Solving The Long Haul Crew Pairing Problem

by

Rajesh Gopalakrishna Shenoi

B.Tech., Indian Institute of Technology, Madras (1991)

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements for the degrees of

MASTER OF SCIENCE
in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1994

Author .... [illegible] ....
Department of Civil and Environmental Engineering
December 31,1993

Certified by .... [illegible] ....
Cynthia Barnhart
Assistant Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .... [illegible] ....
Joseph Sussman
Chairman, Departmental Committee on Graduate Students

Solving The Long Haul Crew Pairing Problem

by

Rajesh Gopalakrishna Shenoi

Submitted to the Department of Civil and Environmental Engineering
on December 31,1993, in partial fulfillment of the
requirements for the degrees of
MASTER OF SCIENCE
in Transportation

Abstract

A crew pairing is a sequence of flights, beginning and ending at the same crew base, that satisfies numerous Federal Aviation Administration and contractual requirements. The crew pairing problem, an integral part of the planning process in the airline industry, involves the construction of a set of valid crew pairings that assign a crew to every flight and minimize total costs. This problem is formulated as a set covering problem, and is solved using a branch and bound framework in which bounds are provided by solving a linear program at each node of the branch and bound tree. The LP relaxation is solved using a column generation algorithm. We modify the conventional column generation algorithm by embedding within it a dual ascent heuristic that speeds up the convergence of the column generation algorithm and provides lower bounds on the optimal solution value. The dual ascent bounds are used together with a bounding scheme based on Farley's method (1990). The solution method is tested in a case study using data provided by a long-haul airline. The results show that early termination of column generation is possible using the bounds generated by the dual ascent heuristic and Farley's bounding scheme, giving solutions that are close to the optimal LP solution value and reducing overall solution time.

Thesis Supervisor: Cynthia Barnhart
Title: Assistant Professor of Civil and Environmental Engineering

Acknowledgments

I take this opportunity to thank Prof. Cynthia Barnhart for being a wonderful advisor, a good friend, and a constant source of inspiration in my research.

I wish to thank Prof. Ellis Johnson, Pam Vance, Ram Pandit, Steve Querido, and Lloyd Clarke of Georgia Institute of Technology, and Daeki Kim, Ching-Ju Juan, Rajiv Lochan, Yuting Kuo and Yusin Lee whose ideas and comments were very useful in providing me with research directions.

I wish to thank Andras Farkas, Adriana Bernardino, Amalia Polydoropoulou, Qi Yang, Rabi Mishalani and all other members of 5-008 whose coffee breaks were things to remember.

I would like to thank Salal Humair, Amjad Shahbazker, Antulio Richetta, and all the 1.00 teaching assistangs who made teaching a fun thing to do.

I wish to thank Mark Hickman, Theodore Botimer, Rick Halvorsen, Scott Freedman, Oh Kyoung Kwon, Bill Cowart, Dan Turk and Hans Klein for being very cooperative officemates.

I wish to thank Dinesh Gopinath, Ashok Kalidas, Anil Mukundan, Nagaraja "Hash" Harshdeep, Nageshwara Rao, Ravi Sundaram, Prodyut Dutt and all my other friends whose friendship and support made my life in this country very pleasant and enjoyable.

I wish to thank my parents, my brother, my sister, and Ana Laplaza for their encouragement and unconditional support. Some drafts of this thesis were printed on recycled paper.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Edwin McDowell(1993) reports that "Delta, which lost about $540 million last year on $1.78 billion in revenue, has trimmed its work force 7.7 percent .,. since June 1992 ... The carrier also recently offered early retirement to 3,000 of its remaining 73,400 employees." He further states that Delta will layoff as many as 600 pilots in the next few months, with these being the first layoffs since 1957. Additionally, American Airlines is considering laying off hundreds of employees by the end of 1993 in order to trim the 1994 budgets by about 10 percent. AMR corporation, parent of American Airlines, has lost more than $1.4 billion in the last three years, and the airline industry has lost $10 billion in the same period (McDowell,1993). In another recent article, Prof. Stephen Solomon(1993) describes the problems that the AMR corporation has been undergoing for the last few years. Robert Crandall, the CEO of American Airlines, spent $20 billion in the 1980's to increase the market share of the airline in the domestic market from 14.4% to 20.4%. But Crandall's growth plan failed and the massive spending has not brought in the expected profits. In the last decade, fare wars have driven down the prices of tickets by over 20% (adjusted for inflation). In fact, in a recent price war (McDowell,1993b), the nation's airlines temporarily cut fares by upto 45% on domestic flights. Moreover, new low-cost regional carriers are entering the market each year and many of the competitors of American have made serious cost cuts while still in bankruptcy.

The airline industry is facing a serious financial crisis and low revenues have forced all airlines to review their strategies and cut costs as much as possible. To understand how this can be done, consider the breakdown of the airline industry operating costs. Table 1.1 gives a list of major airline operating costs for 5 different years from 1970 to 1992 (Simpson and Belobaba,1993). For the years 1970 through 1985, crew costs (the sum of pilot and cabin crew costs) is the second highest flight operating cost in the airline industry, next only to fuel costs. In 1992, the crew costs (14.4%) are higher than the fuel costs(12.9%). (Some data was not available for the years of 1970 and 1975.)

Consider, for example, that the annual crew costs at American Airlines are about \$1.3 billion (Anbil, et al.,1991). A lower bound on total crew costs can be expressed as a function of the flying time of all flights flown by the airline. (It is not sure whether this bound can ever be achieved.) Currently, American Airlines has achieved solutions which are about 3% higher than this bound. Reducing this gap to 2% implies savings of about \$13 million a year for American Airlines. Thus, even minor improvements can cause huge savings in operating costs. This provides the main motivation for this thesis.

1.2 Overall View of the Airline Problem

The overall airline problem of operating and managing passenger transport is too complex and large to be modeled in a single practicable formulation. Elce(1970) states that the planning process is complicated by the large number of variables. He suggests that it is quite difficult to suggest a single technique that can be applied to simultaneously achieve the multiplicity of goals. Airlines usually manage this problem by breaking it up into a multi-stage process. Many of the stages seen below are by themselves highly intractable and would thus render a single-stage solution of the overall problem impossible given the current state of technology. Thus, a tradeoff

Type of cost ↓	*% Operating costs (by year)*				
	1970	1975	1980	1985	1992
A Flight Operating Costs (FOC)					
Fuel Costs	12.5	17.9	29.8	21.8	12.9
Pilots Costs	13.0	11.9	11.1	10.7	9.7
Direct Engine costs	7.7	6.6	4.8	4.9	6.1
Indirect (Burden)	6.3	5.1	3.9	3.3	4.0
Depreciation	8.8	6.3	4.0	4.7	5.1
Rentals	2.8	2.3	1.0	1.2	7.8
Insurance (hull)	0.5	0.3	0.1	0.3	0.4
Total FOC	**52.4**	**51.0**	**55.6**	**47.4**	**50.4**
B Ground Operating Costs (GOC)					
Traffic servicing	8.7	9.0	8.0	9.4	10.1
Aircraft servicing	8.0	7.4	5.9	5.9	6.1
Service administration	1.1	1.0	8.0	1.1	1.0
Commisions	2.5	3.4	4.8	7.8	7.6
Reservations & sales	6.6	5.6	5.5	7.7	6.7
Total GOC	**26.9**	**26.4**	**25.1**	**31.9**	**30.7**
C System Operating Costs (SOC)					
Cabin Crew Costs	-	-	5.0	3.9	4.7
Meals	-	-	4.4	3.4	3.5
Advertising	2.5	1.9	1.8	2.3	1.2
General & Administrative	4.5	4.2	3.2	4.2	3.7
Ground Equipment	3.2	3.0	3.5	1.8	2.8
Other	-	-	1.2	1.8	1.2
Total SOC	**20.7**	**22.6**	**19.3**	**20.7**	**18.9**
Total Operating Costs	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 1.1: US Domestic Major Airline Costs - % Breakdown

