METHODOLOGY FOR CREW-PAIRING PROBLEM IN AIRLINE CREW SCHEDULING

by

Uğur Özdemir

B.S., Computer Science Engineering, Marmara University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in System and Control Engineering
Boğaziçi University
2009

METHODOLOGY FOR CREW-PAIRING PROBLEM IN AIRLINE CREW SCHEDULING

APPROVED BY:

Assoc. Prof. Tunga Güngör . . . .
(Thesis Supervisor)

Prof. Ahmet Ademoğlu . . . .

Dr. Tamer Şıkoğlu . . . .

DATE OF APPROVAL: 27.01.2009

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Tamer Şıkoğlu for his invaluable guidance and help during the preparation of this dissertation. I would like to mention his patience, giving me inspiration and hope when I was stuck at dead-ends.

I would like to thank to my family and my friend Engin Akar for their endless support and help.

ABSTRACT

METHODOLOGY FOR CREW-PAIRING PROBLEM IN AIRLINE CREW SCHEDULING

Crew scheduling problem is divided into two sub problems, crew pairing and crew assignment problems. In literature, there are many studies on crew pairing problem and in this study also main subject is crew pairing problem. In this thesis, previous work on crew pairing problem is investigated and a hybrid method is developed by combining previous methods.

In this study, the mathematical representation of the crew pairing problem is explained. Basically, the problem is represented as integer programming problem but to reduce the complexity it is relaxed to linear programming (LP) problem. Most well-known method for solving large size of LP is sprint method. Also in previous studies, another method which is called dynamic pairing generation is developed. Commonly for finding integer values after solving LP, branch and bound or branch on follow-on method is used. Also, Carmen algorithm is another method for getting integer values after finding LP solution. In hybrid method, for solving LP problem sprint method is used. For finding integer values, after solving LP problem Carmen algorithm is implemented. Experiments show that Carmen algorithm is very fast but it gives bad results. Therefore, another version of hybrid method is implemented which also starts with sprint method and uses branch on follow-on method to find integer solutions. In the last section, the comparison of the test results between Carmen algorithm and branch on follow-on method is given. As a conclusion a brief summary on hybrid method is explained and for future work is given.

ÖZET

HAVAYOLU MÜRETTEBAT ZAMAN ÇİZELGESİNDE UÇUŞ DİZİSİ BULMA PROBLEMİ İÇİN METODOLOJİ

Mürettebat zaman çizelgesi problemi iki alt probleme bölünür, uçuş dizisi bulma problemi ve mürettebat atama problemi. Literatürde uçuş dizisi bulma problemi üzerinde çok çalışma vardır ve bu çalışmanın da ana konusu budur. Bu çalışmada önceki metotlar incelenerek hibrid bir metod geliştirilmiştir.

Bu çalışmada uçuş dizisi bulma probleminin matematiksel ifade ediliş şekli açıklandı. Temel olarak problem tamsayı programlama problemi şeklinde ifade edilir fakat problemdeki karmaşıklığı azaltmak için doğrusal programlama problemine çevriliyor. Doğrusal programlama problemlerini çözmek için kullanılan en iyi bilinen metot Sprint methodudur. Ayrıca önceki çalışmalarda adı dinamik uçuş dizisi üreten olan bir metot geliştirildi. Genellikle doğrusal programlama problemi çözüldükten sonra tamsayı değerleri bulmak için parçala ve sınırla veya art arda uçuşlara göre parçala yöntemi kullanılıyor. Ayrıca doğrusal programlama problemini çözdükten sonra tamsayı değerleri bulmak için kullanılan diğer bir metot da Carmen algorithmasıdır. Hibrid metotda doğrusal programlama problemini çözmek için Sprint metodu kullanıldı. Doğrusal programlama problemi çözüldükten sonra tamsayı değerleri bulmak için Carmen algorithması uygulandı. Deneyler gösterdiki Carmen algoritması çok hızlı fakat çok kötü sonuçlar veriyor. Bu nedenle yine Sprint metodu ile başlayan ve tamsayı değerler bulmak için art arda uçuşlara göre parçala yöntemini kullanan bir başka hibrid versiyonu uygulandı. En son kısımda Carmen algoritması ile art arda uçuşlara göre parçala yönteminin deneylerde çıkan sonuçlarının karşılşatırması verildi. Sonuç olarak ise hibrid metodunun kısa bir özeti açıklandı ve gelecekte yapılabilecek çalışmalar verildi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In airline industry, there are many planning problems at strategic, tactical and operational levels. Airlines spend incredible amount of money on researching and resolving these kinds of problems. Most common research area is *Operation Research* (OR) techniques. They apply these techniques to their planning problems and try to resolve them by representing problem as mathematical optimization. The main idea on working on these problems is of course in order to decrease their cost and increase their market share. There are four common steps in the airline planning process. These are timetable, fleet assignment, crew pairing and crew assignment.

- Timetable: First a timetable is created. In the timetable, flights are scheduled based on market demands and competitive analysis. As air traffic is increasing, allocation of time slots is difficult task to do.

- Fleet Scheduling: With given timetable, aircrafts are assigned to the flights. There are several types of aircrafts in airlines and aircrafts are divided into separate fleets depending on aircraft type. Aircraft maintenance and service is the most important task in this step. Also aircrafts visit maintenance depot for checkup and service within regular periods.

- Crew Pairing: When the timetable is constructed and aircrafts are assigned to the flights next step is considering crews. Typically a crew is composed of a pilot, co-pilot and a number of flight attendants. For every type of aircraft there are different numbers of crews for each flight. A crew pairing is one or several days long and they should be checked based on rules and regulations. In this step crew pairings are selected such that they should cover all the flights to be planned.

- Crew Assignment: In this step, crews are assigned to given pairings. Most airlines use a kind of bidding system to assign pairings to crews. It is a complicated step and based on strict seniority. It can also include an optimization step where the objective is some kind of total “fairness”. Also there are crews to be a backup for other crews in case of illness or other disruptions to the schedule.

Generally, a crew scheduling problem is divided into two problems one is crew pairing and the other one is crew assignment. First crew pairing problem is solved and good pairings are found by giving flight schedule (timetable) as an input. After solving crew pairing problem, crews are assigned to these pairings. To get a good final solution the input data (crew pairings) has to have a good quality. Good quality of input data does not imply good outcome but with bad input data feasible solution is impossible. That is why crew pairing is the most important step and saves more money in crew scheduling problem. In literature, there are many studies on crew pairing problem. In this study also after investigating previous studies, a hybrid method is developed for finding best crew pairings.

In section 1 crew pairing problem is described and its mathematical model is given. In section 2, restriction rules on pairings and a few mathematical concepts is presented. In section 3 first the algorithm for pairing generation is given and then previous methods are explained and a few examples are shown. In the next section hybrid method is explained and its advantages and disadvantages are presented. In the experiments section, observed experiment results are given. In section 5, a summary on hybrid and previous methods is presented.

2. PROBLEM DESCRIPTION

Before explaining the problem a few definitions should be given. A flight leg or segment is a single nonstop flight. A duty period - mostly a working day of a crew - consists of a sequence of flight legs with short rest periods or sits separating them. Also the duty starts with a brief period and ends with a debrief period. A pairing is a sequence of duties and each pairing begins and ends at the same crew base. In a pairing there is an overnight rest between each duty. Crew base is a city where crews are stationed. To reposition a crew from one base to another base, a pairing might include crews as passengers and this kind of flight is called deadhead. Generally deadheads are used to transport a crew where they are needed to cover a flight or to return to their home base.

Typically, the given timetable which is generally schedule of a month to the crew pairing problem can contain daily, weekly and monthly flights. The crew pairing problem can also be defined as daily, weekly and monthly (transition) problem to handle all kinds of flights in timetable. The daily problem assumes that all flights are flown every day. Every pairing is flown by a different crew and it starts each day of the week. Also in daily problem, a flight can not appear more than once in a pairing. To consider flights that are not operating every day of the week, modified version of daily problem is used and it is called weekly (exceptions) problem. There are also other flights different than daily and weekly flights in monthly schedule. Airlines might modify some of the flights that were already in the timetable. In this case, transition (monthly) problem is used to find crew schedule for covering flights during a transition between old and new flight schedule. In this study, only daily problem will be considered.

In the crew pairing problem the objective is finding a set of pairings from all possible pairings. While selecting pairings all flights must be covered exactly once and cost of the all selected pairings should be kept in the minimum. Also selected pairings should be legal according to the specified regulations. When we model this optimization it can be considered as an integer programming problem. Pairings are the variables of the problem and they can only get values 0 (not selected) or 1 (selected). Cost of the pairings is the coefficient of the variables and covering flights exactly once represents the constraints of the optimization. Therefore,

$$\text{Minimize} \sum_{j \in P} c_j \quad x_j \qquad (2.1)$$

$$\text{subject to} \quad x_j \in \{0, 1\} \qquad \text{for all } j \in \text{P}$$

$$\sum_{j \in P^i} x_j = 1 \qquad \text{for all } i \in \text{F}$$

where **F** is the set of all flights, **P** is the set of all possible pairings, for each $i \in$ **F, P$^{\mathbf{i}}$** is the set of pairings which cover the flight *i,* c_j is the cost of the pairing $j \in$ **P**. Let **A** = $\{a_{ij} : i \in \mathbf{F}, j \in \mathbf{P}\}$ denote the binary matrix where each row represents a flight leg and each column represents a pairing, $a_{ij} = 1$ if the flight leg *i* is covered by pairing *j*, and 0 otherwise. x_j is a binary variable assuming a value of 1 if pairing *j* is selected and 0 otherwise. The objective is to minimize the total cost of selected pairings. The set of equality constraints guarantees that each flight is covered only once. Here is a simple example for this optimization.

Consider the following flights;

Flight 1: City A – City B 08:00 – 09:00
Flight 2: City B – City C 10:00 – 11:00
Flight 3: City C – City D 13:00 – 14:00
Flight 4: City C – City A 07:00 – 08:00
Flight 5: City D – City A 07:00 – 08:00
Flight 6: City A – City B 17:00 – 18:00
Flight 7: City B – City C 11:00 – 12:00

And suppose the pairings are

$\mathbf{P_1} = \{F_1\ F_2\ F_4\}$	and $c_1 = 4$
$\mathbf{P_2} = \{F_1\ F_3\ F_5\ F_7\}$	and $c_2 = 4$
$\mathbf{P_3} = \{F_2\ F_3\ F_5\ F_6\}$	and $c_3 = 4$

P₄ = {F$_4$ F$_6$ F$_7$} and $c_4 = 4$

P₅ = {F$_7$ F$_4$ F$_7$} and $c_5 = 5$

When we put the values to the model;

Table 2.1 The model with the values

min $4x_1 + 4x_2 + 4x_3 + 4x_4 + 5x_5$				A will be				
$x_1 + x_2 + x_5$	=	1	(flight 1)	1	1	0	0	1
$x_1 + x_3$	=	1	(flight 2)	1	0	1	0	0
$x_2 + x_3$	=	1	(flight 3)	0	1	1	0	0
$x_1 + x_4 + x_5$	=	1	(flight 4)	1	0	0	1	1
$x_2 + x_3$	=	1	(flight 5)	0	1	1	0	0
$x_3 + x_4$	=	1	(flight 6)	0	0	1	1	0
$x_2 + x_4 + x_5$	=	1	(flight 7)	0	1	0	1	1

An integer programming problem is a NP-hard but many methods are developed to solve these kinds of problems. A few examples to them are branch and bound method, branch and cut method and branch and price method. Since there many millions of variables, applying these methods directly to crew pairing problem are not the efficient way. Mostly an intermediate step is built up to select best variables first then these methods are applied to find integer values. The intermediate step is relaxing the integer programming problem to the linear programming problem and solving it. That is, IP problem is transformed to the LP problem. Considering crew pairing problem with this transformation, $x_j = 0$ or 1 is relaxed to $0 \leq x_j \leq 1$ and therefore x_j values can be fractional which means that a flight segment may be covered by fractional values of two or more pairings.

$$\text{Minimize} \sum_{j \in P} c_j \quad x_j \tag{2.2}$$

$$\text{subject to} \quad 0 \leq x_j \leq 1 \qquad \text{for all } j \in \mathbf{P}$$

$$\mathbf{A}x = 1$$

With this relaxation, *xj* takes fractional values but in practice it is unusable. So to find integer solution, after solving linear programming problem, special version of the branch and bound method which is branch on follow-ons method is used. In figure 2.1, another very small example for pairing problem is shown.

