle routing and personnel dispatching applications, including the assignment of technicians to customer locations for after-sales service and *ILOG Configurator* which is a C++ library that provides faster and easier development of high-performance configuration engines.

ILOG OPL Studio 3.7

ILOG OPL Studio 3.7 is an integrated development environment for combinatorial optimization applications. It delivers the tools needed to quickly and easily to create business optimization models. From the OPL language, users have Access to the powerful algorithms of ILOG CPLEX, ILOG Solver and ILOG Scheduler. After developing a model with OPL, the OPL Component Libraries can be used to integrate the model into an application developed in C++, Visual Basic, or Java.

OPTIMIZATION PROGRAMMING LANGUAGE (OPL)

OPL (Optimization Programming Language) is the core of ILOG OPL Studio. OPL is a modeling language for combinatorial optimization that may simplify the optimization problems substantially. OPL lets the user state optimization problems quickly, accurately and naturally, without the low-level complexities of ordinary programming languages. OPL is unique in its support of both mathematical programming and constraint programming which allows the expression of an extremely wide range of business optimization models. OPL is strongly typed, which simplifies model debugging compared to other mathematical programming modeling languages OPL was motivated by modeling languages such as AMPL and GAMS that provide computer equivalents to traditional algebraic notation. It provides similar support for modeling linear and integer programs and but OPL adds several new dimensions to modeling language beyond the traditional support for linear and integer programming. The new dimensions of the OPL are follows:

- It supports for constraint programming. The essence of constraint programming is a two-level architecture integrating a constraint and a programming component. The constraint component provides the basic operations of the architecture and consists of a system that reasons about fundamental properties of constraint systems. Programming language component specifies how to combine the basic operations, often in nondeterministic ways, since algorithms for searching the solution space are so fundamental in combinatorial optimization.
- It supports for scheduling and resource allocation applications. OPL provides novel modeling concepts such as activities and resources and provides access to special-purpose algorithms.
- OPL improves the expressiveness of traditional modeling languages by offering new concepts such as higher order constraints, logical combinations of constraints and many other new modeling tools.

- OPL provides an abstract connection to databases, allowing integration of external data stored in ODBC, Oracle, Informix and other databases into an optimization application, and to export solutions to those same databases. Direct connections to Microsoft Excel spreadsheets allow two-dimensional data and solutions to be rapidly integrated into an OPL application.
- OPL Script provides a powerful high-level procedural language for solving sequences of OPL models and developing sophisticated solution strategies.

The ILOG Optimization Suite has been successfully implemented in very diverse projects. The types of problems to which it can be applied and some of the solution strategies are as follows:

Mathematical Programming

As one of the core engines of the ILOG Optimization Suite, ILOG CPLEX provides unparalleled algorithms to handle linear and mixed-integer programming problems. The number of such successes has exponentially increased due to the application of ILOG CPLEX for solving mathematical programming problems.

Mathematical programming algorithms are widely used to solve linear problems such as liquid blending and production planning. They are unmatched at computing optimal solutions, computing them rapidly with powerful algorithms.

1. Linear Programming Strategies

ILOG CPLEX is designed to solve the majority of linear programming problems using pre-determined settings for the simplex and barrier algorithms. However, many of these parameters can be fine tuned to meet the needs of a user's specific problem. For example, many highly degenerate problems solve faster using dual simplex as opposed to primal simplex, and many very large problems are better suited for optimizing using barrier. CPLEX gives the user the ability to make these choices.

Within each of these algorithms for solving linear programs, CPLEX also allows the user to fine-tune the solution approach by providing, for example, an advanced

basis (for simplex-based algorithms), upper and lower bound limits, and a convergence tolerance (for barrier).

2. Mixed-Integer Programming Strategies

ILOG CPLEX uses branch and bound to solve mixed-integer programming problems. In the branch-and-bound method, a series of linear programs is solved, causing a branch-and-bound tree to be created. The path that CPLEX takes through this tree can be determined by a number of user inputs.

For example, at each node in the branch-and-bound tree, CPLEX can either delve deeper in the tree or backtrack. The user can determine the setting of the backtrack parameter or leave it set at its default value. And, once CPLEX has backtracked, there are typically a large number of unexplored nodes from which to choose. The node select parameter is used to set the rule for selecting the next node to process when backtracking.

Users also have the opportunity to select the type of algorithm used to solve the linear programs generated in the branch-and-bound search tree. The default strategy is to use the dual-simplex method, but some problems are solved more efficiently using primal simplex or barrier.

Cutting planes can be very effective in improving the performance of CPLEX for solving mixed-integer programming problems. Only certain problems, however, have the underlying structure to use cutting planes, and CPLEX automatically scans the given problem to see if the underlying structure exists. If a user then determines that his or her problem will not be aided by the use of cuts, this option can be turned off. On the other hand, the assessment may be incorrect, and the user can set parameters that allow CPLEX to generate the cuts.

Constraint Programming

ILOG Solver is ILOG's core constraint-programming optimization engine. The result of more than ten years of research and development, it can solve highly combinatorial real-world problems that are impractical with conventional methods.

The more recent innovation of constraint programming is a highly effective technology using domain reduction and constraint propagation to efficiently solve problems that are highly combinatorial with highly logical content. These problems are usually difficulty or impossible to represent with linear expressions. The constraint programming methodology allows the natural expression of very complex relationships including logical expressions.

Constraint programming uses information contained in the problem to "prune" the search space in order to more rapidly identify feasible solutions. The domains of the variables are continually updated during the search for a solution, providing very useful information for refining the search strategy. Constraint programming is ideally suited for operational problems because these problems require fast, feasible answers and because operational problems often require logical constraints to represent the complexities inherent in such problems.

Users can also guide the search process with their own knowledge of the problem.

1. Integration of Operations Research Algorithms

A very attractive property of ILOG Solver is that it easily integrates specialized algorithms for solving a given problem. In other words, constraint-based optimization is a framework for cooperating problem-solving algorithms. Algorithms are implemented as constraint solvers, and the custom algorithms and standard solvers communicate via the variables using the constraint propagation and domain reduction mechanism. For scheduling problems involving time and sequencing, the edge-finder algorithm is implemented in the finite-capacity resource constraints of ILOG Scheduler. This algorithm is one of the most successful operations research algorithms for rapidly updating time windows of activities subject to resource constraints.

2. User-defined global constraints

Global constraints, that is, a single constraint that expresses multiple conditions, can be defined in ILOG Solver.

An example of this is a resource allocation problem such as assigning cashiers to work areas (i.e., cash registers) in a department store. Ten cash registers each require one person to be assigned. In ILOG Solver, the model for this problem includes ten decision variables, one for each person. The possible values for these variables are the cash registers. The only constraint in the problem is that no pair of variables can be assigned the same cash register. This is simple and intuitive. This restriction could be expressed by stating a different inequality constraint for each variable pair. However, if n is the number of people, n^2 constraints are required. Indeed, if there were 1,000 people in a chain of stores, it would be necessary to create 1 million pair-wise inequalities. An alternative provided by ILOG Solver is to use only one global constraint (the IloAllDiff constraint) and state it over all the variables. The space required by the constraint is linear with respect to n.

Hybrid Approaches

Most real-world problems have both planning and operational aspects, making them more naturally suited for representations that include both linear and logical expressions. This requires the combined use of mathematical programming and constraint programming approaches. The ILOG Optimization Suite is unique in its ability to provide hybrid optimization approaches that seamlessly blend mathematical programming and constraint programming technologies together in one application. A typical example is crew scheduling where constraint programming is used to generate a large set of feasible pairings (a highly logical and combinatorial problem), and mathematical programming is used to quickly select the optimal combinations.

Powerful Modeling

ILOG OPL Studio is a revolutionary interactive environment that provides a pre-built library of commonly used optimization models that allow the user to “jump start” application development. After OPL Studio has been used to rapidly prototype the application, the OPL Component Libraries let the user move directly to the application deployment using ILOG Solver, ILOG CPLEX, and ILOG Scheduler.

ILOG Solver and ILOG CPLEX share ILOG Concert Technology, a set of C++ modeling objects that make it easy to declare a problem's characteristics and try alternative solution strategies. This common framework provides true model and algorithm separation, allowing for efficiencies in development time as well as the memory required to solve problems.

Interactive and Reactive Capabilities

The ability to define constraints and variables and find a solution is not enough for a lot of practical applications. Users may need to intensively interact with the planning applications. They would need, for example, to generate several plans, explore scenarios, run simulations and what-if analyses, adding or relaxing constraints. In other applications, *reactivity* is a key issue. This means that the planning applications must be able to update plans according to new data, new constraints, or new priorities.

To satisfy these requirements, the ILOG Optimization Suite provides a way to *manage* constraint models. ILOG Solver and ILOG CPLEX provide object models which allow adding and removing constraints, completely or partially storing the current solution, and restoring the previously stored partial or complete solution.

Moreover, another benefit of the ILOG Solver and ILOG CPLEX object models is the possibility of writing advanced optimization algorithms and heuristics that rely on a limited search in a subtree, iterations on local search, problem decomposition, etc. These allow applications to be easily adapted to scenarios with changing data.

Exploiting Problem Knowledge

Using the ILOG Optimization Suite, the search can itself be made more efficient by exploiting knowledge about the problem. For example, the order in which the nodes are explored in the tree may be very important.

Consider the timetabling application for mainframe operators at the Banque Bruxelles Lambert. Some parts of the year are more difficult to schedule than others. Normally, many engineers want to take their vacations in the summer and around

Christmas. It seems reasonable, therefore, to assign the shifts corresponding to the Christmas week first, as this is the week in which the problem is the most difficult to solve. This kind of information is called “strategic knowledge,” since it deals with the way the problem should be solved. Strategic knowledge is quite easy to use with the ILOG Optimization Suite. In fact, in the search algorithm, ILOG Solver enables the developer to control the order in which the variables are selected. For ILOG CPLEX, the order in which the variables are chosen during branch and bound can also be specified ahead of time.

This principle can be applied to improve performance on very complex problems.

APPENDIX C

OPL MODEL OF AIRCRAFT RECOVERY PROBLEM (MODEL A1)

1) Problem statement

```
==================
range Boolean 0..1;
enum Flights ...;
enum Aircrafts ...;
Aircrafts shortage= MD88b;
int start=735;
int finish=1005;
struct Connection { Flights i; Flights j; };
struct Route { Aircrafts a; Connection c; };

{Route} Routes = ...;
{Connection} Connections = { c | <a,c> in Routes };
Boolean trans[Routes]=...;
int+ cost[Routes] = ...;
int dept[Flights]=...;
int arr[Flights]=...;
int org[Flights]=...;
int dest[Flights]=...;
{Connection} CP[a in Aircrafts] = { c | <a,c> in Routes };

var Boolean reroute[Routes];
var Boolean cancel[Flights];
var float+ nSeq[Aircrafts];

minimize
  sum(l in Routes) cost[l] * reroute[l] +sum(i in Flights) 1000 *cancel[i]

subject to{
  forall (a in Aircrafts, i in Flights: i=source)
        sum (<i,j> in CP[a]) reroute[< a,<i,j> >]=1;
  forall (a in Aircrafts, j in Flights:j=sink)
        sum (<i,j> in CP[a]) reroute[< a,<i,j> >]=1;
  forall(a in Aircrafts, <i,j> in CP[a]:a=shortage & dept[i]>=start & dept[j]<finish)
        reroute[<a,<i,j>>]=0;
  forall(a in Aircrafts, <i,j> in CP[a]: a<>shortage & dept[j]<start)
        reroute[<a,<i,j>>]=trans[<a,<i,j>>];
  forall(a in Aircrafts, <i,j> in CP[a]: dept[i]>= finish)
```

```
        reroute[<a,<i,j>>]=trans[<a,<i,j>>];
  forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
        sum (<i,j> in CP[a]) reroute[< a,<i,j> >]
        - sum (<k,i> in CP[a]) reroute[< a,<k,i> >]=0;
  forall (a in Aircrafts,<i,j> in CP[a]:dept[i]>=start)
        reroute[<a,<i,j>>]*(arr[i]+45-dept[j])<=0;
  forall(i in Flights : i<>source & i<> sink)
        cancel[i]+sum (<i,j> in Connections) sum(a in Aircrafts)
        reroute[<a,<i,j>>]=1;
  forall (a in Aircrafts :a<>shortage)
         sum(<i,j> in CP[a]) reroute[<a,<i,j>>]<=7;
  forall (a in Aircrafts :a<>shortage)
        sum(<i,j> in CP[a]) reroute[<a,<i,j>>]>=5;
  forall(a in Aircrafts)
        nSeq[a]=sum(<i,j> in CP[a])reroute[<a,<i,j>>];
};
display (l in Routes : reroute[l] > 0) <reroute[l]>;
display (a in Aircrafts) <nSeq[a]>;
display (i in Flights : cancel[i] > 0) <cancel[i]>;
```