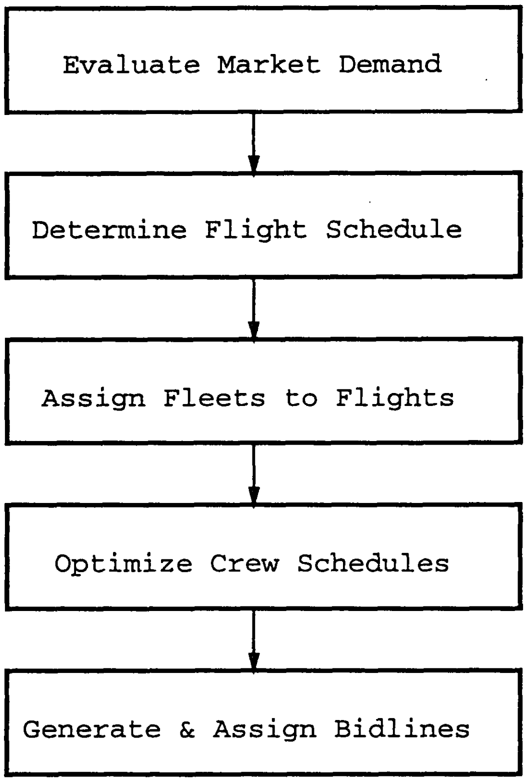


Figure 1-1: Overall View of the Problem - A Schematic Diagram

is made between optimality[1] and ease of computation. Computational difficulty is lessened by breaking down the overall problem into many phases. These phases are shown schematically in figure 1-1 and are typically as follows.

1. Firstly, market demand must be determined. This demand should be calculated for every two cities between which the airline wishes to fly. The methods used to calculate this demand can be found in Ben-Akiva and Lerman(1985). Elce(1970) states that forecasts are made by extrapolating historical trends and making modifications according to various assumptions.

2. The market demand forecasts are then used in designing flight schedules and aircraft routes. At this stage, the flights to be flown with their respective departure and arrival times are determined. No decisions are made about the fleet types,

[1]Hara and Koyama(1973) suggest a definition for optimality. They state that optimum is achieved when profits are maximized as a whole utilizing existing resources as much as possible under various existing restrictions.

sizes and assignments. Elce(1970), Hall(1989), and Balakrishnan, et al.(1990) suggest alternate methods for designing routes and schedules. Balakrishnan, et al., in particular, formulate the problem as a mixed-integer program and solve it using a Lagrangian based solution technique that exploits the special structure of the program.

3. Next, fleet sizing and assignment decisions are made. Fleet assignment involves the decision of what type of plane flies each flight (Elce,1970;Belgray,1970 and Abara,1988) and fleet sizing involves determining the number of aircraft of each fleet type. Agard (1970), Rapley(1975), Konig(1976), Girard(1973), and Subramanian, et al.(1993) suggest various methods to model and solve the fleet assignment problem.

4. The next step is to determine optimal crew pairings, i.e., optimal assignments of crews to scheduled flights. It is assumed that each crew can be assigned to only one kind of aircraft, that is, to one fleet type. So each aircraft fleet type is considered separately in determining crew assignments. A thorough description of this problem and a review of relevant literature are presented in later chapters.

5. Using the crew pairings generated in the previous step, the final step involves the generation of bidlines (Jones,1989). A bidline is a set of pairings that will be flown by one crew and represents the work to be done by the crew during the planning horizon. The final assignment of bidlines to crews is determined in part by seniority. A big difference between crew-pairing optimization and bid-line generation on one hand and fleet assignment and sizing on the other is that decisions about the fleet are often long term while decisions about crew schedules can be made in the short run.

Before closing this section, it should be mentioned that the above list of problems provides only a narrow view of the challenges faced by airlines. There are, in fact, many other interesting problems in the industry. For example, Vasquez-Marquez(1991) has implemented and developed a network-optimization based system to help reduce delays imposed by air traffic control, Teodorovic and Guberinic(1984) have shown how a new routing and scheduling plan for an airline fleet can be obtained in case of a delay in flight schedule, and Richetta and Odoni(1992,1993) solve the ground holding problem in air traffic control.

1.3 Focus and Outline of Thesis

This thesis studies the optimization of crew pairing problems. The problem is an intractible one and hence, special methods are developed for its solution. Various implementations of the solution techniques are tested empirically in a case study.

The crew pairing problem is formally defined in chapter 2. Alternate mathematical programming formulations for this problem and their relative merits are also discussed in this chapter. Chapter 3 presents the methodology that is used to solve similar large scale problems and presents relevant literature. Chapter 4 describes dual ascent heuristics that can be used to speed up the solution process. Chapter 5 presents our solution method for the crew pairing problem. Implementation issues are also detailed. Chapter 6 details a case study in which typical long haul carrier problems are solved and analyzed using the techniques presented. Chapter 7 presents further work.

Chapter 2

The Crew Pairing Problem Definition & Formulation

2.1 Introduction

This chapter introduces the crew pairing problem and provides some basic definitions. The difference between long-haul and short-haul problems is presented; the crew pairing problem is illustrated with examples; and finally, the crew pairing mathematical program is formulated.

2.2 Definitions

Before proceeding, some of the terms commonly encountered in the crew scheduling literature are presented(Minoux,1984;Rannou,1986).

1. **Flight Segment:** the smallest flight element between two successive stops, also called a **flight leg.**

2. **Duty Period:** a set of one or more flight segments assembled according to contractual rules, sometimes referred to as a **flight service.**

3. **Crew Base:** the city where the crew is domiciled, also called the **crew domicile.**