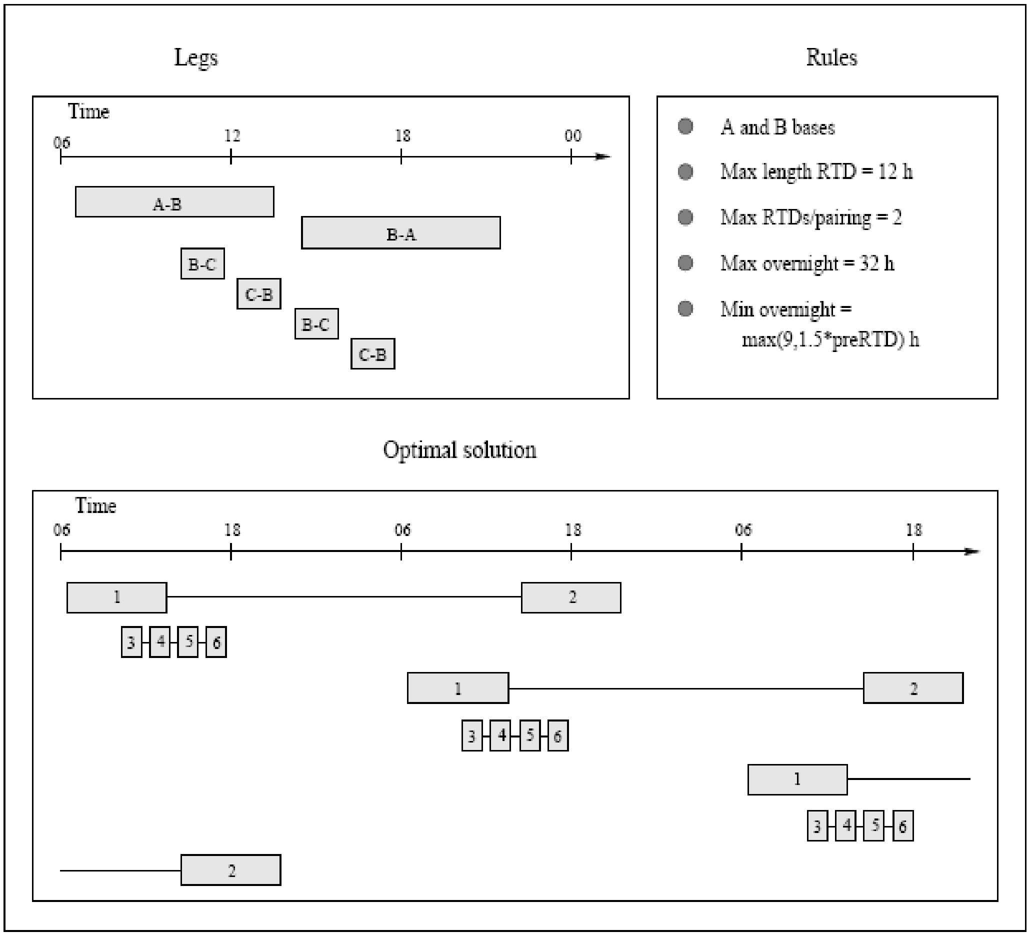


Figure 2.1 A sample example for crew pairing problem [1]

2.1. The Restriction Rules on Pairings

The FAA, operational considerations, and contractual restrictions define the structure and cost of legal duty periods and pairings.

Safety considerations set down some rules like *maxsit* and *minsit,* which define maximum and minimum sitting times between flight segments within duty times. The elapsed time of a duty period called *elapse.* The brief and debrief periods are included in *elapse* and elapse time cannot be greater than *maxelapse* value. In a duty period, *fly*, the total number of hours of actual flying time, must be less than *maxfly* time which is a maximum value. The maximum value of *mg1* (minimum number of hours), *f1* (fraction) times the elapsed time of duty, and the actual flying time is defined as the cost structure for duty periods (b_d). And it can be expressed as;

$$b_{d\,=}\, max\ \{\ mg1, f1\ *\ elapse, fly\ \}. \tag{2.3}$$

Typical values for domestic flights of the defined rules are given below. Also in some cases these values can be changed. Maxsit = 4 hours, minsit = 0.5 hours, maxelapse = 12 hours, maxfly = 8 hours, mg1 = 3 hours, and *f1* = 4/7.

Legal pairings may be composed of up to a maximum number of duty periods called *maxduties* and minimum number of hours of rest between duties called *minrest*. After long *maxfly* or inadequate *minrest*, *compensotory rest* can be given. The maximum value of mg2 (minimum guarantee) times NDP (times the number of duties in pairing), f2 (fraction) times the total elapsed time of the pairing (TAFB) and the sum of the cost of the individual duties gives the cost of a pairing.

$$c_p = \max\{\text{NDP} * mg2, f2 * \text{TAFB}, \Sigma_{d€p}\, b_d\ \} \tag{2.4}$$

Typical values are *maxduties* = 2, *mg2* = 4.75 hours, *f2* = 2/7 hours. There are also other obligations other than pairing and duty rules like crew base constraints. For every crew base, there is a different value of the total amount of flying hours that is assigned to the crew. With this rule, it is ensured that crews at the various bases will all have approximately the same number of flight hours for each work month. In this study this rule will not be considered.

2.2. Dual Variables and Reduced Cost

Before explaining previous methods that were used to solve crew pairing problem, a few mathematical concepts should be clarified. These are not the main purpose of this study but to clearly understand previously applied methods a brief definition should be specified.

When considering the complexity of an optimization problem, constraints are playing an important role in solving an optimization problem. Without constraints an optimization problem is easy to solve. A method called Lagrangian relaxation is developed to simplify optimization problems by moving constraints to the objective function.

$$\min \boldsymbol{cx} \tag{2.5}$$

$$\mathbf{A}\boldsymbol{x} = \mathbf{b}$$

becomes

$$\min \boldsymbol{cx} + \boldsymbol{y}(\boldsymbol{b} - \boldsymbol{Ax}) \tag{2.6}$$

where y values are the lagrangian multipliers of the lagrangian function. One of the mathematical concepts that will be mentioned through this study is the dual variable also known as the shadow price. Dual variable represents the "cost" of having a constraint in the optimization model and it is associated with each constraint. If a constraint is removed from the optimization problem, the optimum value will be improved based on the value of the dual variable of that constraint. More formally, the dual variable is the Lagrange multiplier at the optimal solution which means that if there is a small change in a constraint, it will cause a small change in the optimum solution. Also, with lagrangian function, if a constraint is not satisfied, the penalty can be implied into the objective value.

By using dual variables, reduced cost of a variable can be calculated and this is the main purpose of finding dual values in crew pairing problem. Reduced cost is the value that if a variable whose value is currently 0 is forced to be part of the solution (set to 1), the optimum value will be changed by the amount of the reduced cost of that variable. For example if there is a minimization problem, let x_j is equal to 0 and its reduced cost is negative. If x_j is forced to be 1 meaning to be the part of the solution, the optimum value will decrease (improve) by the amount of the reduced cost value of the variable. For minimization problem if a reduced cost is positive, optimum value will be increased (worsened). If the reduced cost is zero, deciding if that variable should be included into the solution or not is impossible. Reduced cost can be calculated as;

$$\mathbf{c} - \mathbf{y}\boldsymbol{a}_j \qquad (2.7)$$

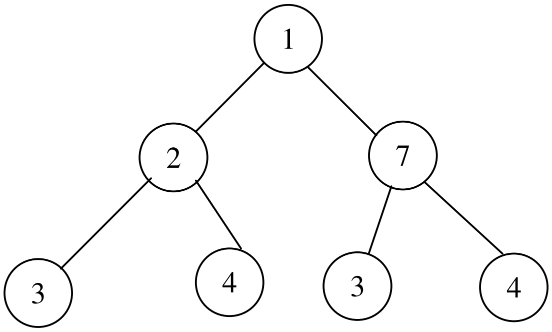
where c is the coefficient of the variable, y is the row vector of the dual variables and a_j is the j^{th} column of the A matrix.

Consider the example given in section Problem Description without the fifth pairing, if it is solved with LP solver the objective value is 8 and the variables will be x_1 = 0. 5, x_2 = 0. 5, x_3 = 0.5 and x_4 = 0. 5. Also for constraint 3 and 4 dual variables will be 4 and for others it will be 0. The zero dual variables mean that there will be no change in the objective even after changing or removing the corresponding constraints. When we calculate the reduced cost with given formula reduced cost for fifth pairing will be 1. This means that if it is included into the problem, the objective value will be increased by 1. After setting fifth variable to 1, it is seen that the objective value is 9 and the values are x_3 = 1 and x_5 = 1. Actually in this example the optimum value is increased but integer solution is obtained. In this example, the effect of using reduced cost to determine which pairing should be included into the problem is shown.

3. PREVIOUS WORK

3.1. Pairing Generation

In pairing generation stage, first all possible duty periods are generated and after generation of duties, all possible pairings are generated by combining these duties. In duty generation for each flight segment, a tree is constructed whose root node represents that flight segment. The root node has children that represent every possible connecting flight segments. At each successive level in the tree, each node has a child for every possible connection. The depth of the tree can be determined by the maximum allowable number of flights in a duty. With depth-first-search approach, visiting every node of the tree all possible duty periods are found. Each path from the root node to any of its descendants (including only itself) in the tree represents a duty period. Possible duty periods are first checked to ensure that they are legal based on the legality restrictions. If a duty is legal its cost is also calculated. A sample flight tree for flight 1 in the example 1 is in the figure 3.1.



Duty 1: F_1, F_2, F_3

Duty 2: F_1, F_2, F_4

•

•

•

Figure 3.1 A sample flight tree for flight 1

Pairings can also be found by following the same procedure. In this case tree is constructed for all duty and duty periods are represented as nodes in the tree. Also again all possible pairings are checked not to violate regulation rules but only the rules that can not be applied to duties because all applicable rules are already checked while generation of duties. One of the examples to those rules is that a pairing should starts and ends at the same crew base. Also a pairing should not contain a flight twice so any duty can not be followed by another duty that has the same flight with the first one. If a pairing is legal then its cost is also calculated with given formula (2.4).

A recursive method is implemented for finding child nodes in the flight and the duty trees. The table 3.1 shows the pseudocode for the recursive method for flight tree;

Table 3.1 Pseudocode for the recursive method for flight tree

```
Step 1: For all flights
        Step 2: Set flight as root node
        Step 3: Find child nodes
        do
                if it is a possible connection
                     Add flight as a child node
                     Find child nodes of the child node (go to step 3)
        Enddo
Step 4: Stop. After visiting every flight tree is constructed.
```

3.2. Solving LP Problem

3.2.1. Sprint Method

The SPRINT method has been introduced by J. Forrest and was described in [2]. It is a successful method for large scale crew scheduling LP's. The main approach on this method is finding optimal solution of a large scale LP by solving small subset of LP's iteratively.

In sprint method, first all possible pairings are generated (most of the time there are many millions of pairings). Since solving all of these pairings at one time is impossible, iteratively a bunch of pairings (usually 5000) are selected and solved. The iteration first begins with generating an initial solution. By using initial optimum dual variables, all reduced cost of the pairings is calculated and based on their reduced costs pairings are selected to be included into the next sub problem. The idea behind using reduced cost is that if the pairing that has smaller negative reduced cost is included into the problem, the objective value will be improved (reducing the cost). Therefore, the parings that has smaller negative reduced cost have highest priority to be selected. The important factor in here is that pairings are selected over all of the pairings every time a new sub problem is created. In the next iteration, selected small subset is solved and again by using new dual variables reduced cost is calculated to be used in the next iteration. The iteration ends when there is no negative reduced cost pairing exists.