2) Results

==========

Optimal Solution with Objective Value: 16035

```
<reroute[#<a:MD88a,c:#<i:source,j:OHY034>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY018,j:OHY019>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY034,j:OHY035>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY035,j:OHY018>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY019,j:sink>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:source,j:OHY066>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY066,j:OHY067>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY067,j:OHY058>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY058,j:sink>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:source,j:OHY010>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY010,j:OHY011>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY011,j:OHY054>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY054,j:OHY055>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY055,j:sink>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:source,j:OHY027>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY027,j:OHY042>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY042,j:OHY043>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY043,j:OHY026>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY026,j:sink>#>#]> = <1>
```

<reroute[#<a:MD88e,c:#<i:source,j:OHY059>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY036,j:OHY037>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY059,j:OHY074>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY074,j:OHY075>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY075,j:OHY036>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY037,j:sink>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:source,j:OHY022>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY016,j:OHY017>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY022,j:OHY023>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY023,j:OHY016>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY017,j:sink>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:source,j:OHY028>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY024,j:OHY025>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY028,j:OHY029>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY029,j:OHY024>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY025,j:sink>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:source,j:OHY050>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY038,j:OHY039>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY039,j:OHY086>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY050,j:OHY051>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY051,j:OHY038>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY086,j:OHY087>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY087,j:sink>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:source,j:OHY080>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY080,j:OHY081>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY081,j:OHY062>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY062,j:OHY063>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY063,j:sink>#>#]> = <1>

APPENDIX D

OPL MODEL OF AIRCRAFT RECOVERY PROBLEM (MODEL A2)

1) Problem statement

```
====================
range Boolean 0..1;
enum Flights ...;
enum Aircrafts ...;
Aircrafts shortage1=MD83b;
Aircrafts shortage2=MD88c;
int start1=570;
int start2=750;
int finish1=720;
int finish2=1050;
struct Connection { Flights i; Flights j; };
struct Route { Aircrafts a; Connection c; };
{Route} Routes = ...;
{Connection} Connections = { c | <a,c> in Routes };
Boolean trans[Routes]=...;
int+ cost[Routes] = ...;
int dept[Flights]=...;
int arr[Flights]=...;
int dept30[Flights]=...;
int arr30[Flights]=...;
int arr60[Flights]=...;
int dept60[Flights]=...;
int org[Flights]=...;
int dest[Flights]=...;
{Connection} CP[a in Aircrafts] = { c | <a,c> in Routes };
var Boolean reroute[Routes];
var Boolean reroute30_1[Routes];
var Boolean reroute30_2[Routes];
var Boolean reroute30_12[Routes];
var Boolean reroute60_1[Routes];
var Boolean reroute60_2[Routes];
var Boolean reroute60_12[Routes];
var Boolean reroute30_60[Routes];
var Boolean reroute60_30[Routes];
var Boolean cancel[Flights];
var float+ nSeq[Aircrafts];
```

```
minimize
 sum(l in Routes) (cost[l]*reroute[l]+(cost[l]+30)*(reroute30_1[l]+reroute30_2[l])+
       (cost[l]+60)*(reroute30_12[l]+reroute60_1[l]+reroute60_2[l])+
       (cost[l]+90)*(reroute30_60[l]+reroute60_30[l])+
       (cost[l]+120)*reroute60_12[l])+
       sum(i in Flights) 1000 *cancel[i]
subject to{
   forall (a in Aircrafts, i in Flights: i=source)
          sum (<i,j> in CP[a])
          (reroute[< a,<i,j>>]+ reroute30_1[< a,<i,j>>]+ reroute30_2[< a,<i,j>>]+
          reroute30_12[< a,<i,j>>]+ reroute60_1[< a,<i,j>>]+ reroute60_2[< a,<i,j>>]+
          reroute60_12[< a,<i,j>>]+
          reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=1;
   forall (a in Aircrafts, j in Flights:j=sink)
          sum (<i,j> in CP[a])
          (reroute[< a,<i,j>>]+reroute30_1[< a,<i,j>>]+ reroute30_2[< a,<i,j>>]+
          reroute30_12[< a,<i,j>>]+ reroute60_1[< a,<i,j>>]+ reroute60_2[< a,<i,j>>]+
          reroute60_12[< a,<i,j>>]+ reroute30_60[<a,<i,j>>]+
          reroute60_30[<a,<i,j>>])=1;
forall(a in Aircrafts, <i,j> in CP[a]:a=shortage1 & dept[i]>=start1 & dept[j]<finish1)
          (reroute[< a,<i,j> >]+ reroute30_1[< a,<i,j> >]+ reroute30_2[< a,<i,j> >]+
          reroute30_12[< a,<i,j> >]+ reroute60_1[< a,<i,j> >]+ reroute60_2[< a,<i,j>
          >]+ reroute60_12[< a,<i,j> >]+ reroute30_60[<a,<i,j> >]+
          reroute60_30[<a,<i,j> >]) =0;
forall(a in Aircrafts, <i,j> in CP[a]:a=shortage2 & dept[i]>=start2 & dept[j]<finish2)
           (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j> >]+
          reroute30_12[< a,<i,j> >]+ reroute60_1[< a,<i,j> >]+ reroute60_2[< a,<i,j>
          >]+ reroute60_12[< a,<i,j> >]+ reroute30_60[<a,<i,j> >]+
          reroute60_30[<a,<i,j> >]) =0;
forall(a in Aircrafts, <i,j> in CP[a]: dept[j]<start1)
          reroute[< a,<i,j> >]=trans[<a,<i,j>>];
forall(a in Aircrafts, <i,j> in CP[a]: dept[i]>= finish2)
           reroute[< a,<i,j> >]=trans[<a,<i,j>>];
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
          sum (<k,i> in CP[a])
          (reroute[<a,<k,i>>]+reroute30_1[<a,<k,i>>]+reroute60_1[<a,<k,i>>])- sum
          (<i,j> in CP[a])
          (reroute30_2[<a,<i,j>>]+reroute60_2[<a,<i,j>>]+reroute[<a,<i,j>>])=0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
          sum (<k,i> in CP[a])
```

```
        (reroute30_2[<a,<k,i>>]+reroute30_12[<a,<k,i>>])- sum (<i,j> in CP[a])
        (reroute30_1[<a,<i,j>>]+reroute30_12[<a,<i,j>>]+reroute30_60[<a,<i,j>>])=
        0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
        sum (<k,i> in CP[a])
        (reroute60_2[<a,<k,i>>] + reroute60_12[<a,<k,i>>])- sum (<i,j> in CP[a])
        (reroute60_1[<a,<i,j>>]+reroute60_12[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=
        0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
         sum (<k,i> in CP[a]) reroute60_30[<a,<k,i>>]- sum (<i,j> in CP[a])
        (reroute30_1[<a,<i,j>>]+reroute30_12[<a,<i,j>>]+reroute30_60[<a,<i,j>>])=
        0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
         sum (<k,i> in CP[a]) reroute30_60[<a,<k,i>>]- sum (<i,j> in CP[a])
        (reroute60_1[<a,<i,j>>]+reroute60_12[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=
        0;
forall( a in Aircrafts, <i,j> in CP[a])
        reroute[< a,<i,j> >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j>
        >]+reroute30_12[< a,<i,j> >]+reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j>
        >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>]<=1;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
         sum (<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
         reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])
          - sum (<k,i> in CP[a]) (reroute[< a,<k,i> >]+reroute30_1[< a,<k,i>
        >]+reroute30_2[< a,<k,i> >]+reroute30_12[< a,<k,i> >]+
         reroute60_1[< a,<k,i> >]+reroute60_2[< a,<k,i> >]+reroute60_12[< a,<k,i>
        >]+reroute30_60[<a,<k,i>>]+reroute60_30[<a,<k,i>>])=0;
forall (a in Aircrafts,<i,j> in CP[a]:dept[i]>=start1)
         (reroute[<a,<i,j>>]*(arr[i]+45-dept[j])+
         reroute30_1[<a,<i,j>>]*(arr30[i]+45-dept[j])+
         reroute30_12[<a,<i,j>>]*(arr30[i]+45-dept30[j])+
         reroute30_2[<a,<i,j>>]*(arr[i]+45-dept30[j])+
         reroute60_1[<a,<i,j>>]*(arr60[i]+45-dept[j])+
         reroute60_2[<a,<i,j>>]*(arr[i]+45-dept60[j])+
         reroute60_12[<a,<i,j>>]*(arr60[i]+45-dept60[j])+
         reroute30_60[<a,<i,j>>]*(arr30[i]+45-dept60[j])+
        reroute60_30[<a,<i,j>>]*(arr60[i]+45-dept30[j]))<=0;
forall(i in Flights : i<>source & i<> sink)
```

```
        cancel[i]+sum (<i,j> in Connections) sum(a in Aircrafts) (reroute[< a,<i,j>
        >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j>
        >]+
        reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=1;
forall (a in Aircrafts :a<>shortage1 & a<>shortage2)
         sum(<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
          reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])<=7;
forall (a in Aircrafts :a<>shortage1 & a<>shortage2)
         sum(<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
          reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])>=5;
forall(a in Aircrafts)
        nSeq[a]=sum(<i,j> in CP[a])(reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
         reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>]);
};
display (l in Routes : reroute[l] > 0) <reroute[l]>;
display (l in Routes : reroute30_1[l] > 0) <reroute30_1[l]>;
display (l in Routes : reroute30_2[l] > 0) <reroute30_2[l]>;
display (l in Routes : reroute30_12[l] > 0) <reroute30_12[l]>;
display (l in Routes : reroute60_1[l] > 0) <reroute60_1[l]>;
display (l in Routes : reroute60_2[l] > 0) <reroute60_2[l]>;
display (l in Routes : reroute60_12[l] > 0) <reroute60_12[l]>;
display (l in Routes : reroute30_60[l] > 0) <reroute30_60[l]>;
display (l in Routes : reroute60_30[l] > 0) <reroute60_30[l]>;
display (i in Flights : cancel[i]> 0) <cancel[i]>;
};
```

2) Results

==========

Optimal Solution with Objective Value: 16035

```
<reroute[#<a:MD88a,c:#<i:source,j:OHY034>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY018,j:OHY019>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY034,j:OHY035>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY035,j:OHY018>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY019,j:sink>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:source,j:OHY080>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY036,j:OHY037>#>#]> = <1>
```

<reroute[#<a:MD88b,c:#<i:OHY028,j:OHY029>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY029,j:OHY036>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY080,j:OHY081>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY081,j:OHY028>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY037,j:sink>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:source,j:OHY022>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY022,j:OHY023>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY023,j:sink>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:source,j:OHY050>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY050,j:OHY051>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY054,j:OHY055>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY051,j:OHY054>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY055,j:sink>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:source,j:OHY066>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY038,j:OHY039>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY039,j:OHY058>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY066,j:OHY067>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY067,j:OHY038>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY058,j:sink>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:source,j:OHY074>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY074,j:OHY075>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY075,j:OHY086>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY086,j:OHY087>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY087,j:sink>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:source,j:OHY027>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY024,j:OHY025>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY027,j:OHY062>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY062,j:OHY063>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY063,j:OHY024>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY025,j:sink>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:source,j:OHY010>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY010,j:OHY011>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY016,j:OHY017>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY011,j:OHY016>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY017,j:sink>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:source,j:OHY059>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY042,j:OHY043>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY043,j:OHY026>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY059,j:OHY042>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY026,j:sink>#>#]> = <1>