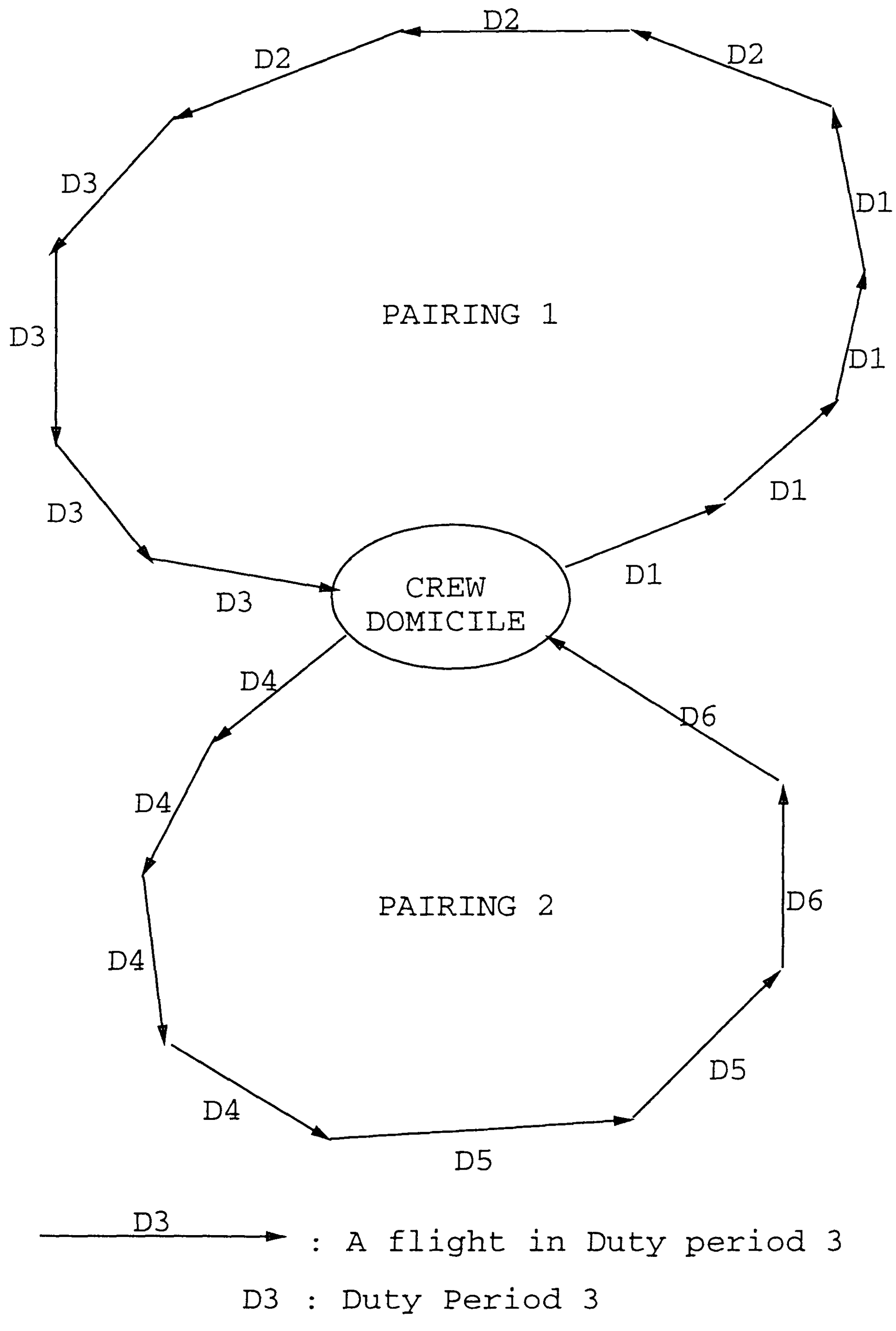


Figure 2-1: Hierarchial View of Pairings, Duty Periods and Flights

4. **Crew Pairing (CP):** a sequence of flight services with the first flight beginning and the last flight ending at the same crew base, sometimes referred to as a **crew rotation.**

5. **Time-Away-From-Base (TAFB):** the total duration of a pairing, that is, the elapsed time from departure to return of the crew, also referred to as the **absence time.**

6. **Overnight:** the time interval separating two successive duty periods in a crew pairing, also referred to as the **stop time.**

7. **Rest Time:** the idle time at a crew base after the last duty period in the crew pairing.

8. **Deadhead:** a flight segment used to position a crew from one station to another, the crew is transported on that flight segment as passengers.

9. **Brief Time:** the time taken to brief the crew before the first flight in a duty period.

10. **Report Time:** the time that the crew should report for briefing.

11. **Debrief Time:** the time taken by the crew for debriefing.

12. **Release Time:** the time the crew can leave for rest.

13. **Flying Time:** the total duration of a schedule spent in flying. It does not include the brief and debrief times.

14. **Minimum Connect Time:** the minimum time interval between two successive flight segments belonging to the same duty period.

15. **Maximum Sit Time:** the maximum time that a crew is allowed to wait on the ground between flights at a city other than the crew domicile.

16. **Maximum Pairing Length:** the maximum length of a valid pairing.

Typical values for some of these parameters in long haul problems are given in appendix A.

Example 1 *Figure 2-1 gives a diagrammatic view of the hierarchy of pairings, duty periods and flight segments (Ball and Roberts,1985). All the arcs in the figure represent flight segments. Each flight is given a number eg. D1, D2 etc. All flights with the same number belong to a single duty period. Thus, duty periods D1 through D6 contain 4, 3, 4, 4, 2 and 2 flights respectively. A crew pairing is a set of flights that start from the crew base (domicile) and end at the same crew base. There are two pairings shown in the figure. Pairing 1 contains 3 duty periods and 10 flights. Pairing 2 contains the same number of duty periods but only 8 flights.*

An interesting thing to note is that different terms are used to describe similar activities in other crew scheduling applications. For example, in the transit crew scheduling problem (Desrochers and Soumis,1989), a *vehicle block* is defined as a bus trip starting and ending at the depot. A block is divided by *relief points.* A *task* is that portion of a block between two consecutive relief points. A *piece of work* is one or more consecutive tasks performed by a driver. A *workday* consists of one or more pieces of work executed by the same driver.

2.2.1 Long Haul, Medium Haul, and Short Haul Crew Pairing Problems

Airline crew pairing problems are classified as either long haul, medium haul or short haul. Long haul problems usually deal with international flights. In long haul problems, pairings are as long as 12 to 15 days (Barnhart, et al.,1991), while short haul problems (also called *domestic* problems) involve short flights, usually within the country, and pairing lengths are usually between 2 and 4 days (Rannou,1986). For example, in the work done by Vance(1993), the maximum pairing length for American Airlines' domestic problem is 3 days. Most other crew pairing problems, i.e., those classified as medium haul problems, have maximum pairing lengths between those of long haul and short haul problems. For example, Odier, et al.(1983) solves a medium

Number of Flights	*Number of Duty Periods*	*Number of Pairings*
144	1795	93,851
174	2716	467,671
202	3203	1,878,614
253	4865	5,833,004

Table 2.1: Number of Duties and Pairings as Functions of Flights - Short Haul

haul problem in which the pairing length is kept close to 6 days but is not allowed to exceed 6 days.

In addition to maximum pairing length, several differences exist between short and long haul problems. One big difference is that, unlike long haul problems, the number of duty periods in domestic short haul problems combinatorially explodes as a function of the number of flights. Table 2.1 (Vance,1993) shows how the number of duty periods for short haul problems increases as a function of the number of flights. This explosion in size may make it difficult in short haul problems to even simply generate all possible duty periods. In long haul problems however, generation of all the duty periods is not problematic since, as reported by Barnhart, et al.(1991), the number of duty periods may be about twice the number of flights.

The two main reasons for this explosion in problem size in the domestic crew pairing problem are:

1. The flight network in the domestic short haul case is a hub and spoke network, while the long haul network is a point to point network. A hub is a central airport where a lot of flights land at about the same time and allow passengers to disembark. The planes that have landed depart together taking the passengers to their respective destinations. Due to the hub and spoke nature of the domestic network, a flight coming into a hub can connect with many other flights to form duty periods. Thus, many feasible duty periods are possible. Long haul flight networks on the other hand are typically relatively sparse and thus, the number of possible duty periods is relatively small.

2. The second reason is that many domestic flights are short in length. This implies

that several flights can be linked together to form a legal duty period. Long haul flights on the other hand are usually quite long and thus, the number of flights allowed in a duty period is quite restricted.

Finally, short haul and long haul problems have distinguishing properties that make long haul problems more amenable to efficient solution procedures. For example, an assumption made about pairing cost allows the use of a simple shortest path algorithm to generate pairings for long haul problems.