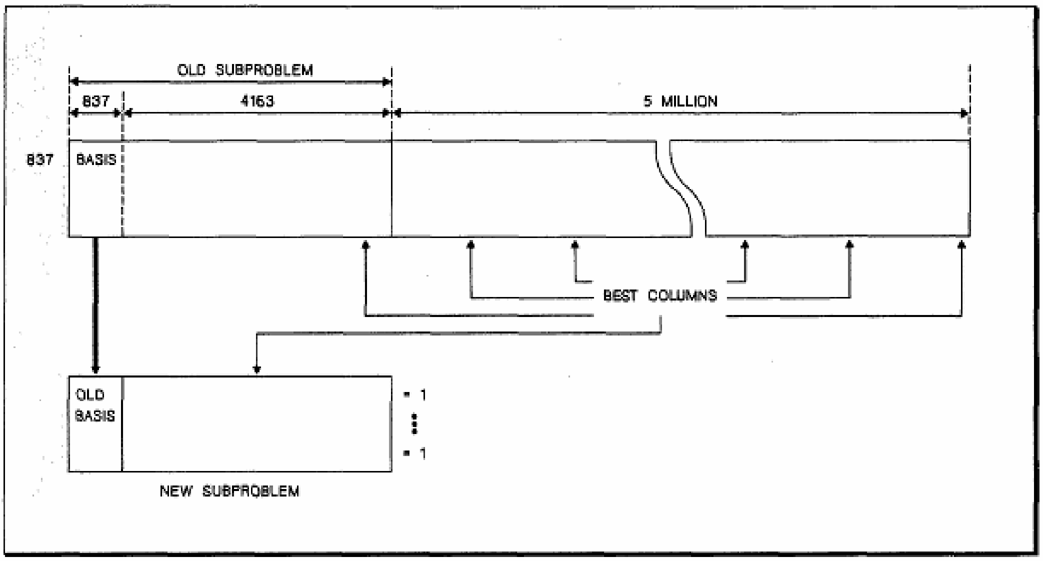


Figure 3.2   Selection process in Sprint method [2]

In pairing selection process, first all pairings are divided into “buckets” based on their reduced costs. New sub problem is filled up by selecting from the best buckets first. But if the required number of pairings is exceeded, pairings are selected from other buckets randomly. The next subset of columns consists of, the current optimal basis, pairing (columns) in the best “bucket” and a random selection of columns from the other buckets. In figure 3.2 a sample selection process is shown.

Since remarkable small number of sub problems is solved in order to achieve globally optimum value for large problems, sprint method has better performance than other methods. Even for large scale problems i.e. 5.5 million pairings only 25 sub problems are solved [2]. If 5000 pairings are chosen, only 25 x 5000 columns out of 5.5 million are used for solving the global problem [2]. Also since only 5000 columns are selected at each iteration, sub problems are solved relatively quickly. The pseudo code of the sprint method is in table 3.2.

Table 3.2 Pseudocode of the Sprint method

```
Step 1: Find and initial solution.
Step 2: Using initial dual variables calculate reduced cost
Step 3: Generate a sub problem by selecting pairings based on their reduced costs
Step 4: Solve the sub problem.
Step 5: Calculate reduced costs
Step 6: If there is no negative reduced cost
            Stop. Optimal solution is found for LP problem.
Step 7: Else go to Step3
```

3.2.2. A Heuristic Approach for the Airline Crew Pairing Problem

In this method, pairing generation and solving the problem is not separated strictly as in sprint method. First a set of pairings are generated and this small subset is solved. If the there is no feasible solution for this sub problem artificial pairings with high cost are used. The dual values from the optimum solution to this small subset are then used to generate more pairings and generated subset is solved again. The iteration continues until finding an optimum solution. Solving these sub problems is named as restricted master problem (RMP).

For pairing generation, first a structured network is constructed using flights or duties. From constructed network, by solving constrained shortest path problem new pairings are generated. There are two types of networks that is used in the literature, one is flight network and the other one is duty network.

In a flight network, one type of arc represents each flight and the other type of arc represents the legal connection between flights. Also there are two types of nodes, one is source or destination of the flight and the other one is the sinks where pairings either start or end. A sample flight network is shown in figure 3.3.

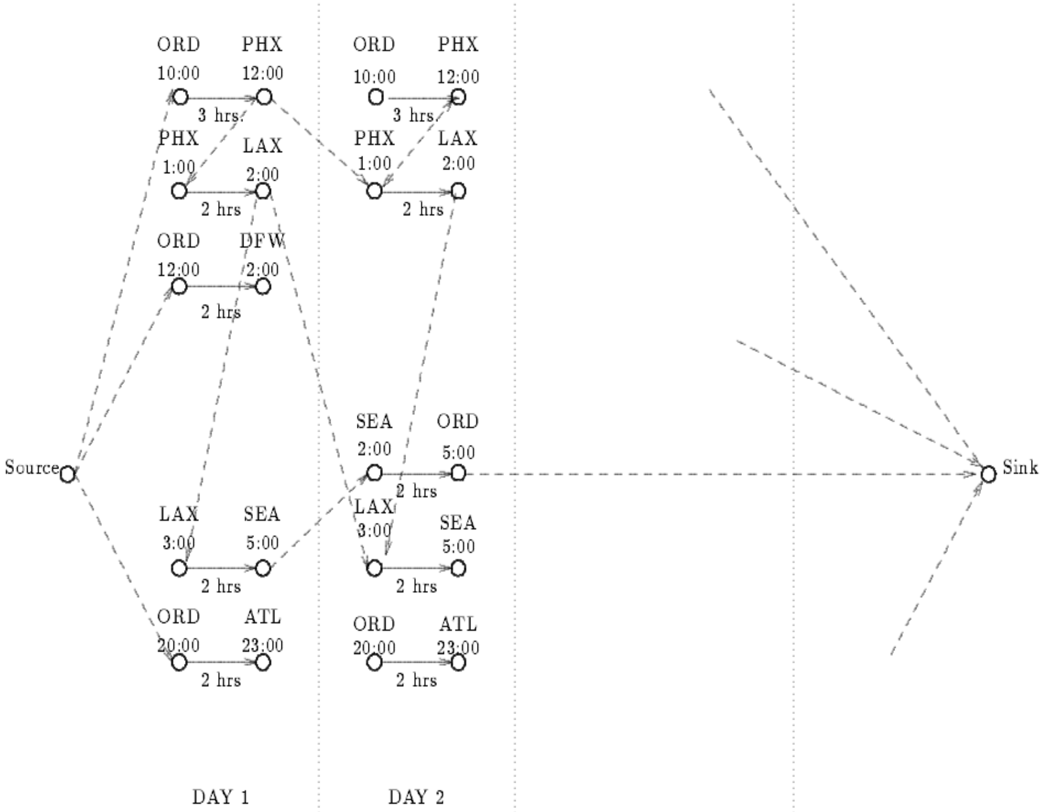


Figure 3.3 A sample flight network [3]

For daily problems, each flight is replicated as many times as the calendar allows. If it is weekly problem, each flight is replicated twice so that pairings that cross over from the end of the week (from Sunday to Monday) can be generated. As shown in the figure, the source node is connected to the departure node of each flight if it is the specified crew base and also the arrival node of the each flight is connected to the sink node if it ends at the specified crew base. Also each arc contains additional attributes to use in calculation of cost or checking legality of a pairing. These attributes are arrival, departure and flying time of a flight and dual values of the current LP solution to the restricted master problem.

The duty network has the same types of arcs and nodes and also it is constructed by using the same rules and attributes in flight network. A sample duty network is shown in figure 3.4.

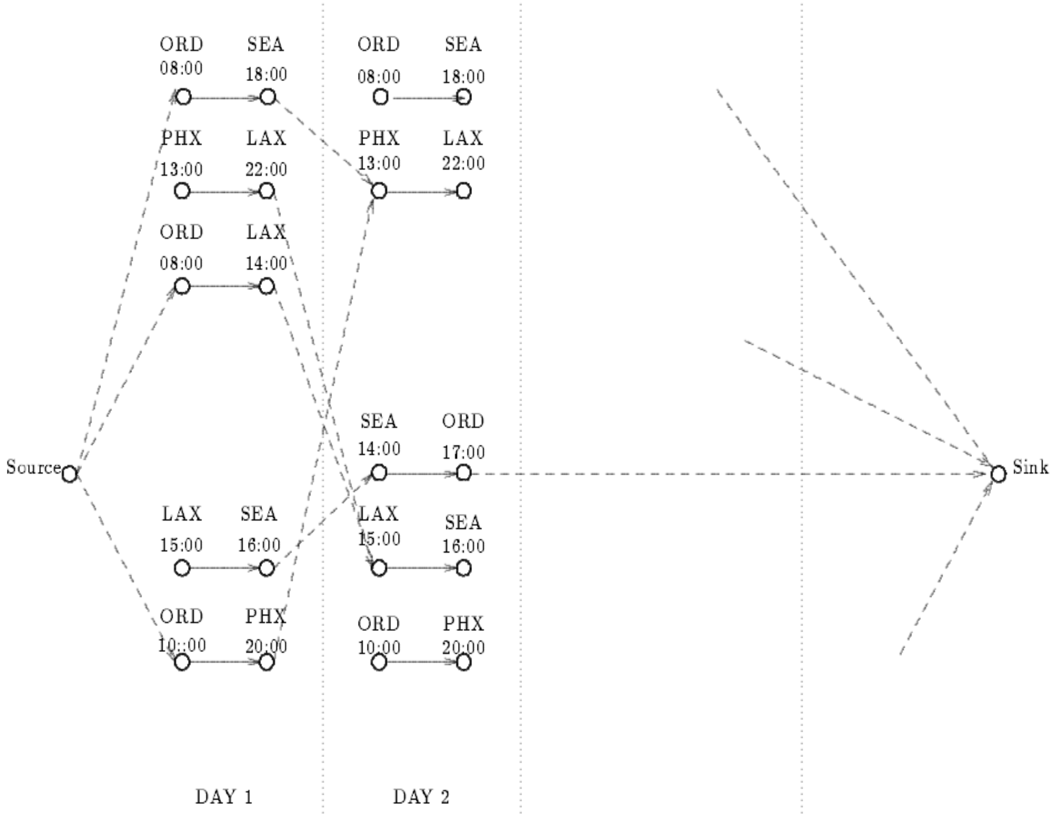


Figure 3.4 A sample duty network [3]

In flight and duty networks all paths represent a possible pairing. Since there many possible paths in the networks, an "intelligent" way are used to determine which pairings should be chosen for constructing a sub problem. The important factor behind this idea is generating pairings such that which can reduce the overall cost of the problem. This algorithm is called pricing step. In this step, dual values of the LP solution to previous sub problem is used to calculate reduced cost of all candidate pairings and based on the reduced cost, pairings are selected. The pairings that have negative reduced cost have higher priority than others to be selected (priced-out). Because as described in section 2.2, if the pairing has negative reduced cost, including that pairing into the problem will reduce the overall cost. Also besides considering the reduced cost, shortest path algorithm is used in networks to find pairings which have lower cost. An overall algorithm of the dynamic pairing generation and pricing step is shown in figure 3.5.