APPENDIX E

OPL MODEL OF AIRCRAFT RECOVERY PROBLEM WITH CREW CONSIDERATIONS (MODEL A3)

```
1) Problem statement
====================
range Boolean 0..1;
enum Flights ...;
enum Aircrafts ...;
Aircrafts shortage1=MD88e;
Aircrafts shortage2=MD88c;
int start1=430;
int start2=750;
int finish1=720;
int finish2=1050;

struct Connection { Flights i; Flights j; };
struct Route { Aircrafts a; Connection c; };
{Route} Routes = ...;
{Connection} Connections = { c | <a,c> in Routes };
Boolean trans[Routes]=...;

int+ cost[Routes] = ...;
int dept[Flights]=...;
int arr[Flights]=...;
int dept30[Flights]=...;
int arr30[Flights]=...;
int arr60[Flights]=...;
int dept60[Flights]=...;
int org[Flights]=...;
int dest[Flights]=...;
int crewno[Flights]=...;

{Connection} CP[a in Aircrafts] = { c | <a,c> in Routes };

var Boolean reroute[Routes];
var Boolean reroute30_1[Routes];
var Boolean reroute30_2[Routes];
var Boolean reroute30_12[Routes];
var Boolean reroute60_1[Routes];
var Boolean reroute60_2[Routes];
```

```
var Boolean reroute60_12[Routes];
var Boolean reroute30_60[Routes];
var Boolean reroute60_30[Routes];
var Boolean cancel[Flights];
var float+ nSeq[Aircrafts];
var float+ recrew[Flights];

minimize
sum(l in Routes)
        (cost[l]*reroute[l]+(cost[l]+30)*(reroute30_1[l]+reroute30_2[l])+(cost[l]+60)
        *(reroute30_12[l]+reroute60_1[l]+reroute60_2[l])+(cost[l]+90)*(reroute30_6
        0[l]+reroute60_30[l])+(cost[l]+120)*reroute60_12[l]) +sum(i in Flights)
        1000 *cancel[i]

subject to{
forall (a in Aircrafts, i in Flights: i=source)
        sum (<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
        reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=1;
forall (a in Aircrafts, j in Flights:j=sink)
        sum (<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
         reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=1;
forall(a in Aircrafts, <i,j> in CP[a]:a=shortage1 & dept[i]>=start1 & dept[j]<finish1)
        (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j>
       >]+reroute30_12[< a,<i,j> >]+
         reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
       >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=0;
forall(a in Aircrafts, <i,j> in CP[a]:a=shortage2 & dept[i]>=start2 & dept[j]<finish2)
         (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j>
       >]+reroute30_12[< a,<i,j> >]+
        reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
       >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=0;
forall(a in Aircrafts, <i,j> in CP[a]: dept[j]<start1)
         reroute[< a,<i,j> >]=trans[<a,<i,j>>];
forall(a in Aircrafts, <i,j> in CP[a]: dept[i]>= finish2)
        reroute[< a,<i,j> >]=trans[<a,<i,j>>];
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
        sum (<k,i> in CP[a])
        (reroute[<a,<k,i>>]+reroute30_1[<a,<k,i>>]+reroute60_1[<a,<k,i>>])- sum
```

```
        (<i,j> in CP[a])
        (reroute30_2[<a,<i,j>>]+reroute60_2[<a,<i,j>>]+reroute[<a,<i,j>>])=0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
        sum (<k,i> in CP[a]) (reroute30_2[<a,<k,i>>]+reroute30_12[<a,<k,i>>])-
        sum (<i,j> in CP[a])
        (reroute30_1[<a,<i,j>>]+reroute30_12[<a,<i,j>>]+reroute30_60[<a,<i,j>>])=
        0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
        sum (<k,i> in CP[a]) (reroute60_2[<a,<k,i>>] + reroute60_12[<a,<k,i>>])-
        sum (<i,j> in CP[a])
        (reroute60_1[<a,<i,j>>]+reroute60_12[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=
        0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
        sum (<k,i> in CP[a]) reroute60_30[<a,<k,i>>]- sum (<i,j> in CP[a])
        (reroute30_1[<a,<i,j>>]+reroute30_12[<a,<i,j>>]+reroute30_60[<a,<i,j>>])=
        0;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
        sum (<k,i> in CP[a]) reroute30_60[<a,<k,i>>]- sum (<i,j> in CP[a])
        (reroute60_1[<a,<i,j>>]+reroute60_12[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=
        0;
forall( a in Aircrafts, <i,j> in CP[a])
        reroute[< a,<i,j> >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j>
        >]+reroute30_12[< a,<i,j> >]+reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j>
        >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>]<=1;
forall (a in Aircrafts, i in Flights:  i<>sink & i<>source)
         sum (<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
         reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])
        - sum (<k,i> in CP[a]) (reroute[< a,<k,i> >]+reroute30_1[< a,<k,i>
        >]+reroute30_2[< a,<k,i> >]+reroute30_12[< a,<k,i> >]+
         reroute60_1[< a,<k,i> >]+reroute60_2[< a,<k,i> >]+reroute60_12[< a,<k,i>
        >]+reroute30_60[<a,<k,i>>]+reroute60_30[<a,<k,i>>])=0;
forall (a in Aircrafts,<i,j> in CP[a]:dept[i]>=start1)
        (reroute[<a,<i,j>>]*(arr[i]+45-dept[j])+
        reroute30_1[<a,<i,j>>]*(arr30[i]+45-dept[j])+
        reroute30_12[<a,<i,j>>]*(arr30[i]+45-dept30[j])+
        reroute30_2[<a,<i,j>>]*(arr[i]+45-dept30[j])+
        reroute60_1[<a,<i,j>>]*(arr60[i]+45-dept[j])+
        reroute60_2[<a,<i,j>>]*(arr[i]+45-dept60[j])+
        reroute60_12[<a,<i,j>>]*(arr60[i]+45-dept60[j])+
```

```
        reroute30_60[<a,<i,j>>]*(arr30[i]+45-dept60[j])+
        reroute60_30[<a,<i,j>>]*(arr60[i]+45-dept30[j]))<=0;
forall(a in Aircrafts, <i,j> in CP[a]:a=shortage1 & dept[i]>=start1 & dept[j]<finish1)
         recrew[j]=crewno[j]-1;
forall(a in Aircrafts, <i,j> in CP[a]:a=shortage2 & dept[i]>=start2 & dept[j]<finish2)
        {recrew[i]=crewno[i]-1;
        recrew[j]=crewno[j]-1;};
forall(a in Aircrafts, <i,j> in CP[a]: dept[j]<start1)
        {recrew[i]=crewno[i];
         recrew[j]=crewno[j];};
forall(a in Aircrafts, <i,j> in CP[a]: dept[i]>= finish2)
         {recrew[i]=crewno[i];
        recrew[j]=crewno[j];};
forall (a in Aircrafts, <i,j> in CP[a]:dept[i]>=finish1 & dept[j]<start2)
         {recrew[i]=crewno[i];
         recrew[j]=crewno[j];};
forall(i in Flights : i<>source & i<> sink)
        sum (a in Aircrafts, <i,j> in Connections: a<>shortage1 & a<>shortage2)
         (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j>
        >]+reroute30_12[< a,<i,j> >]+
         reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])<= recrew[i];
forall(i in Flights : i<>source & i<> sink)
        cancel[i]+sum (<i,j> in Connections) sum(a in Aircrafts) (reroute[< a,<i,j>
        >]+reroute30_1[< a,<i,j> >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j>
        >]+
        reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])=1;
forall (a in Aircrafts :a<>shortage1 & a<>shortage2)
        sum(<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
        reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])<=7;
forall (a in Aircrafts :a<>shortage1 & a<>shortage2)
         sum(<i,j> in CP[a]) (reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
        reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>])>=5;
forall(a in Aircrafts)
        nSeq[a]=sum(<i,j> in CP[a])(reroute[< a,<i,j> >]+reroute30_1[< a,<i,j>
        >]+reroute30_2[< a,<i,j> >]+reroute30_12[< a,<i,j> >]+
```

```
        reroute60_1[< a,<i,j> >]+reroute60_2[< a,<i,j> >]+reroute60_12[< a,<i,j>
        >]+reroute30_60[<a,<i,j>>]+reroute60_30[<a,<i,j>>]);
};
search{
  forall(a in Aircrafts)
    tryall(<i,j> in Connections) //ordered by increasing dsize(reroute[<a,<i,j>>])
      reroute[<a,<i,j>>]=trans[<a,<i,j>>];
};
display (l in Routes : reroute[l] > 0) <reroute[l]>;
display (l in Routes : reroute30_1[l] > 0) <reroute30_1[l]>;
display (l in Routes : reroute30_2[l] > 0) <reroute30_2[l]>;
display (l in Routes : reroute30_12[l] > 0) <reroute30_12[l]>;
display (l in Routes : reroute60_1[l] > 0) <reroute60_1[l]>;
display (l in Routes : reroute60_2[l] > 0) <reroute60_2[l]>;
display (l in Routes : reroute60_12[l] > 0) <reroute60_12[l]>;
display (l in Routes : reroute30_60[l] > 0) <reroute30_60[l]>;
display (l in Routes : reroute60_30[l] > 0) <reroute60_30[l]>;
display (a in Aircrafts) <nSeq[a]>;
display (i in Flights : cancel[i] > 0) <cancel[i]>;
```

2) Results

```
==========
Optimal Solution with Objective Value: 18825
<reroute[#<a:MD88a,c:#<i:source,j:OHY034>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY018,j:OHY019>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY034,j:OHY035>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY035,j:OHY018>#>#]> = <1>
<reroute[#<a:MD88a,c:#<i:OHY019,j:sink>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:source,j:OHY080>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY036,j:OHY037>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY080,j:OHY081>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY081,j:OHY036>#>#]> = <1>
<reroute[#<a:MD88b,c:#<i:OHY037,j:sink>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:source,j:OHY022>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY022,j:OHY023>#>#]> = <1>
<reroute[#<a:MD88c,c:#<i:OHY023,j:sink>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:source,j:OHY059>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY042,j:OHY043>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY043,j:OHY054>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY054,j:OHY055>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY059,j:OHY042>#>#]> = <1>
<reroute[#<a:MD88d,c:#<i:OHY055,j:sink>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:source,j:OHY062>#>#]> = <1>
```

<reroute[#<a:MD88e,c:#<i:OHY062,j:OHY063>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY063,j:OHY058>#>#]> = <1>
<reroute[#<a:MD88e,c:#<i:OHY058,j:sink>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:source,j:OHY050>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY038,j:OHY039>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY050,j:OHY051>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY051,j:OHY038>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY039,j:OHY086>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY086,j:OHY087>#>#]> = <1>
<reroute[#<a:MD83a,c:#<i:OHY087,j:sink>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:source,j:OHY028>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY024,j:OHY025>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY028,j:OHY029>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY029,j:OHY024>#>#]> = <1>
<reroute[#<a:MD83b,c:#<i:OHY025,j:sink>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:source,j:OHY010>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY010,j:OHY011>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY016,j:OHY017>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY011,j:OHY016>#>#]> = <1>
<reroute[#<a:MD83c,c:#<i:OHY017,j:sink>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:source,j:OHY027>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY027,j:OHY074>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY074,j:OHY075>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY075,j:OHY026>#>#]> = <1>
<reroute[#<a:MD83d,c:#<i:OHY026,j:sink>#>#]> = <1>