2.3 Problem Definition and Formulation

Barnhart, et al.(1991) defines the crew pairing problem as "*the construction of a set of valid crew pairings allowing an assignment of crews to pairings such that total costs are minimized, every flight is covered by one crew, and every crew is assigned to at most one pairing at any point in time*". The pairings should be valid pairings, i.e. they should satisfy rules governing the definition of legal duty and rest periods. As an example, a sampling of the rules that apply to the domestic operations of one U.S. airline are (Vance,1993):

1. A single pilot cannot fly more than eight hours in any twenty four hour period. This is often referred to as the "8 in 24" rule.

2. The elapsed time of a duty period cannot be more than ten hours and the total flying time of a duty period cannot be more than 8 hours.

3. A pilot can change or *swap* planes, within a duty period, only a limited number of times. The maximum number of swaps is fixed at one.

4. Given the elapsed and flying time of a duty period, there is a minimum amount of rest requried for the pilot after the duty period, where the hours of rest must be consecutive. The rest required is given by the table 2.2.

The rules for rest in the long haul problem are similar. The required rest increases with the length of the duty period. Single flight duty periods typically require short

Elapsed Time $\downarrow$	Minimum rest needed
Less than 8 hours	9 hours
Between 8 and 9 hours	10 hours
More than 9 hours	11 hours

Table 2.2: Rest Required After Flying a Duty Period

rest times, while duty periods with international flights require longer rest times. Long duty periods require the longest rest times. Rules for a typical long haul carrier are described in appendix B.

2.3.1 Pairing Cost Structure

Pairing cost structure is complicated. For example, in the domestic problem solved by Vance(1993), the cost of a crew pairing P, is a non-linear function of different costs. Specifially, the cost of pairing P is equal to

$$\max\{\sum_d MIN\_GRNT_d, \sum_d FLY\_TIME_d, TIME\_FROM\_BASE_p\} \tag{2.1}$$

where d is the set of duty periods in a pairing, MIN_GRNT_d is the minimum guarantee offered to the pilots for flying the duty period d, FLY_TIME_d is the prorated flying time of duty period d, and $TIME\_FROM\_BASE_p$ is the prorated time-away-from-base of pairing p. Observe that it is impossible to allocate one cost to each flight or duty period since multiple costs may be relevant, i.e., the cost of a flight depends on the pairing in which the flight is included. In long haul problems, however, due to the sparsity of the network, the length of the flights, and the fact that crews have relatively large mandatory rest periods, the time away from base cost component is typically dominant. Thus, an assumption that the long haul crew pairing cost is equal to the time away from base cost is not too restrictive.

2.3.2 Problem Formulation

The crew pairing problem, denoted CPP, can be formulated as a set partitioning problem (Arabeyre, et al.,1969; Barutt and Hull,1990; Marsten and Shepardson,1981 and Barnhart, et al.,1991):

$$(CPP) \quad \min \quad \sum_j c_j x_j \tag{2.2}$$

subject to

$$\sum_j a_{ij} x_j = 1 \quad i = 1, \ldots, m \tag{2.3}$$

$$x_j \in \{0, 1\} \quad j = 1, \ldots, n \tag{2.4}$$

where m is the number of flight segments, n is the number of crew pairings, a_{ij} is equal to 1 if the flight leg i is covered by the crew pairing j and 0 otherwise, c_j is the cost of crew pairing j and x_j is a 0-1 binary variable that has a value of 1 if a crew is assigned to pairing j (i.e. pairing j is selected in the current solution) and 0 otherwise.

From the above formulation, it can be seen that there are as many constraints as there are flights. The ith constraint, together with the binary restrictions on the decision variables, require that flight i be covered by exactly one pairing, thus ensuring that each flight is covered by one and only one crew. (Exactly one crew is associated with each pairing and hence, there are no variables or constraints for crews specifically, just pairings.) The objective function gives the cost of the current solution, with the optimal solution having minimal cost.

Example 2 *Table 2.3 shows a typical constraint matrix for the airline crew scheduling problem. This table has 5 rows and 7 columns (flights and pairings respectively). Each pairing is defined by the set of flights it covers. Pairing 1 covers the flights 1 and 4. Similarly, the pairings from x_2 through x_7 contain flights $\{2,4\}$, $\{3\}$, $\{1,2\}$, $\{4,5\}$, $\{2,5\}$, and $\{2,3\}$ respectively. This example is that of a set-partitioning problem and each flight has to be covered exactly once. One feasible solution is pairings x_3, x_4, and x_5. Another feasible solution is x_1, x_3, and x_6.*

	x_1	x_2	x_3	x_4	x_5	x_6	x_7		
1	1	0	0	1	0	0	0		1
2	0	1	0	1	0	1	1	=	1
3	0	0	1	0	0	0	1		1
4	1	1	0	0	1	0	0		1
5	0	0	0	0	1	1	0		1

Table 2.3: A Typical Crew Pairing Constraint Matrix

2.3.3 Problem Formulation with Deadheading

The set-partitioning formulation can be enhanced to allow the possibility of *deadheading.* Crew deadheading means repositioning crews to increase their utilization (Barnhart, et al.,1991). While deadheading can be expensive because deadheading crews are paid their full wages and they remove seats from inventory, deadheading may be advantageous. Specifically,

1. **Deadheading may be cost effective:** Since the long haul crew scheduling flight networks are relatively sparse, a crew may land at a city, be forced to depart that flight to rest, and then wait until the next flight is scheduled to depart that city before resuming work. In these cases, it may be more economical for a crew to deadhead out of that city rather than wait for the next scheduled flight.

2. **Deadheading may be necessary for feasibility:** Consider, for example the problem given in figure 2.4 with no feasible integer solution. If deadheading is allowed however, $\{x_1, x_2, x_3\}$, $\{x_1, x_2, x_4\}$, and $\{x_1, x_2, x_3, x_4\}$ are feasible solutions.

To model deadheading, the CPP can be modified by adding surplus variables with cost equal to those of deadheading. The modified formulation, denoted CPPD is:

$$(CPPD) \quad \min \quad \sum_j c_j x_j + \sum_i d_i y_i \tag{2.5}$$

$$\begin{array}{c|cccc|cc}
 & x_1 & x_2 & x_3 & x_4 & & \\
\hline
1 & 1 & & & & & 1 \\
2 & 1 & 1 & & & = & 1 \\
3 & & 1 & & & & 1 \\
4 & & & 1 & 1 & & 1
\end{array}$$

Table 2.4: An Infeasible Problem

subject to

$$\sum_j a_{ij}x_j - y_i = 1 \quad i = 1,\ldots,m \tag{2.6}$$

$$x_j \in \{0,1\} \quad j = 1,\ldots,n \tag{2.7}$$

$$y_i \geq 0 \quad i = 1,\ldots,m \tag{2.8}$$

where y_i is a variable that indicates the number of times flight i is deadheaded, d_i is the cost of deadheading flight i, and the rest of the variables are as defined in formulation CPP (equations 2.2 - 2.4).

2.3.4 Formulation Drawback

The CPP formulation has the drawback that, while the number of rows (flights) is usually manageable, the number of columns is extremely large. For example, (Barutt and Hull,1990) report that the number of columns for the domestic problem can be as large as $e^{24,000}$. As another example, table 2.1 shows how the number of pairings for the domestic operation of a U.S. airline increases with an increase in the number of flights (Vance,1993). The number of pairings is about 6 million for a set of only 253 flights, and the total number of domestic flights is typically in the order of thousands. The reason for this explosion is that a crew pairing corresponds to a subtour in the flight network. From graph theory, it is known that the number of tours in a graph is exponentially large. This implies that the number of subtours is also exponential in size.