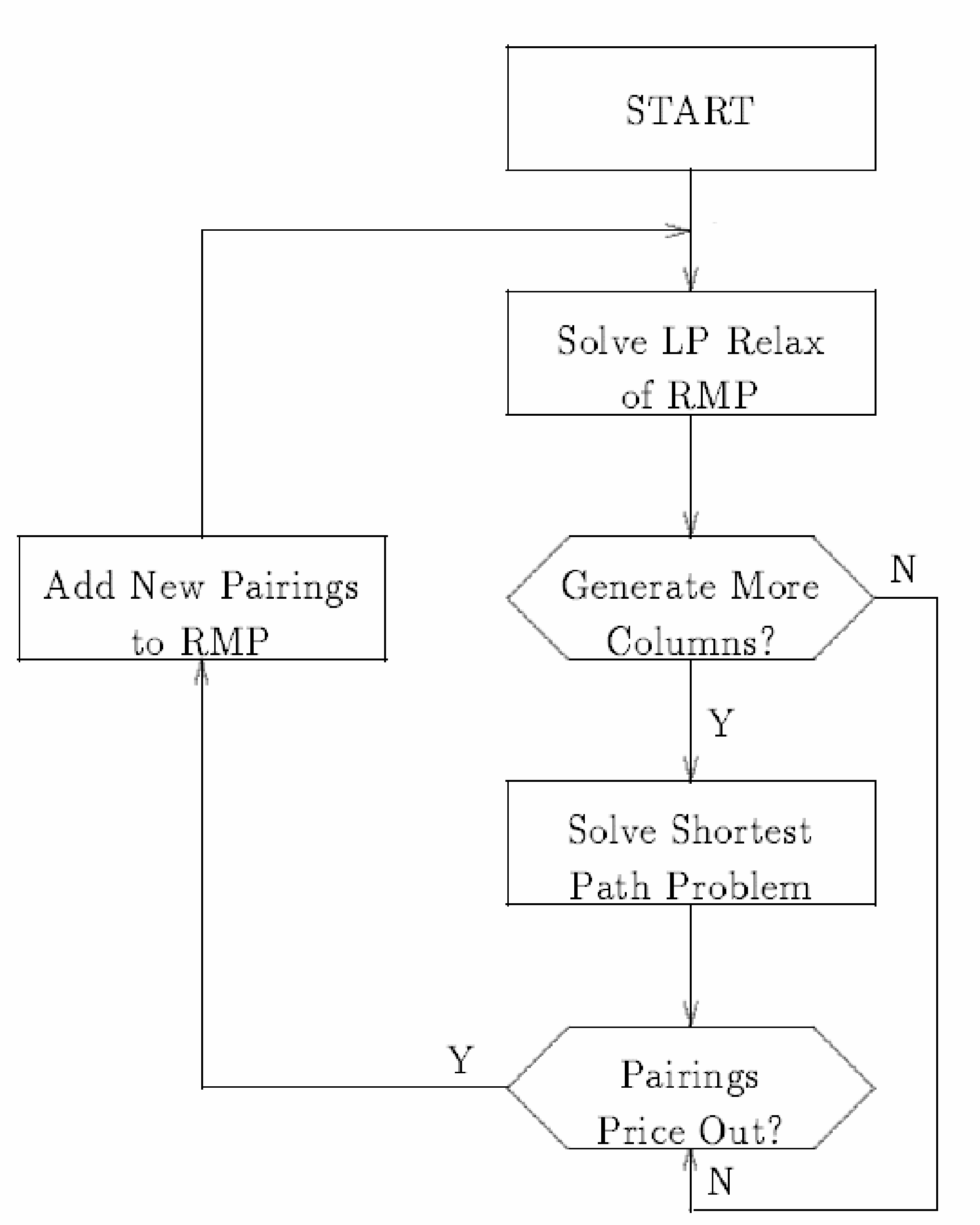


Figure 3.5 Overall algorithm of the dynamic pairing generation and pricing step [3]

3.3. Finding Integer Solution

Since the optimum x_j values are fractional, it is not practical to use them directly. After finding fractional values by solving LP problem, these values should be converted into integer values to be more useful for real world problems. Simply rounding-up the values to integer might be a solution but for large size problem most of the time it fails. Most common algorithms for solving integer linear programming problems are branch and bound, branch and cut and branch and price methods. In this section branch and bound algorithm and slightly changed version of it will be explained. Also another method which is an algorithm for solving large size integer programming problems will be presented.

3.3.1. Branch and Bound Algorithm

Branch and bound is simply a search method. The algorithm searches the complete space of solutions for a given problem for the best solution. First, the algorithm tries to find an optimum solution by considering the original problem with the complete feasible region which is called the root problem. If the optimum solution could not be found, the feasible region is divided into two or more regions and all sub regions should cover the whole feasible region of the original. These sub regions represent the sub problems of the root problem and become the children of the root search node. The algorithm is applied recursively to the sub problems by generating a tree of sub problems. If an optimum solution is found to a sub problem it is feasible solution to the root problem but not necessarily globally optimal. Dividing original problem to sub problems is called branching and generated tree from sub problems is called branching tree. For large size of problems, branching tree can be huge and hence it is impossible to solve all sub problems in a sequence. Therefore bounding step is used not to solve all of the sub problems. In the bounding step the optimum solution of sub problem can be used to prune the tree. If the lower bound for a sub problem (node) exceeds the best known feasible solution, it can be removed from consideration because no globally optimal solution can exist in the subspace of the feasible region represented by the sub problem (node).

For better understanding here is an example for branch and bound method. Let the problem is

Maximize $8x_1 + 11x_2 + 6x_3 + 4x_4$

subject to $5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$

$x_j \in \{0, 1\}$ j = 1, … 4.

The linear relaxation solution is $x_1 = 1$, $x_2 = 1$, $x_3 = 0.5$, $x_4 = 0$ with the objective value of 22. Since x_3 is not integer, the solution is not integer. To force x_3 to be integer, we branch on x_3 by creating two new sub problems. In one, $x_3 = 0$ constraint is added and in the other one $x_3 = 1$ constraint is added. Figure 3.6 shows the branching tree

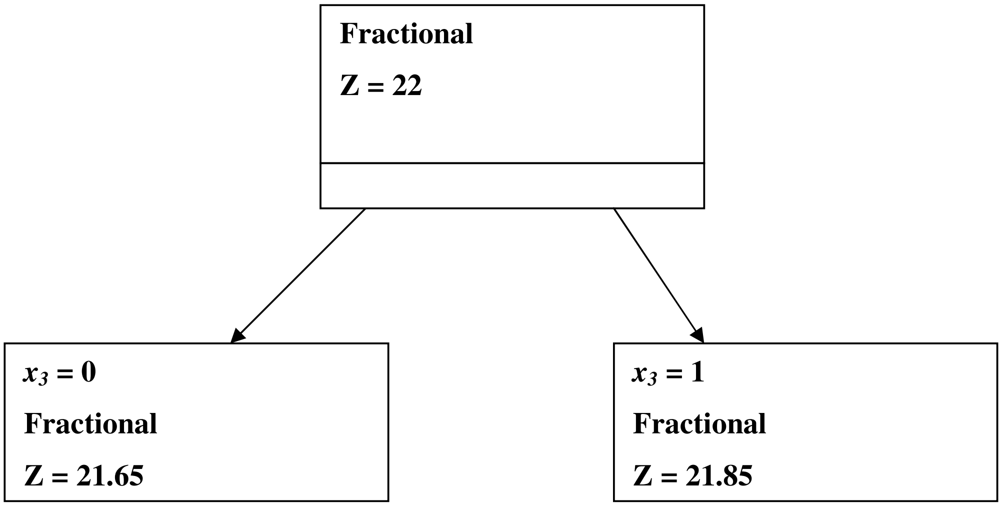


Figure 3.6 First branching tree

For $x_3 = 0$ objective value is 21.65 and values are $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0.667$;
For $x_3 = 1$ objective value is 21.85 and values are $x_1 = 1$, $x_2 = 0.714$, $x_3 = 1$, $x_4 = 0$;

At this point we have two possible paths to find the integer solution. While choosing the sub problem, in general the following rules are taken into account

- Choose one that have not been chosen before
- Choose the sub problem with the best solution value (highest for maximization and the lowest for minimization problems).

So, we choose sub problem with $x_3 = 1$ and branch on x_2. After solving sub problems, the branching tree can be seen in figure 3.7

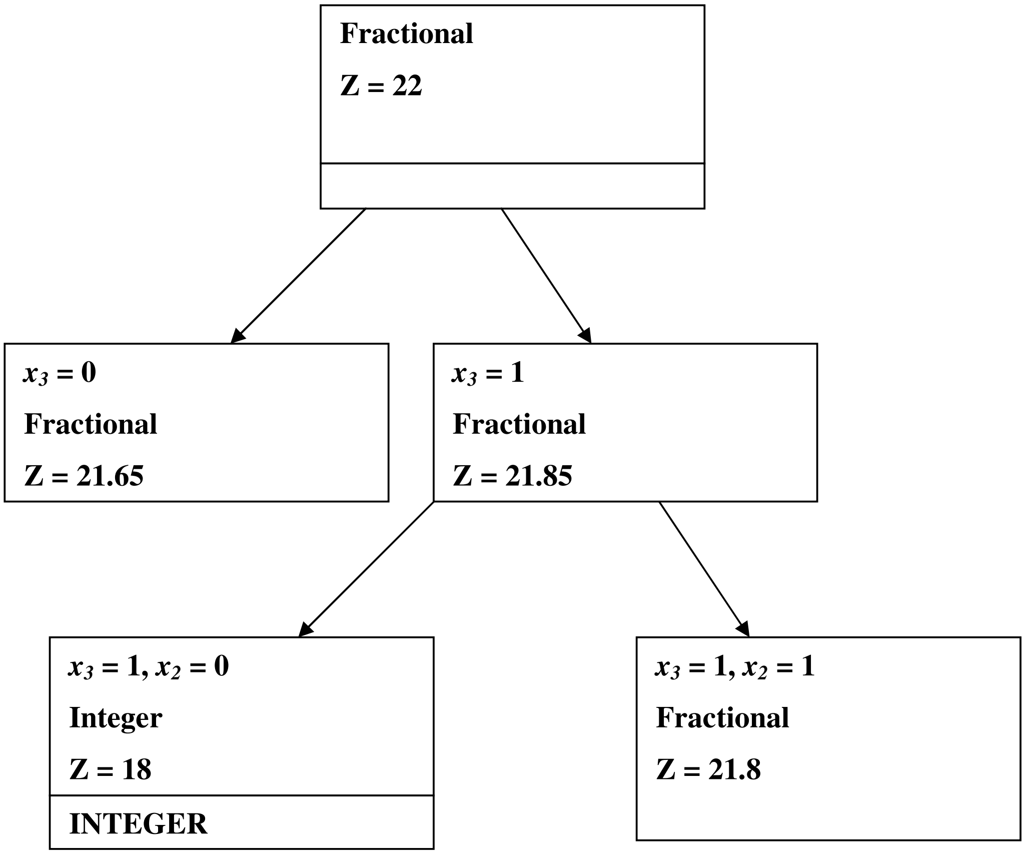


Figure 3.7 Second branching tree

For $x_3 = 1$, $x_2 = 0$ the objective value is 18 and values are $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$;
For $x_3 = 1$, $x_2 = 1$ the objective value is 21.8 and values are $x_1 = 0.6$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$;

We now have a feasible solution with value 18 for $x_3 = 1$, $x_2 = 0$. But there are still sub problems to be solved. So for $x_3 = 1$, $x_2 = 1$, we branch on x_1 and the figure 3.8 shows branching tree