APPENDIX F
OPL CODES FOR MODEL A4

APPENDIX F1
OPL SCRIPT OF MODEL A4

```
Model cp ( "constraint_aircraft_recovery_route.mod", "CP_arr_dept.dat",
"CP_dest_org.dat", "CP_flights.dat", "CP_city.dat", "CP_aircrafts.dat") editMode;

data "CP_initial.dat";
import enum Flights cp.Flights;
import enum Aircrafts cp.Aircrafts;
import enum City cp.City;
int seq;
float time:=0;

Flights initial[Aircrafts,1..4]:=...;
Flights initial2[Aircrafts,1..6]:=...;
int inCost[Aircrafts]:=...;

Open int pair[Aircrafts,Flights, int+];
Open int colno[int+];
int recBegin:= 720;
int recFinish:= 1200;
City recSt := IST;
Aircrafts shortage := MD88c;

forall(a in Aircrafts){
  colno.addh();
  colno[colno.up]:=inCost[a];
  if (a<> MD83b) then{
  forall(i in 1..4){
    {
    Flights j :=initial[a,i];
    pair[a,j].addh();
    pair[a,j,pair[a,j].up] := colno.up;
    }  }  }
  if (a=MD83b) then{
  forall(i in 1..6){
    {
```

```
      Flights j :=initial2[a,i];
      pair[a,j].addh();
      pair[a,j,pair[a,j].up] := colno.up;
      } } } }
forall ( f in Flights:cp.dept[f]>=recBegin & cp.arr[f]<= recFinish)
{
if ( cp.org[f]<>cp.dest[f]) then{
cp.reset();
cp.cover := f;
cp.seq :=4;
while cp.nextSolution()
do {
   forall ( a in cp.Aircrafts){
   colno.addh();
   colno[colno.up]:=cp.duty[a];
   forall ( i in cp.fltrange)
      {
      Flights j :=cp.fltseq[a,i];
      pair[a,j].addh();
      pair[a,j,pair[a,j].up] := colno.up;
      }   }    }
time := time +cp.getTime();
cp.reset();
cp.cover := f;
cp.seq :=6;
while cp.nextSolution()
do {
   forall ( a in Aircrafts){
   colno.addh();
   colno[colno.up]:=cp.duty[a];
   forall ( i in cp.fltrange)
      {
      Flights j :=cp.fltseq[a,i];
      pair[a,j].addh();
      pair[a,j,pair[a,j].up] := colno.up;
      }   }   }
time := time +cp.getTime();
} }
forall( f in Flights : cp.dept[f]<=recBegin \/ cp.arr[f]>=recFinish){
cp.reset();
cp.cover := f;
cp.seq :=2;
```

```
while cp.nextSolution()
do {
  forall ( a in Aircrafts){
  colno.addh();
  colno[colno.up]:=cp.duty[a];
  forall ( i in cp.fltrange)
    {
    Flights j :=cp.fltseq[a,i];
    pair[a,j].addh();
    pair[a,j,pair[a,j].up] := colno.up;
    }
  }
  time := time +cp.getTime();
}   }
cout<<"SOLUTION:"<<endl;
cout<<"Total number of new pairings: "<<colno.size<<" have been found in time
"<<time<<endl;

Model sl ("constraint_aircraft_recovery_opt.mod", "CP_arr_dept.dat",
"CP_dest_org.dat", "CP_assign.dat");
sl.solve();
time := time +sl.getTime();
Open Flights sol[sl.Columns,int+];
Open Aircrafts ac[sl.Columns, int+];
forall ( a in sl.Aircrafts){
forall ( i in Flights) {
  forall(j in [pair[a,i].low..pair[a,i].up]:sl.choose[pair[a,i,j]] =1)
  {
  int k:=pair[a,i,j];
  sol[k].addh();
  sol[k,sol[k].up]:=i;
  ac[k].addh();
  ac[k,ac[k].up]:=a;
  } } }

float Obj := sl.objectiveValue();
cout<<"Objective Value = "<<Obj<<endl;
cout<<"Solution Time = "<<time<<endl;
forall( i in Flights){
if(sl.cancel[i]=1)then{
cout<<"Cancelled Flights: "<<i<<endl;
} }
```

```
forall ( j in sl.Columns: sl.choose[j] =1){
  cout<<endl;
  cout<<"Aircraft "<<ac[j,0]<<" uses route "<<j+1<<endl;
  forall(k in [sol[j].low..sol[j].up]){
    cout<<sol[j,k]<<" ";
  City org :=cp.org[sol[j,k]];
  City dest :=cp.dest[sol[j,k]];
  int dep :=cp.dept[sol[j,k]];
  int arr :=cp.arr[sol[j,k]];
  int depH :=(dep/60) mod 12;
  int arrH :=(arr/60) mod 12;
  int depM := dep mod 60;
  int arrM := arr mod 60;
  string depZ := "";
  string arrZ :="";
  string depAP :=" pm";
  string arrAP :=" pm";
  if (depH = 0) then depH :=12;
  if (arrH = 0) then arrH :=12;
  if (depM < 10) then depZ := "0";
  if (arrM < 10) then arrZ := "0";
  if (dep <720 ) then depAP := " am";
  if (arr <720 ) then arrAP := " am";
  cout<<"From "<<org<<" to "<<dest<<" at ";
  cout<<depH<<":"<<depZ<<depM<<depAP;
  cout<<" --> ";
  cout<<arrH<<":"<<arrZ<<arrM<<arrAP;
  cout<<endl;
  }
int duty :=max (k in [sol[j].low..sol[j].up])cp.arr[sol[j,k]]
       - min (k in [sol[j].low..sol[j].up])cp.dept[sol[j,k]];
int flight := sum(k in [sol[j].low..sol[j].up])(cp.arr[sol[j,k]]- cp.dept[sol[j,k]]);
cout<<"Duty time = "<<duty;
cout<<", Flight time = "<<flight<<endl;
}
```

APPENDIX F2

OPL MODEL FIRST-STAGE PROBLEM

CONSTRAINT PROGRAMMING (MODEL A4)

```
enum Flights...;
enum City...;
enum Aircrafts...;
City dest[Flights]=...;
City org[Flights]=...;
int arr[Flights]=...;
int dept[Flights]=...;
Flights cover = ...;
int stop =45;
int seq=...;

range cityrange[0..seq];
range fltrange[1..seq];
var Flights fltseq[Aircrafts, fltrange];
var City cityseq[Aircrafts, cityrange];
var int duty[Aircrafts] in 0..1440;
var int flight[Aircrafts] in 0..1440;
var int+ startTime[Aircrafts] in 0..1440;
var int+ endTime[Aircrafts] in 0..1440;
var Flights startFlight[Aircrafts];
var Flights endFlight[Aircrafts];

predicate p(Flights f, City o, City d)
  return d=dest[f] & o=org[f];
solve {
forall( a in Aircrafts){
startFlight[a] =fltseq[a,1];
endFlight[a] = fltseq[a,seq];
cityseq[a,0] = cityseq[a,seq];
};
forall (a in Aircrafts)
  forall(i in fltrange : i<seq)
        dest[fltseq[a,i]] = org[fltseq[a,i+1]];
forall (a in Aircrafts, i in fltrange: i<seq)
         arr[fltseq[a,i]] + stop <= dept[fltseq[a,i+1]];
forall (a in Aircrafts)
        cityseq[a,0]<>cityseq[a,1];
```

```
forall (a in Aircrafts)
  forall( i,k in fltrange : 1<i<k)
        (cityseq[a,i-1] = cityseq[a,i]) =>(cityseq[a,k-1] = cityseq[a,k]);
forall (a in Aircrafts)
  forall( i in fltrange)
        p (fltseq[a,i], cityseq[a,i-1], cityseq[a,i]);
forall ( a in Aircrafts){
        duty[a] = arr[fltseq[a,seq]]-dept[fltseq[a,1]];
flight[a] = sum( i in fltrange) (arr[fltseq[a,i]]-dept[fltseq[a,i]]);
startTime[a] = dept[fltseq[a,1]];
endTime[a] = arr[fltseq[a,seq]];
};
forall( a in Aircrafts)
  forall ( f in Flights)
        (org[f]<>dest[f]) => sum (i in fltrange) (fltseq[a,i] =f ) <= 1;
forall( a in Aircrafts)
        sum (i in fltrange) (fltseq[a,i] = cover ) = 1;
forall(i in fltrange)
  forall ( a,b in Aircrafts: a<b )
        (fltseq[a,i] = fltseq[b,i]);
};
search{
  generate(fltseq);
  generate(cityseq);
};
data{
   cover = OHY034;
   seq = 4;
};
```

APPENDIX F3

OPL MODEL SECOND-STAGE PROBLEM

INTEGER PROGRAMMING (MODEL A4)

```
import enum Aircrafts;
import enum Flights;
import enum City;
City dest[Flights]=...;
City org[Flights]=...;
int arr[Flights]=...;
int dept[Flights]=...;
range Boolean 0..1;
Boolean assign[Aircrafts, Flights]=...;
int  AC[Aircrafts, Flights]=...;
import Open colno;
import Open pair;
range Columns[colno.low..colno.up];
var int+ choose[Columns] in 0..1;
var Boolean cancel[Flights];
var Boolean x[Aircrafts, Flights];
var int dif[Aircrafts, Flights] in 0..1;
import int recBegin;
import int recFinish;
import City recSt;
import Aircraft shortage;

minimize
  sum (i in Columns) colno[i] * choose[i] + sum(i in Flights) 1000*cancel[i] + sum (
         a in Aircrafts, i in Flights) 100*dif[a,i]

subject to {
  forall( i in Flights)
     (sum (a in Aircrafts,j in [pair[a,i].low..pair[a,i].up]) choose[pair[a,i,j]]) +
         cancel[i] = 1;

  forall ( a in Aircrafts)
      sum( i in Flights, j in [pair[a,i].low..pair[a,i].up]) choose[pair[a,i,j]] >= 1;

 sum ( i in Columns) choose[i]=9;

forall( a in Aircrafts, i in Flights: dept[i]<recBegin)
```

```
        ((sum (j in [pair[a,i].low..pair[a,i].up]) choose[pair[a,i,j]]) = assign[a,i]);
forall( a in Aircrafts, i in Flights: arr[i]>recFinish)
         ((sum (j in [pair[a,i].low..pair[a,i].up]) choose[pair[a,i,j]]) = assign[a,i]);
forall( a in Aircrafts: a=shortage, i in Flights: dept[i]>=recBegin & arr[i]<=recFinish)
           ((sum (j in [pair[a,i].low..pair[a,i].up]) choose[pair[a,i,j]])=0);
forall(i in Flights: dept[i]>=recBegin & arr[i]<=recFinish)
       ((sum (a in Aircrafts: a<>shortage,j in [pair[a,i].low..pair[a,i].up])
       choose[pair[a,i,j]])=1);
forall(a in Aircrafts : a<>shortage, i in Flights)
     ((sum( j in [pair[a,i].low..pair[a,i].up]) choose[pair[a,i,j]])= x[a,i]);
  forall(a in Aircrafts : a<>shortage, i in Flights)
     ((sum( j in [pair[a,i].low..pair[a,i].up]) choose[pair[a,i,j]])<= AC[a,i]);
forall(a in Aircrafts, i in Flights : assign[a,i] = 1) dif[a,i] = assign[a,i]-x[a,i];
forall(a in Aircrafts, i in Flights : assign[a,i] = 0) dif[a,i] = x[a,i]-assign[a,i];
};
```

APPENDIX F4

AIRCRAFT RECOVERY PROBLEM SOLUTION (MODEL A4)

(shortage on MD88a from 06:50 to 12:00)

SOLUTION:
Total number of new pairings: 3096 have been found in time 1.9660
Objective Value = 6695.0000
Solution Time = 2.0910

Aircraft MD88a uses route 2710
OHY016 From IST to ADA at 3:00 pm --> 4:30 pm
OHY017 From ADA to IST at 5:30 pm --> 7:00 pm
Duty time = 240, Flight time = 180

Aircraft MD88b uses route 614
OHY022 From IST to AYT at 8:15 am --> 9:15 am
OHY023 From AYT to IST at 10:45 am --> 11:45 am
OHY054 From IST to ADB at 4:30 pm --> 5:30 pm
OHY055 From ADB to IST at 6:50 pm --> 7:45 pm
Duty time = 690, Flight time = 235

Aircraft MD88c uses route 957
OHY034 From IST to DIY at 6:45 am --> 8:30 am
OHY035 From DIY to IST at 9:30 am --> 11:15 am
OHY038 From IST to ERZ at 12:30 pm --> 2:20 pm
OHY039 From ERZ to IST at 3:20 pm --> 5:10 pm
OHY086 From IST to TZX at 6:45 pm --> 8:20 pm
OHY087 From TZX to IST at 9:15 pm --> 10:50 pm
Duty time = 965, Flight time = 620

Aircraft MD88d uses route 4
OHY010 From IST to ADA at 6:45 am --> 8:15 am
OHY011 From ADA to IST at 9:10 am --> 10:40 am
OHY028 From IST to BJV at 11:55 am --> 12:55 pm
OHY029 From BJV to IST at 1:45 pm --> 2:45 pm
Duty time = 480, Flight time = 300

Aircraft MD88e uses route 2192
OHY026 From IST to AYT at 8:30 pm --> 9:30 pm
OHY027 From AYT to IST at 7:30 am --> 8:30 am
OHY074 From IST to SZF at 9:30 am --> 10:45 am
OHY075 From SZF to IST at 11:45 am --> 1:00 pm
Duty time = 840, Flight time = 270

Aircraft MD83a uses route 1806
OHY062 From IST to MLX at 12:15 pm --> 1:45 pm
OHY063 From MLX to IST at 2:30 pm --> 4:00 pm
OHY066 From IST to KSY at 7:15 am --> 8:30 am
OHY067 From KSY to IST at 9:15 am --> 10:30 am
Duty time = 525, Flight time = 330

Aircraft MD83b uses route 1303
OHY024 From IST to AYT at 4:45 pm --> 5:45 pm
OHY025 From AYT to IST at 6:45 pm --> 7:45 pm
OHY042 From IST to GZT at 9:30 am --> 11:15 am
OHY043 From GZT to IST at 12:15 pm --> 2:00 pm
OHY058 From IST to ADB at 8:45 pm --> 9:45 pm
OHY059 From ADB to IST at 7:30 am --> 8:25 am
Duty time = 855, Flight time = 445

Aircraft MD83c uses route 8
OHY018 From IST to ADA at 7:00 pm --> 8:30 pm
OHY019 From ADA to IST at 9:30 pm --> 11:00 pm
OHY080 From IST to TZX at 6:50 am --> 8:25 am
OHY081 From TZX to IST at 9:20 am --> 10:55 am
Duty time = 970, Flight time = 370

Aircraft MD83d uses route 9
OHY036 From IST to DIY at 6:45 pm --> 8:30 pm
OHY037 From DIY to IST at 9:30 pm --> 11:15 pm
OHY050 From IST to ADB at 7:45 am --> 8:45 am
OHY051 From ADB to IST at 10:50 am --> 11:45 am
Duty time = 930, Flight time = 325

APPENDIX F5

AIRCRAFT RECOVERY PROBLEM SOLUTION (MODEL A4)

(shortage on MD88c from 12:00 to 17:30)

SOLUTION:
Total number of new pairings: 3465 have been found in time 1.8870
Objective Value = 6390.0000
Solution Time = 2.0120