Hence, it is not just inefficient, but also impractical, to solve CPP directly for problems of the size encountered by many airlines. This motivates the need for a solution technique that does not require explicit consideration of all the variables. Column generation, discussed in the next chapter, is one method that achieves this.

Chapter 3

Review of Relevant Methodologies and Literature

This chapter presents exact and heuristic methodologies that are useful in solving the crew pairing problem. In particular, discussed are the Branch and Bound algorithm for solving integer programs (IP's), the Column Generation algorithm for solving linear programs (LP's), Shortest Path algorithms, and a technique that combines branch and bound and column generation and gives exact solutions to large IP's.

3.1 Heuristics for the Crew Pairing Problem

Early methods used to solve crew pairing problems were usually inexact. The main reason was the dearth of computing time. A survey of the methods used is given in Arabeyre, et al.(1969) and Etschmaier and Mathaisel(1985). Although many of these methods are obsolete due to the rapid advances made in computer science and technology, insights may be gained by studying them.

Baker and Fisher(1981) model the airline crew pairing problem as a set covering problem. They solve this problem using a heuristic that simply attempts to cover uncovered flights by crew pairings with the minimum cost per uncovered flight leg. Baker(1981b) presents efficient heuristic algorithms and Baker, et al.(1979) study efficient heuristics to the airline crew scheduling problem. Odier (1983) solves the medium haul crew pairing problem as a linear program and generates all possible 1-day and 2-day pairings by brute force. Ball and Roberts (1985) present a graph partitioning approach to the crew pairing problem, Thiriez (1969) uses a group theoretic approach, and Marsten and Shepardson (1981) use lagrangean relaxation and subgradient optimization to heuristically solve the crew pairing problem.

Heuristic methods, however, have the disadvantage that there is no method of measuring how much the solution can be improved and hence, no method of ensuring global optimality. This motivates the need for solution techniques that are guaranteed to achieve exact optimal solutions to the CPP. Optimal solutions can be obtained using an IP solution framework using branch and bound, column generation, and shortest path problems. These methods are reviewed in the following sections.

3.2 Branch and Bound

Branch and bound is a "divide and conquer" strategy (Bradley, et al.,1977) that can find the optimal solution of an IP. The first step in the solution process is to relax the integrality constraints and solve the resulting LP. This corresponds to solving the LP at the root node of the branch and bound *enumeration tree.* If the optimal solution of the root node LP satisfies the integer constraints, then the solution is also optimal to the IP.

If however, at least one variable x_j^* in the optimal root node LP solution is nonintegral, the variable x_j can be used in a *branching rule* to partition the feasible IP solution space. A branching rule is a pair (or set) of constraints that divide the feasible IP solution space into mutually exclusive, collectively exhaustive regions. Each subdivision of the feasible space corresponds to a node in the branch and bound tree.

There are a number of ways to subdivide the feasible region, i.e., to perform branching and thus, there are a variety of branching strategies. Figure 3-1 shows two nodes of a branch and bound tree, both of which represent the root node LP with one added constraint. The right node has the added constraint $x_j \geq \lceil x_j^* \rceil$, while the left node has the added constraint $x_j \leq \lfloor x_j^* \rfloor$. Assume that the problem is to *minimize* some objective function.

An LP is solved at each node in the branch and bound tree with four possible outcomes, namely:

1. the LP is infeasible;
2. the optimal value of the LP is worse than the best integer solution value found so far;
3. the optimal value of the LP is better than the best integer solution value so far and the LP solution is integral; or
4. the optimal value of the LP is better than the best integer solution so far but the LP solution is not integral.

Outcomes 1, 2 and 3 bound the node. Outcome 1 indicates that the feasible IP solution space at that node is empty and hence further exploration of the node is not possible. Since an LP solution gives a lower bound (in a minimization problem) on the best IP solution, outcome 2 indicates that further exploration of the node will not yield better results. Outcome 3 indicates that an improved IP solution has been determined and again further exploration of the node in unnecessary. Outcome 4 indicates non-integrality and further branching is required.

3.2.1 Example of Branch and Bound

Consider the following *maximization* IP(Bradley, et al.,1977). The (partial) solution procedure is shown systematically in the steps below and diagrammatically in figure 3-2.

$$\max z = 5x_1 + 8x_2 \tag{3.1}$$

subject to

$$x_1 + x_2 \leq 6 \tag{3.2}$$

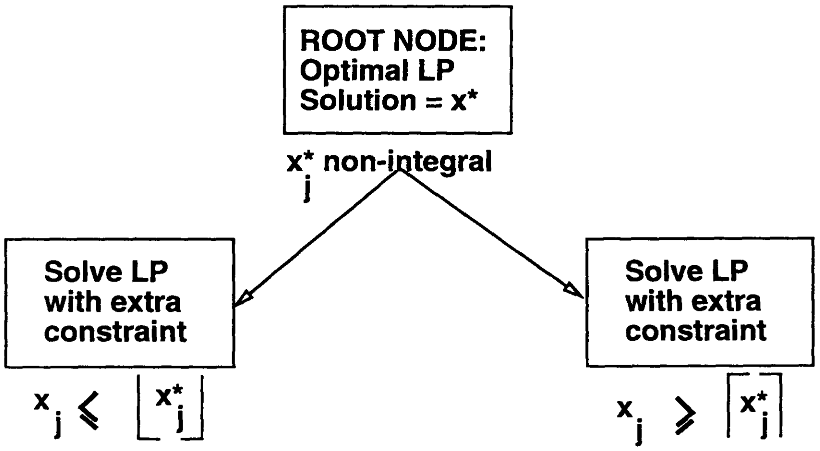

ROOT NODE: Optimal LP Solution = x*
x^*_j non-integral
Solve LP with extra constraint
Solve LP with extra constraint
$x_j \leq \lfloor x^*_j \rfloor$
$x_j \geq \lceil x^*_j \rceil$

Figure 3-1: Branch and Bound Logic

$$5x_1 + 9x_2 \leq 45 \tag{3.3}$$

$$x_1, x_2 \geq 0 \tag{3.4}$$

$$x_1, x_2 \in Z \tag{3.5}$$

1. **Solve the Root Node:** The integrality constraints are relaxed and the optimal solution of the LP relaxation has an objective function value of $41\frac{1}{4}$. This is a upper bound on the optimal IP solution. The values of x_1 and x_2 are $2\frac{1}{4}$ and $3\frac{3}{4}$ respectively.

2. **Branching on Node 1:** x_2 is chosen (arbitrarily) for branching. Branching creates two active nodes (nodes that have not been *explored*), namely nodes 1 and 2. Node 1 has the added constraint $x_2 \leq 3$ and node 2 has the added constraint $x_2 \geq 4$.

3. **Solving Node 1:** Node 1 is chosen (again arbitrarily) for exploration. Solving the LP associated with node 1 gives an optimal objective function value of 39 and x_1 and x_2 have values of 3 each.