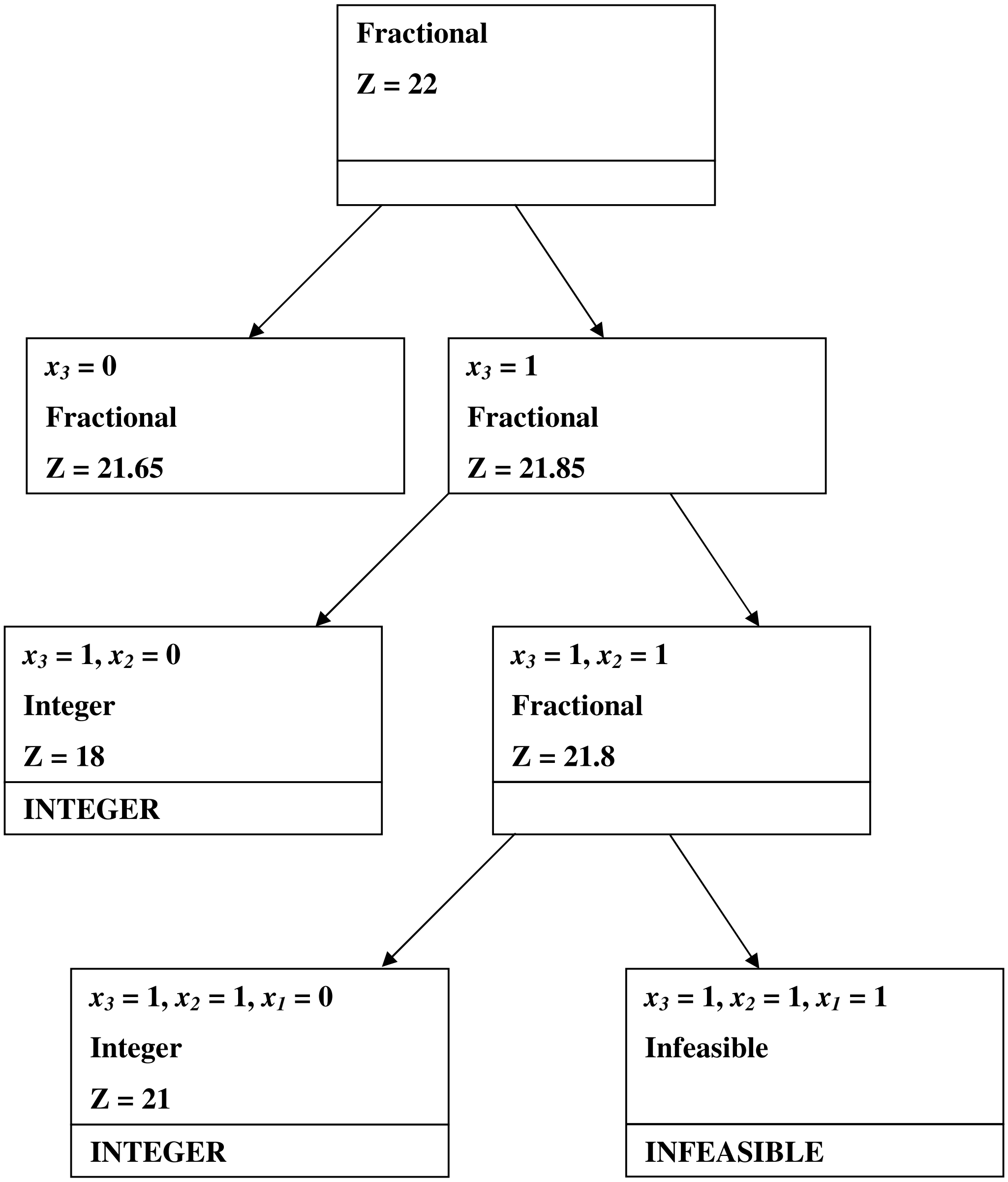


Figure 3.8 Third branching tree

For $x_3 = 1$, $x_2 = 1$, $x_1 = 0$ objective value is 21 and values are $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$; For $x_3 = 1$, $x_2 = 1$, $x_1 = 1$ it is infeasible.

Our best integer solution now has value 21 and it is the lower bound of the problem. But we still have sub problem to be solved. It is the sub problem with the solution value 21.65 that was found in the first branching. But branching on this node will not give an optimum value bigger than 21 so this node is completely discarded based on the bounding argument.

3.3.2. Branch on Follow-ons Method

In the previous section the standard rule of branching is explained. Simply *x* values are fixed to 0 or 1 to divide original problem to sub problems. However, applying standard branching rule to crew pairing problem is difficult to implement and the problem converges very slowly because there many millions of possibilities for all pairings (*x* values). Therefore, another branching rule that was developed by Ryan and Foster [4] is used. This rule is applied after finding fractional solution to the LP relaxation of the set partitioning problem. In this rule, if one pairing contains both of flights r and s, there must exist another pairing that contains only one of the two flights. With this fact, a general branching rule can be constituted that on one branch two flights r and s are covered by the same pairing and by different pairings on the other branch. In general, forcing two specific flights either to appear in only one pairing (first branch) or to never appear in the same pairing (second branch) is difficult. However, forcing two flights to appear consecutively in a pairing or not is an easy task to do. So, the original rule is slightly changed. If two specific flights satisfies the Ryan and Foster rule and they appear consecutively in at least one of the fractional pairings that contains them both, then we can branch by forcing two specific flights to appear consecutively in the pairing for constructing one of the branching node and forcing them to never appear consecutively in any pairings in the solution for constructing the other branching node. A sample branching tree is shown in figure 3.9. The explained strategy is called branch on follow-ons because flights are forced to appear consecutively or not.

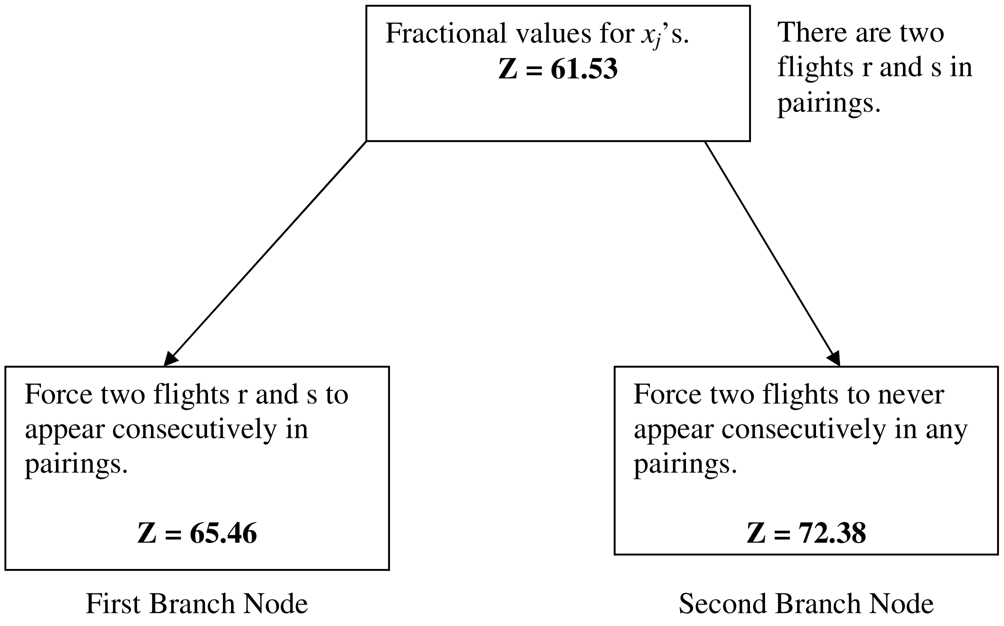

Fractional values for x_j's.
Z = 61.53
There are two flights r and s in pairings.
Force two flights r and s to appear consecutively in pairings.
Z = 65.46
Force two flights to never appear consecutively in any pairings.
Z = 72.38
First Branch Node
Second Branch Node

Figure 3.9 A sample branching tree for branch on follow-on method

In figure 3.9, in first branch node that is forcing two flights r and s to appear consecutively in pairings, any pairing that contains flight r and/or s but not appearing consecutively are removed from the root problem. In the second branch that is forcing two flights to never appear consecutively in any pairings, any pairing that contains flight r and s and they appear consecutively is removed from the root problem. In both nodes, a set of pairings are removed from the root problem that is, x_j values of these pairings are fixed to 0.

In branch on follow-ons method, first follow-on pair is determined that will be branched on and then which part of the branch tree to search next is decided. In general, there are many follow-on pairs in a fractional LP solution. So there needs to be a strategy to decide to which follow-on pair should be selected to be branched on. The strategy is to compare the values of all possible follow-ons that can be calculated by the following formula.

$$f(r,s) = \sum_{j \in P_{r,s}} x_j \tag{3.1}$$

where $\mathbf{P}_{\mathbf{r,s}}$ is a set of pairings that contains flights r and s consecutively. If the value of $f(r,s)$ for flight r and s is the highest value, this follow-on has the highest priority to be selected.

First follow-on pairs that have $f(r,s) = 1$ is fixed and then from the remaining paths, the follow-on pair r,s which has the highest value of $f(r,s)$ but not equal to 1 is fixed. The main idea behind the fixing follow-on is to force it to be appearing consecutively in pairings. Experiments show that searching in the branching tree on the side of the tree where follow-on is fixed gives better solution than on the side where forcing follow-ons not to be appeared consecutively any pairing. In this study also, the branching tree on the side of the tree is searched where follow-on is fixed. The table 3.3 shows the full pseudo-code of the branch on follow-on method.

Table 3.3 Pseudocode of the branch on follow-on method

```
Step 1: Solve LP and find $x_j$ values
Step 2: Select a bunch of pairings (10000 or 15000) based on their reduced cost from all pairings.
Step 3: Find the best follow-on based on $x_j$ values
Step 4: Create the branch
    Step 4: Fix the best follow-on. Remove pairings that does not contain the best follow-on.
    Step 5: If the size of the root problem is too small select more pairings.
    Step 6: First select pairings that contain follow-on flights.
    Step 7: If still it is not enough then select pairings that does not contain both follow-on flights
Step 8: Solve the branch.
Step 9: If it is integer stop. If it is not go to step 4.
```

3.3.3. Carmen Algorithm

In common solution methods for integer programming (IP) problems, first IP problem is relaxed to LP problem as described in the Sprint method. If LP problem gives integer solution then IP solution is found but if it can not give integer solution, other methods like branch and bound algorithm is used. In this algorithm unlike to common methods, it does not use the solution of the LP problem as an intermediate step instead it finds the integer solution in a more direct way. The main idea behind this algorithm is that it uses dual values and reduced cost to find integer x values.

In this method first IP problem is reformulated as an unconstrained nonlinear problem by using lagrangian relaxation (R) as shown in (3.2).

$$\min \boldsymbol{cx} + \boldsymbol{y}(\boldsymbol{b} - \boldsymbol{Ax}) \tag{3.2}$$

where $\bar{c}$ is the reduced cost vector of the variables. Any x values that feasible for IP problem is also feasible for R problem. So if x values are found for R, the solution can also be considered as the feasible solution for IP problem. But solving R is trivial, so instead some vector of y is found and for fixed vector of y, the solution in x is written as

$$x_j = \begin{array}{ll} 1 & \text{if } \bar{c}_{\text{i}} < 0, \\ 0 & \text{if } \bar{c}_{\text{i}} > 0, \\ 0 \text{ or } 1 & \text{if } \bar{c}_{\text{i}} = 0, \end{array} \tag{3.3}$$

As shown in the expression, if $\bar{c}$ is nonzero the solution is integer and unique but if not, then it is difficult to find an integer solution. For most of the problems especially for large size problems, it is almost difficult to find nonzero $\bar{c}$ values. Therefore, a simple coordinate search method is applied to every element of y to find nonzero $\bar{c}$ values.

In the simple coordinate search, iteratively y_i values are found that the solution x to R satisfies the constraint i of Ax=b. For every row i, a $\bar{c}^{\text{i}}$ is written as the elements of $\bar{c}$ and also another array r^i is written as,

$$r^i = \bar{c}^{\,i} + y_i \tag{3.4}$$

The aim of finding r^i is to cancel out the effect of the previous value of y_i from the $\bar{c}$ vector. By using the elements of the r^i the new value of the y_i is found. Let r^- be the lowest element of the r^i and r^+ be the second lowest element of the r^i vector. Then the y_i is written as

$$y_i = \frac{r^- \quad + \quad r^+}{2} \tag{3.5}$$

If r^- and r^+ are nonzero after update $\bar{c}$ vector with the new value of y_i, only one element of the $\bar{c}$ will be negative. In here updating $\bar{c}$ vector means that every element of the r^i is subtracted by exactly the new $\mathrm{y_i}$ value. Also since the average value of the two lowest elements is between the two values, after subtracting the average value the lowest element of the r^i will become negative and the others will be positive. Therefore, the corresponding x_j variable of the negative reduced cost is considered as 1 and added to integer solution. For every y value exactly one x_j value is found and after iterating through every y values which is called row-wise update of reduced cost, an integer solution can be found.