Aircraft MD88a uses route 28
OHY016 From IST to ADA at 3:00 pm --> 4:30 pm
OHY017 From ADA to IST at 5:30 pm --> 7:00 pm
OHY034 From IST to DIY at 6:45 am --> 8:30 am
OHY035 From DIY to IST at 9:30 am --> 11:15 am
Duty time = 735, Flight time = 390

Aircraft MD88b uses route 3107
OHY022 From IST to AYT at 8:15 am --> 9:15 am
OHY023 From AYT to IST at 10:45 am --> 11:45 am
Duty time = 210, Flight time = 120

Aircraft MD88c uses route 3441
OHY086 From IST to TZX at 6:45 pm --> 8:20 pm
OHY087 From TZX to IST at 9:15 pm --> 10:50 pm
Duty time = 245, Flight time = 190

Aircraft MD88d uses route 4
OHY010 From IST to ADA at 6:45 am --> 8:15 am
OHY011 From ADA to IST at 9:10 am --> 10:40 am
OHY028 From IST to BJV at 11:55 am --> 12:55 pm
OHY029 From BJV to IST at 1:45 pm --> 2:45 pm
Duty time = 480, Flight time = 300

Aircraft MD88e uses route 2183
OHY026 From IST to AYT at 8:30 pm --> 9:30 pm
OHY027 From AYT to IST at 7:30 am --> 8:30 am
OHY054 From IST to ADB at 4:30 pm --> 5:30 pm
OHY055 From ADB to IST at 6:50 pm --> 7:45 pm
OHY074 From IST to SZF at 9:30 am --> 10:45 am
OHY075 From SZF to IST at 11:45 am --> 1:00 pm
Duty time = 840, Flight time = 385

Aircraft MD83a uses route 2751

OHY062 From IST to MLX at 12:15 pm --> 1:45 pm
OHY063 From MLX to IST at 2:30 pm --> 4:00 pm
OHY066 From IST to KSY at 7:15 am --> 8:30 am
OHY067 From KSY to IST at 9:15 am --> 10:30 am
Duty time = 525, Flight time = 330

Aircraft MD83b uses route 583
OHY024 From IST to AYT at 4:45 pm --> 5:45 pm
OHY025 From AYT to IST at 6:45 pm --> 7:45 pm
OHY042 From IST to GZT at 9:30 am --> 11:15 am
OHY043 From GZT to IST at 12:15 pm --> 2:00 pm
OHY058 From IST to ADB at 8:45 pm --> 9:45 pm
OHY059 From ADB to IST at 7:30 am --> 8:25 am
Duty time = 855, Flight time = 445

Aircraft MD83c uses route 1727
OHY018 From IST to ADA at 7:00 pm --> 8:30 pm
OHY019 From ADA to IST at 9:30 pm --> 11:00 pm
OHY038 From IST to ERZ at 12:30 pm --> 2:20 pm
OHY039 From ERZ to IST at 3:20 pm --> 5:10 pm
OHY080 From IST to TZX at 6:50 am --> 8:25 am
OHY081 From TZX to IST at 9:20 am --> 10:55 am
Duty time = 970, Flight time = 590

Aircraft MD83d uses route 9
OHY036 From IST to DIY at 6:45 pm --> 8:30 pm
OHY037 From DIY to IST at 9:30 pm --> 11:15 pm
OHY050 From IST to ADB at 7:45 am --> 8:45 am
OHY051 From ADB to IST at 10:50 am --> 11:45 am
Duty time = 930, Flight time = 325

APPENDIX G

OPL MODEL OF CREW RECOVERY PROBLEM (MODEL C1)

```
1) Problem statement
====================
range Boolean 0..1;
enum Flights ...;
enum Crews ...;
Crews shortage= Crew3;
int start=360;
int finish=720;
struct Connection { Flights i; Flights j; };
struct Route { Crews a; Connection c; };

{Route} Routes = ...;
{Connection} Connections = { c | <a,c> in Routes };
Boolean pairings[Routes]=...;
int+ cost[Routes] = ...;
int dept[Flights]=...;
int arr[Flights]=...;
int org[Flights]=...;
int dest[Flights]=...;
{Connection} CP[a in Crews] = { c | <a,c> in Routes };
var Boolean resch[Routes];
var Boolean cancel[Flights];
var Boolean reserve[Flights];
var float+ nSeq[Crews];
var float+ hub1[Crews];
var float+ hub2[Crews];

minimize
  sum(l in Routes) cost[l] * resch[l] +sum(i in Flights) 1000 *cancel[i] + sum(i in
Flights) 360 *reserve[i]

subject to{
forall (a in Crews, i in Flights: i=source)
    sum (<i,j> in CP[a]) resch[< a,<i,j> >]=1;
forall (a in Crews, j in Flights:j=sink)
    sum (<i,j> in CP[a]) resch[< a,<i,j> >]=1;
forall (a in Crews)
       hub1[a] =  sum(<i,j> in CP[a]: i=source) resch[<a,<i,j>>]*org[j];
```

```
forall (a in Crews)
        hub2[a] =  sum(<i,j> in CP[a]: j=sink) resch[<a,<i,j>>]*dest[i];
forall (a in Crews)
        hub2[a] - hub1[a] = 0 ;
forall(a in Crews, <i,j> in CP[a]:a=shortage & dept[i]>=start & dept[j]<finish)
     resch[<a,<i,j>>]=0;
forall(a in Crews, <i,j> in CP[a]: a<>shortage & dept[j]<start)
    resch[<a,<i,j>>]=pairings[<a,<i,j>>];
forall(a in Crews, <i,j> in CP[a]: dept[i]>= finish)
    resch[<a,<i,j>>]=pairings[<a,<i,j>>];
forall (a in Crews, i in Flights:  i<>sink & i<>source )
    sum (<i,j> in CP[a]) resch[< a,<i,j> >]
  - sum (<k,i> in CP[a]) resch[< a,<k,i> >]=0;
forall (a in Crews,<i,j> in CP[a]:dept[i]>=start & dept[j]<finish)
    resch[<a,<i,j>>]*(arr[i]+45-dept[j])<=0;
forall (a in Crews,<i,j> in Connections : i<>source & j<>sink)
     resch[<a,<i,j>>]*(dest[i]-org[j])=0;
forall(i in Flights : i<>source & i<> sink)
     cancel[i]+ reserve[i]+sum (<i,j> in Connections) sum(a in Crews)
     resch[<a,<i,j>>]>=1;
forall (a in Crews :a<>shortage)
    sum(<i,j> in CP[a]) resch[<a,<i,j>>]<=5;
forall (a in Crews :a<>shortage)
    sum(<i,j> in CP[a]) resch[<a,<i,j>>]>=3;

forall(a in Crews)
     sum (<i,j> in Connections: i<>source)
       resch[<a,<i,j>>]*(arr[i]-dept[i])<=300;
sum (i in Flights) reserve[i]*(arr[i]-dept[i])<=300;
  forall(a in Crews)
     nSeq[a]=sum(<i,j> in CP[a])resch[<a,<i,j>>];
};
search{
  forall(a in Crews)
     tryall(<i,j> in Connections)
      {  hub1[a]-hub2[a] = 0;
       resch[<a,<i,j>>] = pairings[<a,<i,j>>]; };
};
display (l in Routes : resch[l] > 0) <resch[l]>;
display (i in Flights : cancel[i] > 0) <cancel[i]>;
display (i in Flights) <reserve[i]>;
```

2) Results
==========
Optimal Solution with Objective Value: 20465
<resch[#<a:Crew1,c:#<i:source,j:OHY027>#>#]> = <1>
<resch[#<a:Crew1,c:#<i:OHY027,j:OHY062>#>#]> = <1>
<resch[#<a:Crew1,c:#<i:OHY062,j:OHY063>#>#]> = <1>
<resch[#<a:Crew1,c:#<i:OHY063,j:OHY026>#>#]> = <1>
<resch[#<a:Crew1,c:#<i:OHY026,j:sink>#>#]> = <1>
<resch[#<a:Crew2,c:#<i:source,j:OHY022>#>#]> = <1>
<resch[#<a:Crew2,c:#<i:OHY018,j:OHY019>#>#]> = <1>
<resch[#<a:Crew2,c:#<i:OHY022,j:OHY023>#>#]> = <1>
<resch[#<a:Crew2,c:#<i:OHY023,j:OHY018>#>#]> = <1>
<resch[#<a:Crew2,c:#<i:OHY019,j:sink>#>#]> = <1>
<resch[#<a:Crew3,c:#<i:source,j:OHY054>#>#]> = <1>
<resch[#<a:Crew3,c:#<i:OHY054,j:OHY055>#>#]> = <1>
<resch[#<a:Crew3,c:#<i:OHY055,j:sink>#>#]> = <1>
<resch[#<a:Crew4,c:#<i:source,j:OHY050>#>#]> = <1>
<resch[#<a:Crew4,c:#<i:OHY016,j:OHY017>#>#]> = <1>
<resch[#<a:Crew4,c:#<i:OHY050,j:OHY051>#>#]> = <1>
<resch[#<a:Crew4,c:#<i:OHY051,j:OHY016>#>#]> = <1>
<resch[#<a:Crew4,c:#<i:OHY017,j:sink>#>#]> = <1>
<resch[#<a:Crew5,c:#<i:source,j:OHY042>#>#]> = <1>
<resch[#<a:Crew5,c:#<i:OHY042,j:OHY043>#>#]> = <1>
<resch[#<a:Crew5,c:#<i:OHY043,j:sink>#>#]> = <1>
<resch[#<a:Crew6,c:#<i:source,j:OHY036>#>#]> = <1>
<resch[#<a:Crew6,c:#<i:OHY036,j:OHY037>#>#]> = <1>
<resch[#<a:Crew6,c:#<i:OHY037,j:sink>#>#]> = <1>
<resch[#<a:Crew7,c:#<i:source,j:OHY080>#>#]> = <1>
<resch[#<a:Crew7,c:#<i:OHY080,j:OHY081>#>#]> = <1>
<resch[#<a:Crew7,c:#<i:OHY081,j:sink>#>#]> = <1>
<resch[#<a:Crew8,c:#<i:source,j:OHY038>#>#]> = <1>
<resch[#<a:Crew8,c:#<i:OHY038,j:OHY039>#>#]> = <1>
<resch[#<a:Crew8,c:#<i:OHY039,j:sink>#>#]> = <1>
<resch[#<a:Crew9,c:#<i:source,j:OHY066>#>#]> = <1>
<resch[#<a:Crew9,c:#<i:OHY024,j:OHY025>#>#]> = <1>
<resch[#<a:Crew9,c:#<i:OHY066,j:OHY067>#>#]> = <1>
<resch[#<a:Crew9,c:#<i:OHY067,j:OHY024>#>#]> = <1>
<resch[#<a:Crew9,c:#<i:OHY025,j:sink>#>#]> = <1>
<resch[#<a:Crew10,c:#<i:source,j:OHY010>#>#]> = <1>
<resch[#<a:Crew10,c:#<i:OHY010,j:OHY011>#>#]> = <1>
<resch[#<a:Crew10,c:#<i:OHY011,j:OHY028>#>#]> = <1>
<resch[#<a:Crew10,c:#<i:OHY028,j:OHY029>#>#]> = <1>

<resch[#<a:Crew10,c:#<i:OHY029,j:sink>#>#]> = <1>
<resch[#<a:Crew11,c:#<i:source,j:OHY059>#>#]> = <1>
<resch[#<a:Crew11,c:#<i:OHY059,j:OHY074>#>#]> = <1>
<resch[#<a:Crew11,c:#<i:OHY074,j:OHY075>#>#]> = <1>
<resch[#<a:Crew11,c:#<i:OHY075,j:OHY058>#>#]> = <1>
<resch[#<a:Crew11,c:#<i:OHY058,j:sink>#>#]> = <1>
<resch[#<a:Crew12,c:#<i:source,j:OHY086>#>#]> = <1>
<resch[#<a:Crew12,c:#<i:OHY086,j:OHY087>#>#]> = <1>
<resch[#<a:Crew12,c:#<i:OHY087,j:sink>#>#]> = <1>