4. **Bounding Node 1:** The LP solution at node 1 is integral and being the first one found, is the best IP solution so far. This provides a lower bound on

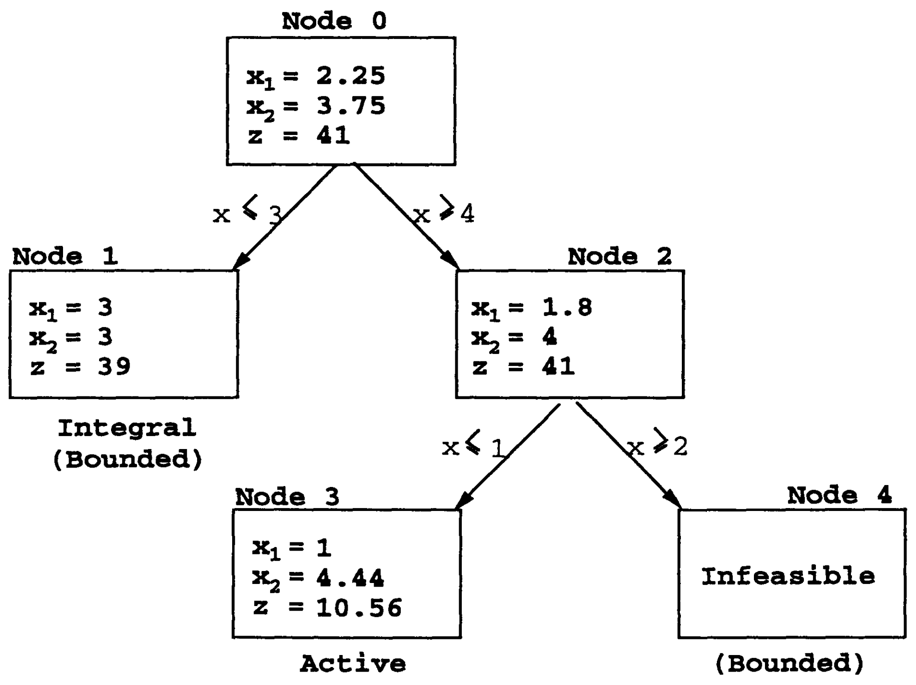


Figure 3-2: (Partial) Branch and Bound Example

the global integer solution value. Node 1 needs no further exploration and is bounded.

5. **Solving Node 2:** The LP associated with node 2 is solved and the optimal LP solution has an objective function value of 41. The values of x_1 and x_2 being $1\frac{4}{5}$ and 4 respectively.

6. **Branching on Node 2:** Variable x_1 is not integral at node 2. Node 2 is branched into nodes 3 and 4, with the added constraints $x_1 \leq 1$ and $x_1 \geq 2$ respectively.

The solution procedure continues in this manner until optimality is achieved.

This example shows that there are two decisions that are made many times in this algorithm, namely:

1. **Branching decision:** This involves deciding the node on which to branch.

2. **Choice of node:** This involves deciding which active node to choose for exploration.

In this example, the decisions were made arbitrarily.

3.3 Column Generation

The solution of the crew pairing problem, using the branch and bound algorithm, involves solving at each node (of the branch and bound enumeration tree) a linear relaxation of the crew pairing IP with some added constraints. The enormity of the number of variables makes it difficult to solve the LP relaxation of the crew pairing problem using traditional methods, such as the SIMPLEX algorithm. This motivates the use of techniques, such as column generation, which do not require explicit enumeration of the entire constraint matrix.

Dantzig and Wolfe (1960,1961) developed a technique to solve large, specially structured LP's. Their technique solves the LP by alternately solving a coordinating restricted master problem and smaller linear sub-problems. Column generation methods, based on the decomposition principle of Dantzig and Wolfe, recognize that it is not necessary to have the entire constraint matrix available during the time of computation; columns need be generated only as and when "necessary" (Ahuja, et al., 1993). Column generation and decomposition are sometimes called generalized linear programming(GLP) (Wolfe, in Dantzig,1963).

3.3.1 The Principle of Column Generation

Consider the following linear program, denoted as the Master Problem (MP), where the number of variables, or columns, n, is very large (Bradley, et al. 1977).

$$
\begin{array}{llllll}
z^* = & \min z = & c_1x_1 & +c_2x_2 & +\dots & +c_nx_n \\
\text{subject to} & & & & & \\
 & & a_{i1}x_1 & +a_{i2}x_2 & +\dots & +a_{in}x_n \quad = b_i \quad (i = 1, 2, \dots, l) \\
 & & & & & x_j \geq 0 \quad (j = 1, 2, \dots, n)
\end{array}
$$

An assumption can be made *a priori* that certain variables, $x_{m+1}, x_{m+2}, \ldots, x_n$ are nonbasic, thereby defining a restricted problem, now called the Restricted Master Problem (RMP), as:

$$\begin{array}{lllll} z^* = & \min z = & c_1x_1 & +c_2x_2 & + \ldots & +c_mx_m \end{array}$$

subject to

$$\begin{array}{lllll} a_{i1}x_1 & +a_{i2}x_2 & + \ldots & +a_{im}x_m & = b_i \quad (i = 1, 2, \ldots, l) \\ & & & & x_j \geq 0 \quad (j = 1, 2, \ldots, m) \end{array}$$

The solution to the RMP, if feasible, may be optimal to the MP. Let $\pi_1, \pi_2, \ldots, \pi_l$ denote the optimal dual variables for the RMP. The reduced cost c_j of variable j is given by:

$$\bar{c_j} = c_j - \sum_{i=1}^{l} \pi_i a_{ij} \tag{3.6}$$

From linear programming theory, if the reduced cost of each variable is nonnegative, then the RMP solution is optimal to the MP. Thus, to determine if optimality of the MP is achieved, the following Subproblem (denoted as SP) is solved:

$$w^* = \min_{j=1,\ldots,n} [c_j - \sum_{i=1}^{l} \pi_i a_{ij}] \tag{3.7}$$

If, in the solution, $w^* \geq 0$, the RMP solution is optimal to the MP. Otherwise, if $w^* < 0$, column k (with $\bar{c_k} < 0$) has been identified, and it is added to the RMP. The RMP is resolved, and the whole process is repeated until no negative reduced cost variables are identified and optimality is reached.

Column generation (for minimization problems) can be summarized as:

- **STEP 0:** Find a feasible starting subset **R** of columns.

- **STEP 1:** Solve the RMP to optimality over the restricted subset **R** and obtain dual prices.

- **STEP 2:** Use the dual prices from step 1 and solve the SP (equation 3.7) to find a new column with minimum reduced cost.

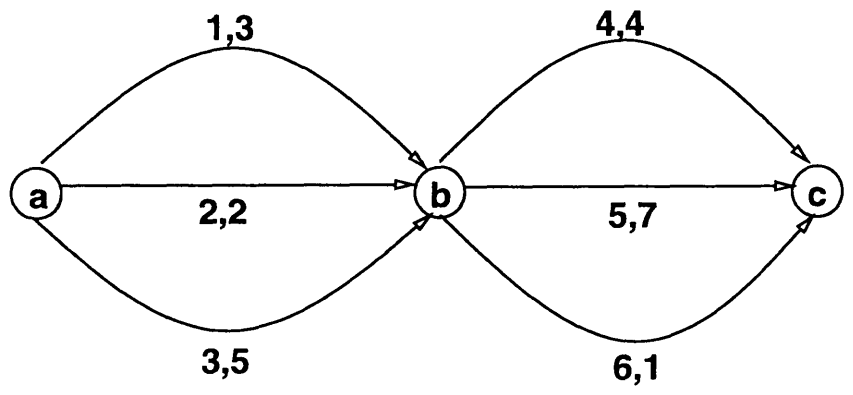


Figure 3-3: Column Generation Example Network

- **STEP 3:** If the minimum reduced cost column has a positive reduced cost, then STOP since global optimality is reached. Otherwise, add the minimum reduced cost column to the restricted subset **R** and go to step 1.