If $r^-$and r^+ values are zero or equal to each other then after calculating $\bar{c}$ vector with the new y_i value, there will be still zero values in the $\bar{c}$ vector. If this is the case, one of the zero values in the $\bar{c}$ vector is chosen randomly to be set as negative and the corresponding x_j variable will be set to 1.

The full pseudo code of the algorithm is shown in table 3.4

Table 3.4 Pseudo code of Carmen algorithm

```
Step 1: For every $i$, find $y_i$ values
        Step 2: Find $r^i$ vector, $r^i = \bar{c}^{\mathrm{i}} + y_i$
        Step 3: Select $r^-$ and $r^+$
        Step 4: Find new value of $y_i$
        Step 5: Calculate the $\bar{c}$ vector with the new value
        Step 6: The corresponding $x_j$ variable of the negative
        reduced cost ($y_i$) is added to integer solution.
Step 7: After finding all $y_i$ values stop.
```

4. HYBRID METHOD

As explained in previous section there are many methods for solving crew pairing problem and every method has its own advantages and disadvantages. One can select one of them to solve crew pairing problem by looking at its advantages and disadvantages. In this section, a hybrid method which is developed by combining some of these methods is explained. The main purpose of developing a hybrid method is finding better solutions and finding it in a more rapid way than the other methods.

In hybrid method for solving LP problem sprint method is used. After solving LP to find integer problems Carmen algorithm is used. Carmen algorithm solves the problem rapidly but it does not give good results. Therefore, for finding integer solution another method (branch on follow-ons) is also combined with sprint method.

First all possible pairings are generated as described in section 3.1. The main reason generating all pairing is that it is also the first step of the sprint method. From all possible pairings, a small amount of them are chosen randomly and solved to find an initial solution. The important thing in here is that the chosen pairings must contain all flights because without containing all flights there will not be a feasible solution. A simple heuristic method is used to cover all flights in selected initial pairings. In pairing generation phase, duties and pairings are generated in an order. For example in duties list, most probably first duties contain first flight and last duties contain last flight. In pairing list the same logic exists such that first pairings contain first duty and also contain first flight. So pairing list is divided into small subsets as much as the flight size. From all subsets randomly pairings are chosen. Hence this heuristics provide us to find initial pairings which cover all flights. After finding initial solution, its dual values are used in the first iteration of sprint method. After finding good linear programming solution, Carmen algorithm and branch on follow-ons methods are used individually to find integer solution.

Sprint method is used to find linear programming solution, because even for large size of pairings, optimum solution is found after a few iterations. The main logic behind the sprint method is to solve large sized problems with solving small subsets iteratively but taking reduced cost into consideration in order to include most probable optimum pairings to small subsets. As it is the fastest method for finding optimum solution for LP problems, it is used as primary method in this study. One of the disadvantages of this method is that all pairings must be generated and this means more memory is needed before starting iteration. Also this disadvantage is the main difference between the dynamic pairing generation methods (A Heuristic Approach for the Airline Crew Pairing Problem) and sprint method. As described in section 3.2.2, pairings are generated based on their reduced costs. But main side effect of this method is that even though pairings are generated dynamically, generating pairings are not straight forward as in sprint method. Because based on their flight and duty networks for generating new pairings different algorithms should be used and these algorithms are mostly time consuming. That is why sprint method is used to solve LP problem as it is faster than the other. Table 4.1 shows the main differences between two methods

Table 4.1 The differences between Sprint and Dynamic Pairing Generation method

Sprint Method	**Dynamic Pairing Generation**
Generate all pairings	Dynamic pairing generation
Needs more memory	Needs less memory
Pairings are generated by simply constructing a tree and using depth-first algorithm.	Pairings are generated by constructing complex network and using shortest path algorithm.
Uses reduced cost values to generate small subsets	Uses reduced cost values to generate pairings

Also experiments show the difference between sprint method and dynamic column generation method. Table 4.2 shows the run time of the methods for various numbers of flights.

Table 4.2 The run time of the methods for various numbers of flights [3] [5]

Sprint Method		**Dynamic Pairing Generation**	
Number of Flights	CPU Time	Number of Flights	CPU Time
837	0.95 hours	123	0.13 hours
1152	6.3 hours	449	3.57 hours

For finding integer solution after solving LP problem, both branch on follow-on method and Carmen algorithm is implemented. Carmen algorithm is faster than branch on follow-on method on the other hand branch on follow on method gives better and more feasible solution than the Carmen method. The main disadvantages of Carmen method is that it tries to find integer variable randomly. As described in section 3.3.3 the main logic behind Carmen algorithm is deciding on a variable to be 0 or 1 based on its reduced cost. If the reduced cost of the variable is negative then it is set to 1 if positive it is set to 0. However if the reduced cost is zero it is not easy to decide that variable should be 0 or 1. To resolve this problem a simple algorithm is implemented but still there will be variables with zero reduced cost. Hence it is decided to be 0 or 1 randomly for that variable and this randomness might cause getting bad results from the Carmen algorithm. That is why after Carmen algorithm implementation also branch on follow-on method is implemented. In branch on follow-ons method as described in section 3.3.2 a special branch rule is used to find integer solutions. Before starting branch on follow-on method, an initial phase should be constructed. The initial phase is created by selecting 10000 or 15000 pairings based on their reduced cost. In every branch one ore more follow-ons are fixed and generated pairing list is solved until no follow-on is left to be fixed. In figure 4.1 the overall algorithm is shown

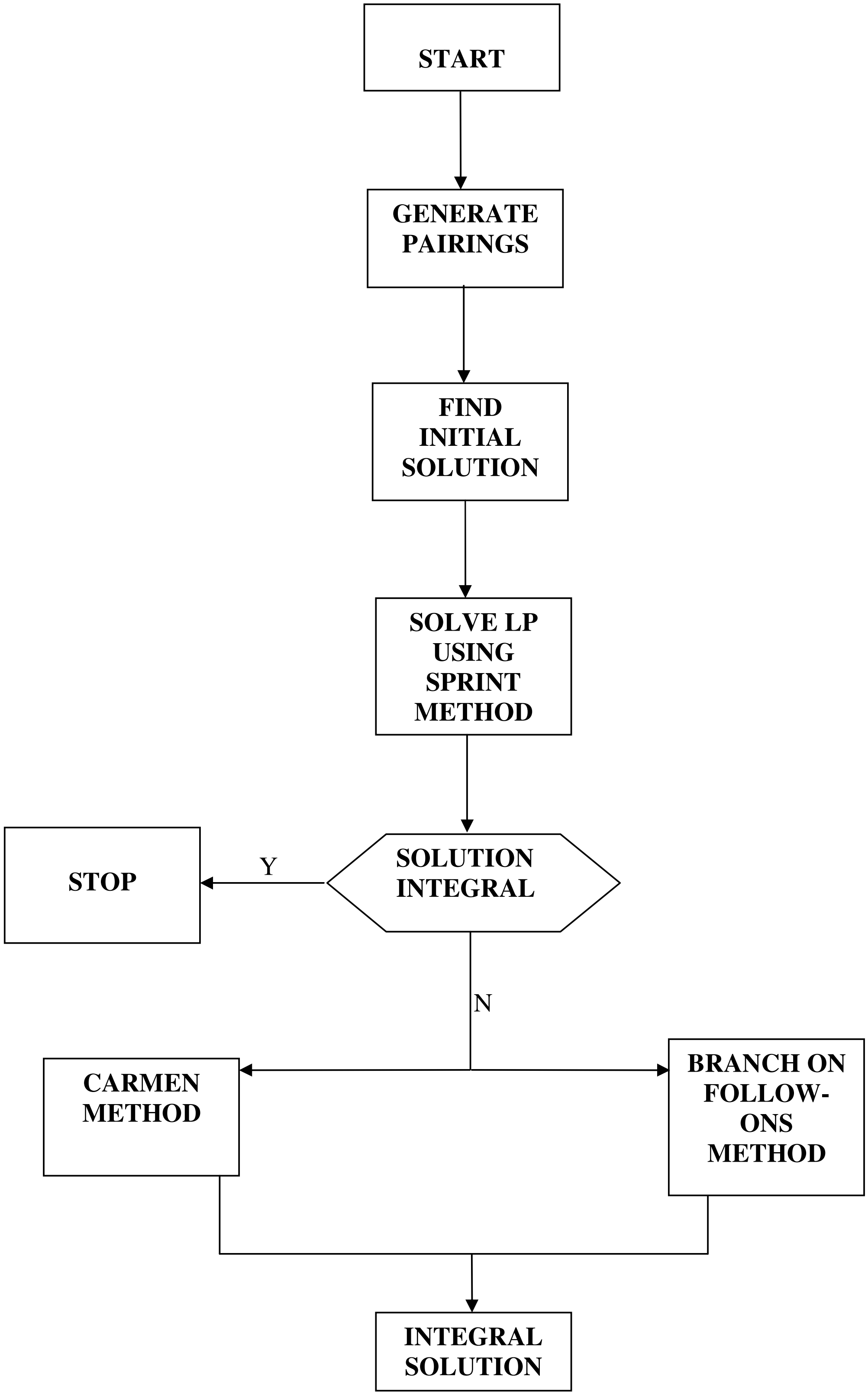


Figure 4.1 The overall algorithm of the hybrid method

4.1. Usage Areas of Hybrid Method

In the previous section, the usage of hybrid method on crew pairing problem is shown. In crew pairing problem typically, flight schedule is given as an input and found pairings are obtained as an output. For different problems like train scheduling and bus scheduling, similar input and output components are used. For example in train schedule problem, train schedule can be considered as flight schedule, train driver can be considered as airline crew and obtained train driver pairings can also be considered as crew pairings. Therefore, the methodology in crew pairing problem (the hybrid method) not only applicable for airline crew scheduling problem, it can also be applied to different crew or driver scheduling problems. More interesting usage area of hybrid method can be doctor and nurse scheduling in hospitals. For example, emergency services should be in operation for 24 hours and at any moment the hospital should be occupied at least one doctor or nurse. In doctor scheduling problem, whole day can be divided into 24 hours and each doctor can be scheduled for two or more hours. But there should not be gaps in the schedule between shift (can be considered as flights) changes unlike in crew pairing problem that there are sitting times between flights or duties. Hence, the hybrid method can be applied to almost every scheduling problem by simply changing parameters or rules.

5. EXPERIMENTS

Two versions of hybrid method is implemented one is using Carmen Algorithm and the other one is using branch on Follow-ons method to find integer solution. The hybrid algorithm is implemented by using Java and as a LP solver Matlab is used. Using special Matlab builder, required functions are converted into Java code and therefore matlab functions are called directly from Java code. All test runs are made on PC Pentium IV with 1GB ram. For sprint method, size of the sub problem is 5000 and Carmen algorithm and branch on follow-ons method is started with 10000 pairings. In experiments, the following values are used;

Table 5.1 The used values in experiments

Duty Rules	
minsit	0.5 hours
maxsit	4 hours
maxelapse	12 hours
maxfly	8 hours
mg1	3 hours
f1	4/7 hours
Pairing Rules	
maxduties	2 duties
mg2	4.75 hours
f2	2/7 hours

In Table 5.2, the comparison between Carmen algorithm and branch on follow-on method is shown and before finding integer values sprint method is used. The column “flights” represents the number of flights; the “objective” gives objective value of the solution; the “CPU Time” column represents the run time of the trial in minutes. As the experiments show that Carmen algorithm is faster than the other but it gives bad results considering the objective value.