APPENDIX H

OPL CODES FOR CREW RECOVERY MODEL C2

APPENDIX H1

THE OPL SCRIPT OF MODEL C2

```
Model cp ( "constraint_crew_pairings_recovery.mod", "CP_arr_dept.dat",
"CP_dest_org.dat", "CP_flights.dat", "CP_city.dat", "CP_crews.dat") editMode;
data "CP_arr_dept.dat";
data "CP_flights.dat";
data "CP_initial_crew.dat";
import enum Flights cp.Flights;
import enum Crews cp.Crews;
import enum City cp.City;
int seq;
float time:=0;
Flights initial[Crews,1..4]:=...;
Flights initial2[Crews,1..2]:=...;
int inCost[Crews]:=...;
Open int pair[Crews,Flights, int+];
Open int colno[int+];
int recBegin:= 400;
int recFinish:= 720;
City recSt := IST;
Crews shortage := C11;

forall(c in Crews){
  colno.addh();
  colno[colno.up]:=inCost[c];
  if ( c=C01)then{
  forall(i in 1..4)
    {
    Flights j :=initial[c,i];
    pair[c,j].addh();
    pair[c,j,pair[c,j].up] := colno.up;
    } }
  if ( c=C02)then{
  forall(i in 1..4)
    {
    Flights j :=initial[c,i];
```

```
  pair[c,j].addh();
  pair[c,j,pair[c,j].up] := colno.up;
  } }
if ( c=C03)then{
forall(i in 1..4)
  {
  Flights j :=initial[c,i];
  pair[c,j].addh();
  pair[c,j,pair[c,j].up] := colno.up;
  } }
if ( c=C05)then{
forall(i in 1..4)
  {
  Flights j :=initial[c,i];
  pair[c,j].addh();
  pair[c,j,pair[c,j].up] := colno.up;
  }  }
if ( c=C07)then{
forall(i in 1..4)
  {
  Flights j :=initial[c,i];
  pair[c,j].addh();
  pair[c,j,pair[c,j].up] := colno.up;
  }  }
if ( c=C08)then{
forall(i in 1..4)
  {
  Flights j :=initial[c,i];
  pair[c,j].addh();
  pair[c,j,pair[c,j].up] := colno.up;
  }  }
if ( c=C12)then{
forall(i in 1..4)
  {
  Flights j :=initial[c,i];
  pair[c,j].addh();
  pair[c,j,pair[c,j].up] := colno.up;
  }  }
if ( c=C04)then{
forall(i in 1..2)
  {
  Flights j :=initial2[c,i];
```

```
      pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } }
   if ( c=C06)then{
   forall(i in 1..2)
      {
      Flights j :=initial2[c,i];
      pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } }
   if ( c=C09)then{
   forall(i in 1..2)
      {
      Flights j :=initial2[c,i];
      pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } }
   if ( c=C10)then{
   forall(i in 1..2)
      {
      Flights j :=initial2[c,i];
pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } }
   if ( c=C11)then{
   forall(i in 1..2)
      {
      Flights j :=initial2[c,i];
      pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } } }
forall ( f in Flights:cp.dept[f]>=recBegin & cp.arr[f]<= recFinish) {
if ( cp.org[f]<>cp.dest[f]) then{
cp.reset();
cp.cover := f;
cp.seq :=2;
while cp.nextSolution()
do {
   forall ( c in cp.Crews){
   colno.addh();
   colno[colno.up]:=cp.duty[c];
   forall ( i in cp.fltrange)
```

```
      {
      Flights j :=cp.fltseq[c,i];
      pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } }  }
time := time +cp.getTime();
cp.reset();
cp.cover := f;
cp.seq :=4;
while cp.nextSolution()
do {
   forall ( c in Crews){
   colno.addh();
   colno[colno.up]:=cp.duty[c];
   forall ( i in cp.fltrange)
      {
      Flights j :=cp.fltseq[c,i];
      pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } }  }
time := time +cp.getTime();
cp.reset();
cp.cover := f;
cp.seq :=6;
while cp.nextSolution()
do {
   forall ( c in Crews){
   colno.addh();
   colno[colno.up]:=cp.duty[c];
   forall ( i in cp.fltrange)
      {
      Flights j :=cp.fltseq[c,i];
      pair[c,j].addh();
      pair[c,j,pair[c,j].up] := colno.up;
      } } } } }
forall( f in Flights : cp.dept[f]<=recBegin \/ cp.arr[f]>=recFinish){
cp.reset();
cp.cover := f;
cp.seq :=2;

while cp.nextSolution()
do {
```

```
  forall ( c in Crews){
  colno.addh();
  colno[colno.up]:=cp.duty[c];
  forall ( i in cp.fltrange)
    {
    Flights j :=cp.fltseq[c,i];
    pair[c,j].addh();
    pair[c,j,pair[c,j].up] := colno.up;
    } }
  time := time +cp.getTime();
} }
cout<<"SOLUTION:"<<endl;
cout<<"Total number of new pairings: "<<colno.size<<" have been found in time
"<<time<<endl;
Model sl ("constraint_crew_recovery_opt.mod", "CP_arr_dept.dat",
"CP_dest_org.dat", "CP_assign_crew.dat");
sl.solve();
time := time +sl.getTime();
Open Flights sol[sl.Columns,int+];
Open Crews cr[sl.Columns, int+];
forall ( c in sl.Crews){
forall ( i in Flights) {
  forall(j in [pair[c,i].low..pair[c,i].up]:sl.choose[pair[c,i,j]] =1)
  {
  int k:=pair[c,i,j];
  sol[k].addh();
 sol[k,sol[k].up]:=i;
  cr[k].addh();
  cr[k,cr[k].up]:=c;
  } } }
float Obj := sl.objectiveValue();
cout<<"Objective Value = "<<Obj<<endl;
cout<<"Solution Time = "<<time<<endl;
forall( i in Flights){
if(sl.cancel[i]=1)then{
cout<<"Cancelled Flights: "<<i<<endl;
}
if(sl.reserve[i]=1)then{
cout<<"Flight: "<<i<<" is flown by reserve crew"<<endl;
} }
forall ( j in sl.Columns: sl.choose[j] =1){
  cout<<endl;
```

```
  cout<<"Crew "<<cr[j,0]<<" uses route "<<j+1<<endl;
  forall(k in [sol[j].low..sol[j].up]){
    cout<<sol[j,k]<<" ";

  City org :=cp.org[sol[j,k]];
  City dest :=cp.dest[sol[j,k]];
  int dep :=cp.dept[sol[j,k]];
  int arr :=cp.arr[sol[j,k]];
  int depH :=(dep/60) mod 12;
  int arrH :=(arr/60) mod 12;
  int depM := dep mod 60;
  int arrM := arr mod 60;
  string depZ := "";
  string arrZ :="";
  string depAP :=" pm";
  string arrAP :=" pm";

 if (depH = 0) then depH :=12;
  if (arrH = 0) then arrH :=12;
  if (depM < 10) then depZ := "0";
  if (arrM < 10) then arrZ := "0";
  if (dep <720 ) then depAP := " am";
  if (arr <720 ) then arrAP := " am";

 cout<<"From "<<org<<" to "<<dest<<" at ";
  cout<<depH<<":"<<depZ<<depM<<depAP;
  cout<<" --> ";
  cout<<arrH<<":"<<arrZ<<arrM<<arrAP;
  cout<<endl;
  }
int duty :=max (k in [sol[j].low..sol[j].up])cp.arr[sol[j,k]]
       - min (k in [sol[j].low..sol[j].up])cp.dept[sol[j,k]];
int flight := sum(k in [sol[j].low..sol[j].up])(cp.arr[sol[j,k]]- cp.dept[sol[j,k]]);
cout<<"Duty time = "<<duty;
cout<<", Flight time = "<<flight<<endl;
 }
```

APPENDIX H2
THE OPL MODEL OF FIRST-STAGE CONSTRAINT PROGRAMMING (MODEL C2)

```
enum Flights...;
enum City...;
enum Crews...;
City dest[Flights]=...;
City org[Flights]=...;
int arr[Flights]=...;
int dept[Flights]=...;
Flights cover = ...;
int stop =45;
int seq=...;
int maxDuty = 900;
int maxFlight = 360;
range cityrange[0..seq];
range fltrange[1..seq];

var Flights fltseq[Crews, fltrange];
var City cityseq[Crews, cityrange];
var int duty[Crews] in 0..1440;
var int flight[Crews] in 0..1440;
var int+ startTime[Crews] in 0..1440;
var int+ endTime[Crews] in 0..1440;
var Flights startFlight[Crews];
var Flights endFlight[Crews];

predicate p(Flights f, City o, City d)
  return d=dest[f] & o=org[f];
solve {

forall( c in Crews){
        startFlight[c] =fltseq[c,1];
        endFlight[c] = fltseq[c,seq];
        cityseq[c,0] = cityseq[c,seq];
};
forall (c in Crews)
  forall(i in fltrange : i<seq)
         dest[fltseq[c,i]] = org[fltseq[c,i+1]];
forall (c in Crews, i in fltrange: i<seq)
```

```
        arr[fltseq[c,i]] + stop <= dept[fltseq[c,i+1]];
forall (c in Crews)
        cityseq[c,0]<>cityseq[c,1];
forall (c in Crews)
  forall( i,k in fltrange : 1<i<k)
        (cityseq[c,i-1] = cityseq[c,i]) =>(cityseq[c,k-1] = cityseq[c,k]);
forall (c in Crews)
  forall( i in fltrange)
        p (fltseq[c,i], cityseq[c,i-1], cityseq[c,i]);
forall ( c in Crews){
        duty[c] = arr[fltseq[c,seq]]-dept[fltseq[c,1]];
        flight[c] = sum( i in fltrange) (arr[fltseq[c,i]]-dept[fltseq[c,i]]);
        startTime[c] = dept[fltseq[c,1]];
        endTime[c] = arr[fltseq[c,seq]];
        duty[c] <= maxDuty;
        flight[c] <= maxFlight;
};
forall( c in Crews)
  forall ( f in Flights)
        (org[f]<>dest[f]) => sum (i in fltrange) (fltseq[c,i] =f ) <= 1;
forall( c in Crews)
        sum (i in fltrange) (fltseq[c,i] = cover ) = 1;
forall(i in fltrange)
        forall ( c,b in Crews: c<b ) (fltseq[c,i] = fltseq[b,i]);

};
search{
  generate(fltseq);
  generate(cityseq);
};
data{
   cover = OHY034;
   seq = 2;
};
```

APPENDIX H3

OPL MODEL OF SECOND-STAGE PROBLEM

INTEGER PROGRAMMING (MODEL C2)

```
import enum Crews;
import enum Flights;
import enum City;
City dest[Flights]=...;
City org[Flights]=...;
int arr[Flights]=...;
int dept[Flights]=...;
range Boolean 0..1;
Boolean pairings[Crews, Flights]=...;
import Open colno;
import Open pair;
range Columns[colno.low..colno.up];

var int+ choose[Columns] in 0..1;
var Boolean cancel[Flights];
var Boolean x[Crews, Flights];
var int dif[Crews, Flights] in 0..1;
var Boolean reserve[Flights];
var int crewno in 0..20;

import int recBegin;
import int recFinish;
import City recSt;
import Crews shortage;

minimize
  sum (i in Columns) colno[i] * choose[i] +
  sum(i in Flights) 1000*cancel[i] + sum ( i in Flights) reserve[i]*300 +
  sum ( c in Crews, i in Flights) 100*dif[c,i]+crewno*1000

subject to {
  forall( i in Flights)
     (sum (c in Crews,j in [pair[c,i].low..pair[c,i].up])
        choose[pair[c,i,j]]) + cancel[i] +  reserve[i]= 1;
   forall ( c in Crews: c<>shortage)
      sum( i in Flights, j in [pair[c,i].low..pair[c,i].up]) choose[pair[c,i,j]] >= 1;
```

```
sum(k in Columns) choose[k]>=11;
crewno = sum( k in Columns)choose[k];
forall( c in Crews, i in Flights: dept[i]<recBegin)
          ((sum (j in [pair[c,i].low..pair[c,i].up]) choose[pair[c,i,j]]) = pairings[c,i]);
forall( c in Crews, i in Flights: arr[i]>recFinish)
          ((sum (j in [pair[c,i].low..pair[c,i].up]) choose[pair[c,i,j]]) = pairings[c,i]);
forall( c in Crews: c=shortage, i in Flights: dept[i]>=recBegin & arr[i]<=recFinish)
        ((sum (j in [pair[c,i].low..pair[c,i].up]) choose[pair[c,i,j]])=0);
forall( i in Flights: dept[i]>=recBegin & arr[i]<=recFinish)
        ((sum (c in Crews: c<>shortage, j in [pair[c,i].low..pair[c,i].up])
        choose[pair[c,i,j]]) + reserve[i]=1);
forall(c in Crews : c<>shortage, i in Flights)
     ((sum( j in [pair[c,i].low..pair[c,i].up]) choose[pair[c,i,j]])= x[c,i]);
forall(c in Crews, i in Flights : pairings[c,i] = 1)  dif[c,i] = pairings[c,i]-x[c,i];
forall(c in Crews, i in Flights : pairings[c,i] = 0) dif[c,i] = x[c,i]-pairings[c,i];
};
```

APPENDIX H4

CREW RECOVERY PROBLEM SOLUTION

(shortage on C11 from 06:50 to 12:00)

Total number of new pairings: 2160 have been found in time 2.2300
Objective Value = 6240.0000
Solution Time = 2.2920
Flight: OHY034 is flown by reserve crew
Flight: OHY035 is flown by reserve crew

Crew C01 uses route 1513
OHY054 From IST to ADB at 4:30 pm --> 5:30 pm
OHY059 From ADB to IST at 7:30 am --> 8:25 am
OHY074 From IST to SZF at 9:30 am --> 10:45 am
OHY075 From SZF to IST at 11:45 am --> 1:00 pm
Duty time = 600, Flight time = 265