3.3.2 An Example of Column Generation

Column generation can be demonstrated by a contrived but simple example. Consider the network shown in figure 3-3. Each network arc has two numbers associated with it. The first number corresponds to the arc number and the second number corresponds to the arc cost. There are 9 possible paths from node a to node c. Suppose the problem is to *find a minimum cost set of paths from node a to node c that cover arcs 1,2 and 3 at least once.* By simple inspection, the optimal solution is a set of three $(a - c)$ paths, namely $\{1,6\},\{2,6\}$, and $\{3,6\}$, with a cost of 13. For the sake of exposition, consider the mathematical programming formulation for this problem.

If all columns are generated, the constraint matrix for the MP, shown in table 3.1, has 6 rows (one for each arc) and 9 columns (one for each $a - c$ path). Each column j contains a '1' in the row corresponding to arc i, if the $a - c$ path covers arc i, and a '0' otherwise. The cost of each column is next to the variable name in the first row of the matrix. Assume that it is not possible to enumerate all columns explicitly and that the RMP is given in table 3.2. There are 6 columns in the initial RMP, namely, paths $\{1,4\}$, $\{1,5\}$, $\{2,4\}$, $\{2,5\}$, $\{3,4\}$, and $\{3,5\}$.

The algorithm proceeds as follows:

	$7x_1$	$10x_2$	$6x_3$	$9x_4$	$9x_5$	$12x_6$	$4x_7$	$3x_8$	$6x_9$		
1	1	1					1			$\geq$	1
2			1	1				1		$\geq$	1
3					1	1			1	$\geq$	1
4	1		1		1					$\geq$	0
5		1		1		1				$\geq$	0
6							1	1	1	$\geq$	0

Table 3.1: The Master Problem

	$7x_1$	$10x_2$	$6x_3$	$9x_4$	$9x_5$	$12x_6$		
1	1	1					$\geq$	1
2			1	1			$\geq$	1
3					1	1	$\geq$	1
4	1		1		1		$\geq$	0
5		1		1		1	$\geq$	0
6							$\geq$	0

Table 3.2: The Restricted Master Problem

1. **Step 0** A feasible starting set of columns, denoted R, is chosen as in table 3.2.

2. **Step 1** The RMP is solved over R and the optimal solution has a cost of 22. Variables x_1, x_3, and x_5 are equal to 1 and the rest are equal to zero. The vector of optimal dual prices is $\{7, 6, 9, 0, 0, 0\}$.

3. **Step 2** The costs on all the arcs are modified by deducting from the actual cost the dual price associated with the arc. The vector giving the reduced costs on all the arcs is $\{-4, -4, -4, 4, 7, 1\}$. SP is solved and the solution is path $\{1, 6\}$, with a reduced cost of -3.

4. **Step 3** Since the reduced cost of $\{1, 6\}$ is negative, optimality is not yet achieved. The column corresponding to path $\{1, 6\}$ is added to R and step 1 is repeated.

5. **Step 1** The RMP (now with 7 columns) has an optimal solution with value 19 (an improvement). Variables x_3, x_5, and x_7 are equal to 1 each, and all other variables are zero. The optimal vector of dual prices is $\{-4, -6, -9, 0, 0, 0\}$.

6. **Step 2** The vector of reduced costs on the network arcs is $\{-1, -4, -4, 4, 7, 1\}$. The SP solution is path $\{2, 6\}$, with a reduced cost of -3.

7. **Step 3** Since the reduced cost of path $\{2, 6\}$ is negative, a column corresponding to path $\{2, 6\}$ is added to R and the new RMP is solved.

8. **Step 1** The RMP (with 8 columns) has an optimal objective function value of 16(another improvement). Variables x_5, x_7, and x_8 are equal to 1 each, and all the other variables are zero. The optimal vector of dual prices is $\{-4, -3, -9, 0, 0, 0\}$.

9. **Step 2** The vector of reduced costs on the network arcs is $\{-1, -1, -4, 4, 7, 1\}$. The SP solution is path $\{3, 6\}$, which has a reduced cost of -3.

10. **Step 3** Since path $\{3, 6\}$ has a negative reduced cost, it is added as a column to R, and RMP is resolved.

11. **Step 1** The RMP (with 9 columns) has an optimal objective function value of 13 (another improvement). Variables x_7, x_8, and x_9 are equal to 1 each, and all the other variables are zero. The optimal vector of dual prices is $\{-4, -3, -6, 0, 0, 0\}$.

12. **Step 2** The vector of reduced costs on the network arcs is $\{-1, -1, -1, 4, 7, 1\}$. The SP solution is path $\{1, 6\}$, with a reduced cost of zero.

13. **Step 3** Since the SP solution has a non-negative reduced cost, global optimality is achieved, and the procedure stops.

The fact that integral solutions are always obtained is purely coincidental.

3.3.3 Applications of Column Generation

Column generation is a widely applied technique. For example, column generation has been applied to crew scheduling by Desrosiers, et al. (1991), Rubin (1973), Anbil, et al. (1991), Rannou (1986), Minoux (1984), Crainic and Rousseau (1987), Lavoie, et al. (1988), Barnhart, et al. (1991), Barutt and Hull (1990), and Vance (1993). Desrochers, et al. (1992) apply column generation to the vehicle routing problem with time windows, while Desrosiers, et al.(1984,1986) use column generation in routing problems with time windows. Desrochers and Soumis (1989) solve the urban transit crew scheduling problem with column generation and Ribeiro, et al.(1989), Parker and Ryan(1993) and Barnhart, et al. (1991c) use the column generation method to solve problems arising in communication systems optimization. Gilmore and Gomory (1961) and Vance(1993) use column generation to solve the cutting stock problem.

3.4 Shortest Path Problems

Column generation relies on the fact that all columns need not be considered explicitly to solve an LP. Implicit evaluation of the columns can be done with the use of the column generation subproblem as given in equation 3.7. The following sections show how the subproblem encountered in the column generation solution of the CPP, denoted CPSP, can be solved as a shortest path problem. An implementation scheme for the solution of the CPSP is presented with an example. This is followed by a literature review of relevant shortest path procedures.

3.4.1 The Column Generation Subproblem as a Shortest Path Problem

This section shows how the CPSP, can be solved *implicitly*, i.e., without explicitly evaluating the reduced cost of *every* variable. In solving the CPSP, if the minimum reduced cost among **all** possible columns is non-negative, then MP is solved. CPSP can be solved using a shortest path procedure on a time-line network. This is illustrated with an example.

The Time-Line Network

The following implementation of the CPSP is called a time-line network (Barnhart, et al.,1991). A typical example of a time line network is given in figure 3-4 (Barnhart, et al.,1991). In a time-line network each duty period is represented by an arc. The start node of the arc represents the starting place (city) and starting time of the first flight in the duty period. The end node of the arc represents the ending place of the last flight in the duty period. The time of the end node is the sum of the ending time of the last flight and the minimum rest required after the duty period. The advantage of adding the rest time to the duty period arc is that while running the shortest path algorithm, once a duty period arc is traversed, a minimum rest is guaranteed for the crew that has flown the duty period. This considerably simplifies modeling the rules of rest for the crew. Once all the arcs are created for the duty periods, all the nodes at any given city are sorted in increasing order of time. Chronologically successive pairs of nodes are connected by arcs (called *ground arcs* since they represent a period of time in which the crew is not flying and is on the ground).