Table 5.2 The comparison between Carmen algorithm and branch on follow-on method

Branch on Follow-on			**Carmen Algorithm**		
Flights	objective (h)	CPU Time (min)	Flights	objective (h)	CPU Time (min)
38	60.15	55	38	95.27	42
58	95	71	58	171	61
96	152.07	102	96	342.12	77
144	251.82	167	144	453.48	125
222	380.43	245	222	610.60	186

Experiments are also proved that Carmen algorithm is the fastest but it does not give good results when considering branch on follow-on method.

Table 5.3 shows the difference between hybrid method and original Carmen algorithm. The column “Flights” represents the number of flights; the “obj.” gives objective value of the solution; the “CPU” column represents the run time of the trial in minutes. Experiments show that hybrid method with Carmen algorithm gives similar results with the original one. It is observed that using sprint method as a preliminary step does not have significant effect on final result. Also the run time of the trials in hybrid method is longer than the original Carmen algorithm. The reason for longer runs is that in hybrid method before starting Carmen algorithm, LP problem is solved with sprint method. Big portion of the time is used for solving LP problem. But in original Carmen algorithm, integer solution is found in a more direct way that is; it does not use the solution of LP problem as an intermediate step. When comparing hybrid method with branch on follow-on method with the original Carmen algorithm, hybrid method gives better results. The similar results are also obtained in table 5.2 when comparing two versions of hybrid method.

Table 5.3 The comparison between hybrid method and original Carmen algorithm

Hybrid Method with Branch on Follow-on Method			Hybrid Method with Carmen Algorithm			Original Carmen Algorithm		
Flights	obj. (h)	CPU	Flights	obj. (h)	CPU	Flights	obj. (h)	CPU
144	251.82	167	144	453.48	125	144	470.37	15
222	380.43	245	222	610.60	186	222	617.95	27

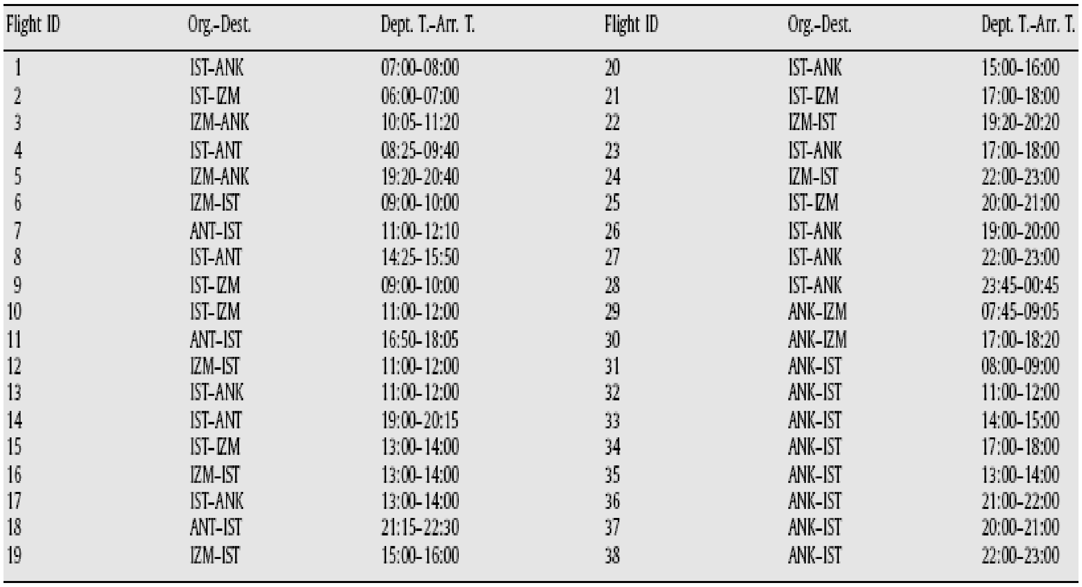
6. CONCLUSION

In this study a hybrid method is modeled for airline crew pairing problem by combining previously developed methods. The aim of developing a hybrid method is to find better solutions than previous methods. The main advantage of our hybrid method is that it finds the solution very fast. But it also most of the time gives bad result because of using Carmen algorithm whose side effect is choosing variables randomly while finding integer values. That is why another version of hybrid method is implemented by using branch on follow-on method. With branch on follow-on method, it is not as fast as Carmen algorithm but it gives better result which is also shown with experiments. Also experiments show that using sprint method before Carmen algorithm does not have significant effect on the final solution.

In sprint method using a good initial solution plays an important role on finding good solution for LP problem. In this study for finding initial solution, pairings are selected randomly. To improve the quality of the sprint method other algorithm can be used for finding an initial solution. For example any solution which is obtained from an earlier similar run can be used as initial solution.

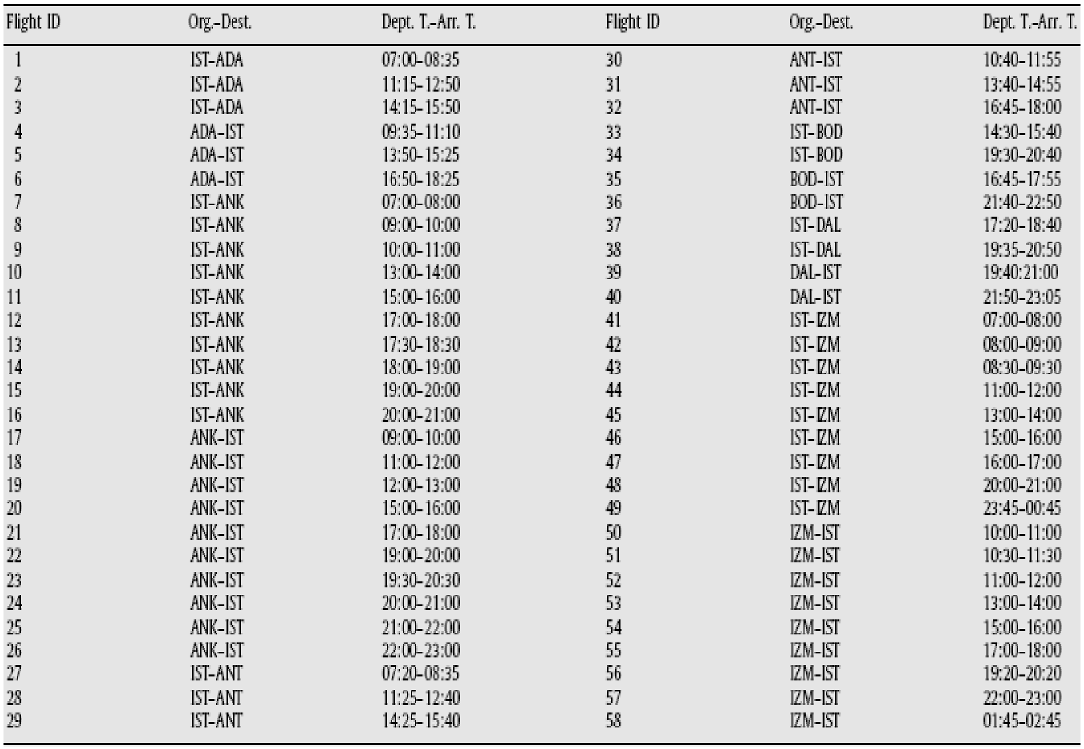
APPENDIX A: DATA USED IN EXPERIMENTS

Table A.1 The sample timetable for 38 flights [6]



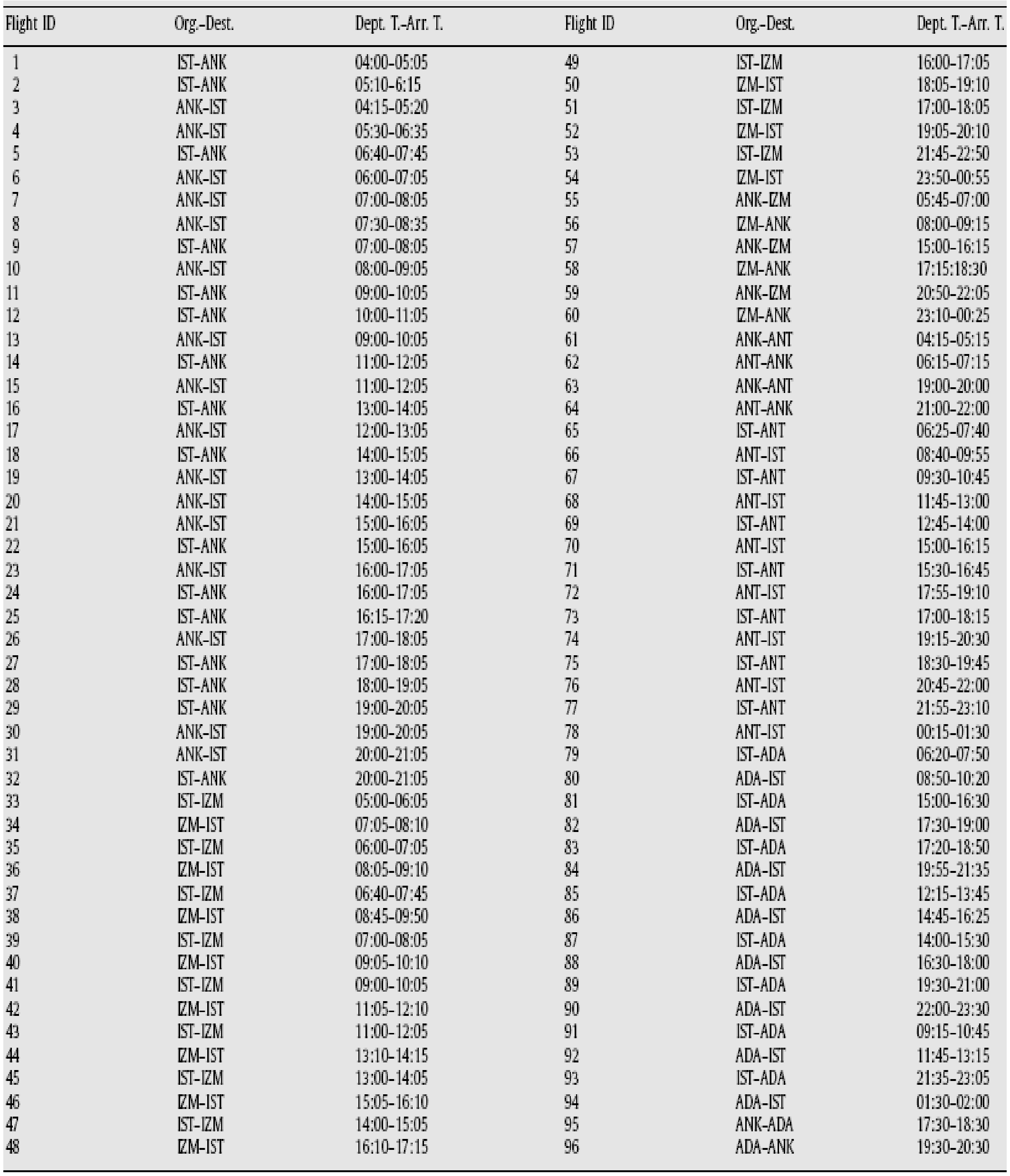
Flight ID	Org.-Dest.	Dept. T.-Arr. T.	Flight ID	Org.-Dest.	Dept. T.-Arr. T.
1	IST-ANK	07:00-08:00	20	IST-ANK	15:00-16:00
2	IST-IZM	06:00-07:00	21	IST-IZM	17:00-18:00
3	IZM-ANK	10:05-11:20	22	IZM-IST	19:20-20:20
4	IST-ANT	08:25-09:40	23	IST-ANK	17:00-18:00
5	IZM-ANK	19:20-20:40	24	IZM-IST	22:00-23:00
6	IZM-IST	09:00-10:00	25	IST-IZM	20:00-21:00
7	ANT-IST	11:00-12:10	26	IST-ANK	19:00-20:00
8	IST-ANT	14:25-15:50	27	IST-ANK	22:00-23:00
9	IST-IZM	09:00-10:00	28	IST-ANK	23:45-00:45
10	IST-IZM	11:00-12:00	29	ANK-IZM	07:45-09:05
11	ANT-IST	16:50-18:05	30	ANK-IZM	17:00-18:20
12	IZM-IST	11:00-12:00	31	ANK-IST	08:00-09:00
13	IST-ANK	11:00-12:00	32	ANK-IST	11:00-12:00
14	IST-ANT	19:00-20:15	33	ANK-IST	14:00-15:00
15	IST-IZM	13:00-14:00	34	ANK-IST	17:00-18:00
16	IZM-IST	13:00-14:00	35	ANK-IST	13:00-14:00
17	IST-ANK	13:00-14:00	36	ANK-IST	21:00-22:00
18	ANT-IST	21:15-22:30	37	ANK-IST	20:00-21:00
19	IZM-IST	15:00-16:00	38	ANK-IST	22:00-23:00