Crew C02 uses route 290
OHY022 From IST to AYT at 8:15 am --> 9:15 am
OHY023 From AYT to IST at 10:45 am --> 11:45 am
OHY036 From IST to DIY at 6:45 pm --> 8:30 pm
OHY037 From DIY to IST at 9:30 pm --> 11:15 pm
Duty time = 900, Flight time = 330

Crew C03 uses route 1275
OHY016 From IST to ADA at 3:00 pm --> 4:30 pm
OHY017 From ADA to IST at 5:30 pm --> 7:00 pm
OHY066 From IST to KSY at 7:15 am --> 8:30 am
OHY067 From KSY to IST at 9:15 am --> 10:30 am
Duty time = 705, Flight time = 330

Crew C04 uses route 4
OHY018 From IST to ADA at 7:00 pm --> 8:30 pm
OHY019 From ADA to IST at 9:30 pm --> 11:00 pm
Duty time = 240, Flight time = 180

Crew C05 uses route 1637
OHY028 From IST to BJV at 11:55 am --> 12:55 pm
OHY029 From BJV to IST at 1:45 pm --> 2:45 pm
OHY080 From IST to TZX at 6:50 am --> 8:25 am
OHY081 From TZX to IST at 9:20 am --> 10:55 am
Duty time = 475, Flight time = 310

Crew C06 uses route 1842
OHY025 From AYT to IST at 6:45 pm --> 7:45 pm
OHY026 From IST to AYT at 8:30 pm --> 9:30 pm
Duty time = 165, Flight time = 120

Crew C07 uses route 559
OHY024 From IST to AYT at 4:45 pm --> 5:45 pm
OHY027 From AYT to IST at 7:30 am --> 8:30 am
OHY042 From IST to GZT at 9:30 am --> 11:15 am
OHY043 From GZT to IST at 12:15 pm --> 2:00 pm
Duty time = 615, Flight time = 330

Crew C08 uses route 8
OHY038 From IST to ERZ at 12:30 pm --> 2:20 pm
OHY039 From ERZ to IST at 3:20 pm --> 5:10 pm
OHY050 From IST to ADB at 7:45 am --> 8:45 am
OHY051 From ADB to IST at 10:50 am --> 11:45 am
Duty time = 565, Flight time = 335

Crew C09 uses route 2073
OHY055 From ADB to IST at 6:50 pm --> 7:45 pm
OHY058 From IST to ADB at 8:45 pm --> 9:45 pm
Duty time = 175, Flight time = 115

Crew C10 uses route 10
OHY086 From IST to TZX at 6:45 pm --> 8:20 pm
OHY087 From TZX to IST at 9:15 pm --> 10:50 pm
Duty time = 245, Flight time = 190

APPENDIX H5

CREW RECOVERY PROBLEM SOLUTION

(shortage on C08 from 12:00 to 20:00)

SOLUTION:
Total number of new pairings: 2544 have been found in time 2.2160
Objective Value = 17785.0000
Solution Time = 2.3100
Flight: OHY038 is flown by reserve crew
Flight: OHY039 is flown by reserve crew

Crew C02 uses route 2
OHY022 From IST to AYT at 8:15 am --> 9:15 am
OHY023 From AYT to IST at 10:45 am --> 11:45 am
OHY036 From IST to DIY at 6:45 pm --> 8:30 pm
OHY037 From DIY to IST at 9:30 pm --> 11:15 pm
Duty time = 900, Flight time = 330

Crew C03 uses route 3
OHY016 From IST to ADA at 3:00 pm --> 4:30 pm
OHY017 From ADA to IST at 5:30 pm --> 7:00 pm
OHY066 From IST to KSY at 7:15 am --> 8:30 am
OHY067 From KSY to IST at 9:15 am --> 10:30 am
Duty time = 705, Flight time = 330

Crew C05 uses route 5
OHY028 From IST to BJV at 11:55 am --> 12:55 pm
OHY029 From BJV to IST at 1:45 pm --> 2:45 pm
OHY080 From IST to TZX at 6:50 am --> 8:25 am
OHY081 From TZX to IST at 9:20 am --> 10:55 am
Duty time = 475, Flight time = 310

Crew C07 uses route 7
OHY024 From IST to AYT at 4:45 pm --> 5:45 pm
OHY027 From AYT to IST at 7:30 am --> 8:30 am
OHY042 From IST to GZT at 9:30 am --> 11:15 am
OHY043 From GZT to IST at 12:15 pm --> 2:00 pm
Duty time = 615, Flight time = 330

Crew C06 uses route 618
OHY025 From AYT to IST at 6:45 pm --> 7:45 pm
OHY026 From IST to AYT at 8:30 pm --> 9:30 pm

Duty time = 165, Flight time = 120

Crew C01 uses route 1513
OHY054 From IST to ADB at 4:30 pm --> 5:30 pm
OHY059 From ADB to IST at 7:30 am --> 8:25 am
OHY074 From IST to SZF at 9:30 am --> 10:45 am
OHY075 From SZF to IST at 11:45 am --> 1:00 pm
Duty time = 600, Flight time = 265

Crew C09 uses route 1581
OHY055 From ADB to IST at 6:50 pm --> 7:45 pm
OHY058 From IST to ADB at 8:45 pm --> 9:45 pm
Duty time = 175, Flight time = 115

Crew C12 uses route 1920
OHY010 From IST to ADA at 6:45 am --> 8:15 am
OHY011 From ADA to IST at 9:10 am --> 10:40 am
OHY062 From IST to MLX at 12:15 pm --> 1:45 pm
OHY063 From MLX to IST at 2:30 pm --> 4:00 pm
Duty time = 555, Flight time = 360

Crew C04 uses route 2080
OHY018 From IST to ADA at 7:00 pm --> 8:30 pm
OHY019 From ADA to IST at 9:30 pm --> 11:00 pm
Duty time = 240, Flight time = 180

Crew C11 uses route 2243
OHY034 From IST to DIY at 6:45 am --> 8:30 am
OHY035 From DIY to IST at 9:30 am --> 11:15 am
Duty time = 270, Flight time = 210

Crew C08 uses route 2312
OHY050 From IST to ADB at 7:45 am --> 8:45 am
OHY051 From ADB to IST at 10:50 am --> 11:45 am
Duty time = 240, Flight time = 115

Crew C10 uses route 2530
OHY086 From IST to TZX at 6:45 pm --> 8:20 pm
OHY087 From TZX to IST at 9:15 pm --> 10:50 pm
Duty time = 245, Flight time = 190

APPENDIX I

THE OPL SCRIPT OF THE SOLUTION ALGORITHM S1

```
enum Crews  = {Crew1, Crew2, Crew3, Crew4, Crew5, Crew6, Crew7, Crew8,
Crew9, Crew10, Crew11, Crew12};

enum Aircrafts = {MD88a, MD88b, MD88c, MD88d, MD88e, MD83a, MD83b,
MD83c, MD83d};
data "flights_crew_script.dat";

Crews shortage:= Crew8;
Aircrafts shortage1 := MD88c;
Aircrafts shortage2 := MD88a;

int start:= 750;
int finish := 1440;
int start1 := 480;
int finish1 := 720;
int start2 := 600;
int finish2 := 900;
float time :=0;

Model cr("recovery_crew_script.mod", "arrival dept.dat", "dest_org.dat",
"flights_crew_script.dat", "route_crews.dat", "cost_crew.dat", "pairings.dat");

import enum Flights cr.Flights;
float crewno[Flights];

Open int noc[Flights];
Open int Diff[Flights];

forall (i in Flights){
  Diff[i] := 0;
  crewno[i] := 1;
}

Model crew ("recovery_crew_script2.mod", "arrival dept.dat", "dest_org.dat",
"route_crews.dat", "cost_crew.dat", "pairings.dat");
```

```
Model craft ("aircraft_crew_recovery_script.mod",  "routes.dat", "arrival dept.dat",
"cost.dat", "trans.dat", "arr_dep_30.dat", "arr_dep_60.dat", "dest_org.dat");

repeat {
    crew.solve();
    time := time + crew.getTime();
    cout << "Crew recovery solution time : "<< crew.getTime()<<endl;
    forall ( i in crew.Flights)
    {
          crewno[i] := crew.noc[i];
    }

    craft.solve();
    time := time + craft.getTime();
    cout <<"Aircraft recovery solution time : "<< craft.getTime()<<endl;
    int nc;
    forall (i in Flights)
          Diff[i] := craft.cancel[i] - crew.cancel[i];
     nc := sum (i in Flights ) Diff[i];
      if (nc > 0) then {
           crew.reset();
           craft.reset();
      } else

          break;
} until 0;

cout<<"total time: "<<time<<endl;
cout<<endl;
cout << "Crew Recovery Solution at cost : " <<crew.objectiveValue() <<" at time:
"<< crew.getTime()<<endl;

cout << "Aircraft Recovery Solution at cost : " <<craft.objectiveValue() <<" at time:
"<< craft.getTime()<<endl;
float time2 := crew.getTime() + craft.getTime();

cout  << "TOTAL COMPUTATION TIME: " << time2 << endl;
cout <<"Crew Recovery Solution:" << endl;

crew.displaySolution ();
cout <<"Aircraft Recovery Solution:" << endl;
craft.displaySolution ();
```

APPENDIX J

THE OPL SCRIPT OF THE SOLUTION ALGORITHM S2

```
//*************************************************//
//Integrated Aircraft and Crew recovery problem
//*************************************************//
data "CP_arr_dept_pgs.dat";
data "CP_arr2_dept2_pgs.dat";
data "CP_flights_pgs.dat";
data "CP_city_pgs.dat";
data "CP_initial_crew_pgs.dat";
data "CP_initial_aircraft_pgs.dat";

Model cp ( "constraint_crew_pairings_recovery_crew2_pgs.mod",
"CP_arr_dept_pgs.dat", "CP_arr2_dept2_pgs.dat", "CP_dest_org_pgs.dat",
"CP_flights_pgs.dat", "CP_city_pgs.dat", "CP_crews_pgs.dat") editMode;

import enum Flights cp.Flights;
import enum Crews cp.Crews;
import enum City cp.City;

Model AcCp ( "constraint_aircraft_recovery_route2_pgs.mod",
"CP_arr_dept_pgs.dat", "CP_arr2_dept2_pgs.dat", "CP_dest_org_pgs.dat",
"CP_aircrafts_pgs.dat") editMode;
import enum Aircrafts AcCp.Aircrafts;

int seq;
int AcSeq;
int colCnt;
int NCVR    := 30;
float time:=0;
float time_total:=0;
float AcTime:=0;
float AcTime_total :=0;

Flights initial[Crews,1..4]:=...;
Flights initial2[Crews,1..2]:=...;
int inCost[Crews]:=...;
Flights AcInitial[Aircrafts,1..4]:=...;
Flights AcInitial2[Aircrafts,1..6]:=...;
int AcInCost[Aircrafts]:=...;

Open int pair[Crews,Flights, int+];
Open int colno[int+];
Open int noc[Flights];
```

```
Open int Diff[Flights];
Open int AcPair[Aircrafts,Flights, int+];
Open int AcColno[int+];

//Crew Shortages
int recBegin:= 780;
int recFinish:= 930;
City recSt := IST;
Crews shortage := C01;

//Aircraft Shortages
int AcRecBegin:= 1120;
int AcRecFinish:= 1270;
City AcRecSt := IST;
Aircrafts AcShortage := B800a12;

//*****************************************************//
//Crew Recovery Problem : Generate Routes
//*****************************************************//
forall(c in Crews){
  colno.addh();
  colno[colno.up]:=inCost[c];
    forall(i in 1..nseq)
    {
    Flights j :=initial[c,i];
    pair[c,j].addh();
    pair[c,j,pair[c,j].up] := colno.up;
    } } } }
forall ( f in Flights:cp.dept2[f]>=recBegin & cp.arr2[f]<= recFinish) {
if ( cp.org[f]<>cp.dest[f]) then{
cp.reset();
cp.cover := f;
cp.seq :=2;
colCnt :=0;
while cp.nextSolution()
do {
  forall ( c in cp.Crews : c<>shortage){
  colno.addh();
  colno[colno.up]:=cp.duty;
  forall ( i in cp.fltrange) {
    Flights j :=cp.fltseq[i];
    pair[c,j].addh();
    pair[c,j,pair[c,j].up] := colno.up;
```

```
      } }
      time := time +cp.getTime();
      colCnt := colCnt + 1;
      if (colCnt = NCVR) then break; }
cp.reset();
cp.cover := f;
cp.seq :=4;
colCnt :=0;
while cp.nextSolution()
do {
  forall ( c in cp.Crews: c<>shortage){
  colno.addh();
  colno[colno.up]:=cp.duty;
  forall ( i in cp.fltrange) {
     Flights j :=cp.fltseq[i];
     pair[c,j].addh();
     pair[c,j,pair[c,j].up] := colno.up;
     } }
    time := time +cp.getTime();
    colCnt := colCnt + 1;
    if (colCnt = NCVR) then break;
   }  } }
forall( f in Flights : cp.arr2[f]<=recBegin \/ cp.dept2[f]>=recFinish) {
if ( cp.org[f]<>cp.dest[f]) then{
cp.reset();
cp.cover := f;
cp.seq :=2;
colCnt :=0;
while cp.nextSolution()
do {
  forall ( c in cp.Crews){
  colno.addh();
  colno[colno.up]:=cp.duty;
  colCnt :=0;
  forall ( i in cp.fltrange) {
     Flights j :=cp.fltseq[i];
     pair[c,j].addh();
     pair[c,j,pair[c,j].up] := colno.up;
     } }
    time := time +cp.getTime();
    colCnt := colCnt + 1;
    if (colCnt = NCVR) then break;
```

```
    } } }

//*****************************************************//
//Aircraft Recovery Problem : Generate Routes
//*****************************************************//
forall(a in Aircrafts){
  AcColno.addh();
  AcColno[AcColno.up]:=AcInCost[a];
    forall(i in 1..nseq){ {
    Flights j :=AcInitial[a,i];
    AcPair[a,j].addh();
    AcPair[a,j,AcPair[a,j].up] := AcColno.up;
    } } }
forall ( f in Flights:AcCp.dept2[f]>=AcRecBegin & AcCp.arr2[f]<= AcRecFinish) {
if ( AcCp.org[f]<>AcCp.dest[f]) then{
AcCp.reset();
AcCp.cover := f;
AcCp.seq :=6;
colCnt :=0;
while AcCp.nextSolution()
do {
  forall ( a in AcCp.Aircrafts : a<> AcShortage){
  AcColno.addh();
  AcColno[AcColno.up]:=AcCp.duty;
  forall ( i in AcCp.fltrange) {
    Flights j :=AcCp.fltseq[i];
    AcPair[a,j].addh();
    AcPair[a,j,AcPair[a,j].up] := AcColno.up;
    } }
    AcTime := AcTime +AcCp.getTime();
    colCnt := colCnt + 1;
    if (colCnt = NCVR) then break; }
AcCp.reset();
AcCp.cover := f;
AcCp.seq :=4;
colCnt :=0;
while AcCp.nextSolution()
do {
  forall ( a in AcCp.Aircrafts : a<> AcShortage){
  AcColno.addh();
  AcColno[AcColno.up]:=AcCp.duty;
  forall ( i in AcCp.fltrange) {
```

```
    Flights j :=AcCp.fltseq[i];
    AcPair[a,j].addh();
    AcPair[a,j,AcPair[a,j].up] := AcColno.up;
    } }
    AcTime := AcTime +AcCp.getTime();
    colCnt := colCnt + 1;
    if (colCnt = NCVR) then break;
    } } }
forall( f in Flights : AcCp.dept2[f]<=AcRecBegin \/ AcCp.arr2[f]>=AcRecFinish) {
  if ( AcCp.org[f]<>AcCp.dest[f]) then{
  AcCp.reset();
  AcCp.cover := f;
  AcCp.seq :=2;
  colCnt :=0;
while AcCp.nextSolution()
do {
  forall ( a in AcCp.Aircrafts){
  AcColno.addh();
  AcColno[AcColno.up]:=AcCp.duty;
  forall ( i in AcCp.fltrange) {
    Flights j :=AcCp.fltseq[i];
    AcPair[a,j].addh();
    AcPair[a,j,AcPair[a,j].up] := AcColno.up;
    } }
    AcTime := AcTime +AcCp.getTime();
    colCnt := colCnt + 1;
    if (colCnt = NCVR) then break;
    } } }

//*************************************************//
//Crew Recovery Problem : Select Best Routes
//*************************************************//
int nc;
forall (i in Flights){
  Diff[i] := 0;
  noc[i] := 1;
}
Model sl ("constraint_crew_recovery_opt_2_pgs.mod", "CP_arr_dept_pgs.dat",
"CP_arr2_dept2_pgs.dat", "CP_dest_org_pgs.dat", "CP_assign_crew_pgs.dat");
Model AcSl ("constraint_aircraft_recovery_opt_crew_pgs.mod",
"CP_arr_dept_pgs.dat", "CP_arr2_dept2_pgs.dat", "CP_dest_org_pgs.dat",
"CP_assign_pgs.dat");
```

```
repeat{
Open Flights sc[sl.Columns,int+];
Open int num[sl.Columns,int+];
sl.solve();
time_total := time + time_total+sl.getTime();
AcSl.getLastIterationResult();
forall ( c in sl.Crews){
forall ( i in Flights) {
  forall(j in [pair[c,i].low..pair[c,i].up]:sl.choose[pair[c,i,j]] =1) {
  int k:=pair[c,i,j];
  sc[k].addh();
  sc[k,sc[k].up]:=i;
  num[k].addh();
  num[k,num[k].up] :=1;
  } } }

//*************************************************//
//Number of Available Crews
//*************************************************//
forall (f in Flights : cp.org[f] = IST) {
  noc[f] := sum(j in sl.Columns: sl.choose[j] =1, k in [sc[j].low..sc[j].up-2]:
              cp.org[sc[j,k]] = cp.org[f] &
              cp.arr2[sc[j,k]]+45 <=cp.dept2[f] &
              cp.dept2[sc[j,k+2]]<(cp.arr2[f]+90+cp.arr2[f]-cp.dept2[f]))
               num[j,k]+
          sum(c in sl.Crews, j in [pair[c,f].low..pair[c,f].up]) sl.choose[pair[c,f,j]] +
           sl.reserve[f] - sl.cancel[f] ;
}
forall (f in Flights : cp.dest[f] = IST) {
 noc[f] := 0;
 noc[f] := sum(j in sl.Columns: sl.choose[j] =1, k in [sc[j].low+1..sc[j].up-1]:
              cp.org[sc[j,k-1]] = cp.dest[f] &
              cp.arr2[sc[j,k-1]] <=cp.dept2[f]-90-(cp.arr2[f]-cp.dept2[f])&
              cp.dept2[sc[j,k+1]]<(cp.arr2[f]+45))
               num[j,k]+
          sum(c in sl.Crews, j in [pair[c,f].low..pair[c,f].up]) sl.choose[pair[c,f,j]] +
           sl.reserve[f] - sl.cancel[f] ;
}
AcSl.solve();
AcTime_total := AcTime + AcTime_total + AcSl.getTime();
forall (i in Flights)
```

```
    Diff[i] := AcSl.cancel[i] - sl.cancel[i];
    nc := sum (i in Flights ) Diff[i];
    if (nc > 0) then {
    sl.reset();
    AcSl.reset();
  }else  break;  } until 0;

//*****************************************************//
//Display Solution
//*****************************************************//
Open Flights sol[sl.Columns,int+];
Open Crews cr[sl.Columns, int+];
Open int num[sl.Columns,int+];
AcSl.getLastIterationResult();
forall ( c in sl.Crews){
forall ( i in Flights) {
  forall(j in [pair[c,i].low..pair[c,i].up]:sl.choose[pair[c,i,j]] =1) {
  int k:=pair[c,i,j];
  sol[k].addh();
  sol[k,sol[k].up]:=i;
  cr[k].addh();
  cr[k,cr[k].up]:=c;
} } }
Open Flights AcSol[AcSl.Columns,int+];
Open Aircrafts ac[AcSl.Columns, int+];
AcSl.getLastIterationResult();
forall ( a in AcSl.Aircrafts){
forall ( i in Flights) {
  forall(j in [AcPair[a,i].low..AcPair[a,i].up]:AcSl.choose[AcPair[a,i,j]] =1) {
  int k:=AcPair[a,i,j];
  AcSol[k].addh();
  AcSol[k,AcSol[k].up]:=i;
  ac[k].addh();
  ac[k,ac[k].up]:=a;
  } } }
cout<<"SOLUTION:"<<endl;
cout<<"Total solution time: "<<time_total+AcTime_total<<endl;
cout<<endl;
cout<<"CREW RECOVERY PROBLEM SOLUTION"<<endl;
cout<<"Total number of new pairings: "<<colno.size<<" have been found in time
"<<time<<endl;
float Obj := sl.objectiveValue();
```

```
cout<<"Objective Value = "<<Obj<<endl;
cout<<"Solution Time = "<<time_total<<endl;
cout<<endl;
forall( i in Flights){
if(sl.cancel[i]=1)then{
cout<<"Cancelled Flights: "<<i<<endl; }
if(sl.reserve[i]=1)then{
cout<<"Flight: "<<i<<" is flown by reserve crew"<<endl;   } }
forall ( j in sl.Columns: sl.choose[j] =1){
  cout<<endl;
  cout<<"Crew "<<cr[j,0]<<" uses route "<<j+1<<endl;
  forall(k in [sol[j].low..sol[j].up]){
        cout<<sol[j,k]<<" ";
        City org :=cp.org[sol[j,k]];
        City dest :=cp.dest[sol[j,k]];
         int dep :=cp.dept[sol[j,k]];
         int arr :=cp.arr[sol[j,k]];
         int depH :=(dep/60) mod 12;
         int arrH :=(arr/60) mod 12;
         int depM := dep mod 60;
         int arrM := arr mod 60;
         string depZ := "";
         string arrZ :="";
         string depAP :=" pm";
         string arrAP :=" pm";
         if (depH = 0) then depH :=12;
         if (arrH = 0) then arrH :=12;
         if (depM < 10) then depZ := "0";
         if (arrM < 10) then arrZ := "0";
         if (dep <720 ) then depAP := " am";
         if (arr <720 ) then arrAP := " am";

        cout<<"From "<<org<<" to "<<dest<<" at ";
        cout<<depH<<":"<<depZ<<depM<<depAP;
        cout<<" --> ";
        cout<<arrH<<":"<<arrZ<<arrM<<arrAP;
        cout<<endl;
        }
int duty :=max (k in [sol[j].low..sol[j].up])cp.arr2[sol[j,k]]
      - min (k in [sol[j].low..sol[j].up])cp.dept2[sol[j,k]];
int flight := sum(k in [sol[j].low..sol[j].up])(cp.arr2[sol[j,k]]- cp.dept2[sol[j,k]]);
cout<<"Duty time = "<<duty;
```

```
cout<<", Flight time = "<<flight<<endl;
}
cout<<endl;
cout<<"AIRCRAFT RECOVERY PROBLEM SOLUTION"<<endl;
cout<<"Total number of new pairings: "<<AcColno.size<<" have been found in time
"<<AcTime<<endl;
float AcObj := AcSl.objectiveValue();
cout<<"Objective Value = "<<Obj<<endl;
cout<<"Solution Time = "<<AcTime_total<<endl;
cout<<endl;
forall( i in Flights){
if(AcSl.cancel[i]=1)then{
cout<<"Cancelled Flights: "<<i<<endl;
} }
forall ( j in AcSl.Columns: AcSl.choose[j] =1){
  cout<<endl;
  cout<<"Aircraft "<<ac[j,0]<<" uses route "<<j+1<<endl;
  forall(k in [AcSol[j].low..AcSol[j].up]){
    cout<<AcSol[j,k]<<" ";
        City org :=AcCp.org[AcSol[j,k]];
        City dest :=AcCp.dest[AcSol[j,k]];
        int dep :=AcCp.dept[AcSol[j,k]];
        int arr :=AcCp.arr[AcSol[j,k]];
         int depH :=(dep/60) mod 12;
         int arrH :=(arr/60) mod 12;
         int depM := dep mod 60;
         int arrM := arr mod 60;
        string depZ := "";
        string arrZ :="";
        string depAP :=" pm";
        string arrAP :=" pm";
         if (depH = 0) then depH :=12;
        if (arrH = 0) then arrH :=12;
        if (depM < 10) then depZ := "0";
        if (arrM < 10) then arrZ := "0";
        if (dep <720 ) then depAP := " am";
        if (arr <720 ) then arrAP := " am";

        cout<<"From "<<org<<" to "<<dest<<" at ";
        cout<<depH<<":"<<depZ<<depM<<depAP;
        cout<<" --> ";
        cout<<arrH<<":"<<arrZ<<arrM<<arrAP;
```

```
          cout<<endl;  }
int AcDuty :=max (k in [AcSol[j].low..AcSol[j].up])AcCp.arr2[AcSol[j,k]]
         - min (k in [AcSol[j].low..AcSol[j].up])AcCp.dept2[AcSol[j,k]];
int AcFlight := sum(k in [AcSol[j].low..AcSol[j].up])(AcCp.arr2[AcSol[j,k]]-
AcCp.dept2[AcSol[j,k]]);
cout<<"Duty time = "<<AcDuty;
cout<<", Flight time = "<<AcFlight<<endl;   }
```