Table A.2 The sample timetable for 58 flights [6]



Flight ID	Org.-Dest.	Dept. T.-Arr. T.	Flight ID	Org.-Dest.	Dept. T.-Arr. T.
1	IST-ADA	07:00-08:35	30	ANT-IST	10:40-11:55
2	IST-ADA	11:15-12:50	31	ANT-IST	13:40-14:55
3	IST-ADA	14:15-15:50	32	ANT-IST	16:45-18:00
4	ADA-IST	09:35-11:10	33	IST-BOD	14:30-15:40
5	ADA-IST	13:50-15:25	34	IST-BOD	19:30-20:40
6	ADA-IST	16:50-18:25	35	BOD-IST	16:45-17:55
7	IST-ANK	07:00-08:00	36	BOD-IST	21:40-22:50
8	IST-ANK	09:00-10:00	37	IST-DAL	17:20-18:40
9	IST-ANK	10:00-11:00	38	IST-DAL	19:35-20:50
10	IST-ANK	13:00-14:00	39	DAL-IST	19:40:21:00
11	IST-ANK	15:00-16:00	40	DAL-IST	21:50-23:05
12	IST-ANK	17:00-18:00	41	IST-IZM	07:00-08:00
13	IST-ANK	17:30-18:30	42	IST-IZM	08:00-09:00
14	IST-ANK	18:00-19:00	43	IST-IZM	08:30-09:30
15	IST-ANK	19:00-20:00	44	IST-IZM	11:00-12:00
16	IST-ANK	20:00-21:00	45	IST-IZM	13:00-14:00
17	ANK-IST	09:00-10:00	46	IST-IZM	15:00-16:00
18	ANK-IST	11:00-12:00	47	IST-IZM	16:00-17:00
19	ANK-IST	12:00-13:00	48	IST-IZM	20:00-21:00
20	ANK-IST	15:00-16:00	49	IST-IZM	23:45-00:45
21	ANK-IST	17:00-18:00	50	IZM-IST	10:00-11:00
22	ANK-IST	19:00-20:00	51	IZM-IST	10:30-11:30
23	ANK-IST	19:30-20:30	52	IZM-IST	11:00-12:00
24	ANK-IST	20:00-21:00	53	IZM-IST	13:00-14:00
25	ANK-IST	21:00-22:00	54	IZM-IST	15:00-16:00
26	ANK-IST	22:00-23:00	55	IZM-IST	17:00-18:00
27	IST-ANT	07:20-08:35	56	IZM-IST	19:20-20:20
28	IST-ANT	11:25-12:40	57	IZM-IST	22:00-23:00
29	IST-ANT	14:25-15:40	58	IZM-IST	01:45-02:45

Table A.3 The sample timetable for 96 flights [6]



Flight ID	Org.-Dest.	Dept. T.-Arr. T.	Flight ID	Org.-Dest.	Dept. T.-Arr. T.
1	IST-ANK	04:00-05:05	49	IST-IZM	16:00-17:05
2	IST-ANK	05:10-6:15	50	IZM-IST	18:05-19:10
3	ANK-IST	04:15-05:20	51	IST-IZM	17:00-18:05
4	ANK-IST	05:30-06:35	52	IZM-IST	19:05-20:10
5	IST-ANK	06:40-07:45	53	IST-IZM	21:45-22:50
6	ANK-IST	06:00-07:05	54	IZM-IST	23:50-00:55
7	ANK-IST	07:00-08:05	55	ANK-IZM	05:45-07:00
8	ANK-IST	07:30-08:35	56	IZM-ANK	08:00-09:15
9	IST-ANK	07:00-08:05	57	ANK-IZM	15:00-16:15
10	ANK-IST	08:00-09:05	58	IZM-ANK	17:15:18:30
11	IST-ANK	09:00-10:05	59	ANK-IZM	20:50-22:05
12	IST-ANK	10:00-11:05	60	IZM-ANK	23:10-00:25
13	ANK-IST	09:00-10:05	61	ANK-ANT	04:15-05:15
14	IST-ANK	11:00-12:05	62	ANT-ANK	06:15-07:15
15	ANK-IST	11:00-12:05	63	ANK-ANT	19:00-20:00
16	IST-ANK	13:00-14:05	64	ANT-ANK	21:00-22:00
17	ANK-IST	12:00-13:05	65	IST-ANT	06:25-07:40
18	IST-ANK	14:00-15:05	66	ANT-IST	08:40-09:55
19	ANK-IST	13:00-14:05	67	IST-ANT	09:30-10:45
20	ANK-IST	14:00-15:05	68	ANT-IST	11:45-13:00
21	ANK-IST	15:00-16:05	69	IST-ANT	12:45-14:00
22	IST-ANK	15:00-16:05	70	ANT-IST	15:00-16:15
23	ANK-IST	16:00-17:05	71	IST-ANT	15:30-16:45
24	IST-ANK	16:00-17:05	72	ANT-IST	17:55-19:10
25	IST-ANK	16:15-17:20	73	IST-ANT	17:00-18:15
26	ANK-IST	17:00-18:05	74	ANT-IST	19:15-20:30
27	IST-ANK	17:00-18:05	75	IST-ANT	18:30-19:45
28	IST-ANK	18:00-19:05	76	ANT-IST	20:45-22:00
29	IST-ANK	19:00-20:05	77	IST-ANT	21:55-23:10
30	ANK-IST	19:00-20:05	78	ANT-IST	00:15-01:30
31	ANK-IST	20:00-21:05	79	IST-ADA	06:20-07:50
32	IST-ANK	20:00-21:05	80	ADA-IST	08:50-10:20
33	IST-IZM	05:00-06:05	81	IST-ADA	15:00-16:30
34	IZM-IST	07:05-08:10	82	ADA-IST	17:30-19:00
35	IST-IZM	06:00-07:05	83	IST-ADA	17:20-18:50
36	IZM-IST	08:05-09:10	84	ADA-IST	19:55-21:35
37	IST-IZM	06:40-07:45	85	IST-ADA	12:15-13:45
38	IZM-IST	08:45-09:50	86	ADA-IST	14:45-16:25
39	IST-IZM	07:00-08:05	87	IST-ADA	14:00-15:30
40	IZM-IST	09:05-10:10	88	ADA-IST	16:30-18:00
41	IST-IZM	09:00-10:05	89	IST-ADA	19:30-21:00
42	IZM-IST	11:05-12:10	90	ADA-IST	22:00-23:30
43	IST-IZM	11:00-12:05	91	IST-ADA	09:15-10:45
44	IZM-IST	13:10-14:15	92	ADA-IST	11:45-13:15
45	IST-IZM	13:00-14:05	93	IST-ADA	21:35-23:05
46	IZM-IST	15:05-16:10	94	ADA-IST	01:30-02:00
47	IST-IZM	14:00-15:05	95	ANK-ADA	17:30-18:30
48	IZM-IST	16:10-17:15	96	ADA-ANK	19:30-20:30

REFERENCES

1. Gustafsson, T., “A heuristic approach to column generation for airline crew scheduling”, *Chalmers University of Technology*, Gothenburg, Sweden, 1999.

2. Anbil, R., R. Tanga, and E.L. Johnson, “A global approach to crew-pairing optimization”, *IBM Systems Journal 31*, 71-78., 1992.

3. Vance, P. H., A. Atamturk, C. Barnhart, E. Gelman, E. L. Johnson, A. Krishna, D. Madidhara, G. L., Nemhauser, and R. Rebello, “A Heuristic Branch-and-Price Approach for the Airline Crew Pairing Problem”, *Technical Report TLI/LEC-97-06*, Georgia Institute of Technology, Atlanta, GA, 1997.

4. Ryan, D. M., and B. A. Foster, “An integer programming approach to scheduling”, A. Wren (ed.) *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, North-Holland, Amsterdam, 269-280, 1981.

5. Chu, H. D., E. Gelman, and E. L. Johnson, “Solving large scale crew scheduling problems”, *European Journal of Operational Research* 97 260-268, 1997.

6. Tekiner H, et al, “Robust crew pairing for managing extra flights”, *Computers and Operations Research* 2008.

7. Wedelin, D., “An algorithm for large scale 0–1 integer programming with application to airline crew scheduling”, *Annals of Operations Research*, 1995.

8. Barnhart, C., E. L. Johnson, G. L. Nemhauser, Martin W. P. Savelsbergh, P. H. Vance, “Branch-and-Price Column Generation for Solving Huge Integer Programs”, *Operations Research*, 46, 316–329, 1998.

9. Ranga, A., J. J. Forrest, and W. R. Pulleyblank, “Column Generation and the Airline Crew Pairing Problem”, *Documenta Mathematica III, 677–686,* 1998.

10. Thorsteinsson, E. S., G. Ottosson, “Linear Relaxations and Reduced-Cost based Propagation of Continuous Varible Subscripts”, *Annals of Operations Research 115, 15–29*, 2002.

11. Zeghal, F. M., and M. Minoux, “Modeling and solving a Crew Assignment Problem in air transportation”, *European Journal of Operational Research 175 187–20,* 2006.

12. Yu, G., “OR In Airline Industury”, *Kluwer Academic Publishers*, 1999.

13. Vance, P. H., C. Barnhart, E.L. Johnson, and G.L. Nemhauser, “Airline Crew Schduling: A new Formulation and Decomposition Algorithm”, *Georgia Institute of Technology*, Atlanta, Gergia, 1994.

14. Klabjan, D., E. L. Johnson, and G. L. Nemhauser, “Solving Large Airline Crew Scheduling Problems: Random Pairing Generation and Strong Branching”, *Computational Optimization and Applications*, 20, 73–91, 2001.

15. Takkula, T. K., “The Dual of Integer Linear Programs”, *Chalmers University of Technology and Gothenburg University,* 2001.

16. Schaefer, A. J., E. L. Johnson, A. J. Kleywegt, and G. L. Nemhauser, “Airline Crew Scheduling under Uncertainty”, *Georgia Institute of Technology*, 2001.

17. Vershelde, J., “Linear Programming in MATLAB”, *Introduction to Symbolic Computation*, Lecture 9, page 1-2, 2003.

18. Klimke, A., “How to Access Matlab from Java”, *Universitat Stuttgart*, 2003.

19. Borndörger, R., U. Schelten, T. Schlechte, and S. Weider, “A Column Generation Approach to Airline Crew Scheduling”, *Konrad-Zuse-Zentrum für Informationstechnik Berlin*, 2006.

20. Crawford, B., C. Castro, E. Monfroy, “A Constructive Hybrid Algorithm for Crew Pairing Optimization” *Pontificia Universidad, Lecture Notes in Computer Science*, 2006.

21. Klabjan, D., E. L. Johnson, and G. L. Nemhauser, “Airline Crew Scheduling with Regularity”, *Transportation Science*, 2001.

22. Medard, C. P., and N. Sawhney, “Airline crew scheduling from planning to operations”, *European Journal of Operational Research 183,* 1013–1027, 2007.

23. Ahmadbeygi, S., A. Cohn, M. Weir, “An integer programming approach to generating airline crew pairings”, *Computers and Operations Research*, 2008.