For long haul problems, it is often the case that pairing cost is determined by the time-away-from-base cost component. Hence, the costs on all duty period arcs

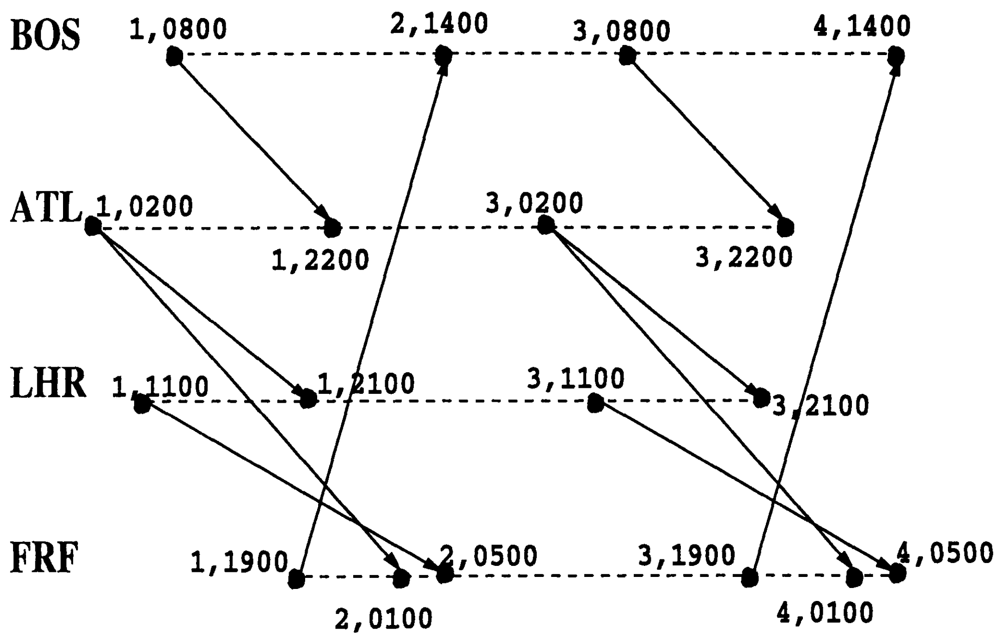


Figure 3-4: A Time-Line Network of Flights

are set equal to the time-away-from-base cost associated with that duty period. The reduced cost of each arc is then easily quantified by subtracting the appropriate dual variables from the arc cost. For example, a duty period d (with cost c_d) consisting of three flights a, b, and c (with dual variables π_a, π_b, and π_c respectively) will have a reduced cost $\bar{c}_d$ given by

$$\bar{c}_d = c_d - \pi_a - \pi_b - \pi_c \tag{3.8}$$

By running the shortest path procedure between each crew base node pair using arc costs equal to the reduced costs, the minimum reduced cost pairing can be generated. If w^* (the minimum reduced cost from equation 3.7) is non-negative, then all pairings have non-negative reduced costs and hence the column generation algorithm can be terminated. Otherwise, negative reduced cost pairings are identified and added as columns to the restricted master problem, and the next iteration of the algorithm can be executed.

An Example of the CPSP

Consider a CPP containing three cities, namely **A**, **B**, and **C**; two crew bases, namely **A** and **C**; and seven duty periods, numbered 1 through 7. Let the reduced costs of duty periods 1 through 7 equal -3, -8, 4, 6, -7, -4, and 9 respectively. A time line network is used to solve the CPSP by *implicitly* finding the minimum negative reduced cost pairings. The time-line network for this problem is illustrated in figure 3-5. The duty period number (reduced cost of each duty period) is depicted to the left (right) of each duty period arc. Assume that the reduced cost of each ground arc is zero. Then, the possible crew pairings are

1. $2 \rightarrow 4$ (reduced cost = -2)
2. $2 \rightarrow 7$ (reduced cost = 1)
3. $2 \rightarrow 3 \rightarrow 5 \rightarrow 7$ (reduced cost = -2)
4. $1 \rightarrow 3$ (reduced cost = 1)
5. $1 \rightarrow 4 \rightarrow 6$ (reduced cost = -1)

Of these, pairings 1,2 and 3 begin and end at crewbase **A** and pairings 4 and 5 begin and end at crewbase **C**. Observe that

1. each of the pairings correspond to a path in the time line network, and
2. the minimum reduced cost pairing (i.e., the CPSP solution) is a shortest path in the time line network with arc lengths equal to the reduced costs.

This implies that CPSP can be solved by solving a series of shortest path problems, one for each pair of nodes belonging to the same crew base.

Handling Special Constraints

The CPSP is usually a *constrained* shortest path problem. One typical constraint restricts the time-away-from-base of each pairing to be less than some maximum allowable time, denoted by MAX_ELAPSE. This constraint can be handled easily

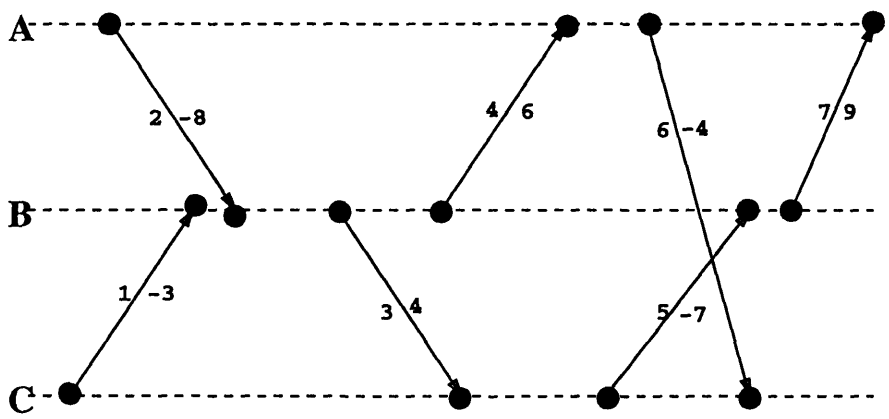


Figure 3-5: A Shortest Path Subproblem

using the time-line representation by the following procedure. For a given source node, any node whose associated time is greater than the sum of the time associated with the source node and MAX_ELAPSE is eliminated. The shortest path procedure is run on the remaining nodes.

However, various other additional constraints cannot be handled efficiently. For example, if the pairing cost structure is given by equation 2.1, the reduced cost of a duty period may be one of several values. Then, a multiple label shortest path procedure must be employed to solve CPSP. As an other example, there may be an excessive number of duty periods and it may not be efficient to include all of them in the time-line network for the solution of the CPSP. The next sections detail simple, constrained, and multi-label shortest path procedures that may have to be used in solving CPSP.

3.4.2 Literature Review

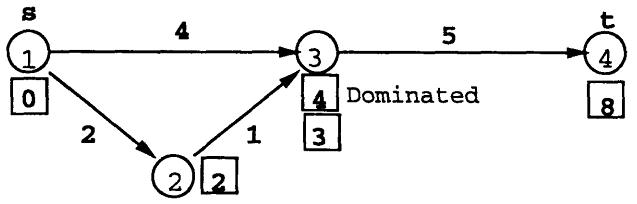
Shortest path problems, fundamental problems in operations research, are frequently encountered in numerous transportation and communication applications. Deo and Pang (1984) develop a thorough classification scheme and provide a comprehensive and updated bibliography for these problems. One of the main reasons for the importance given to shortest path problems is that they are repeatedly solved in larger problems. Hence even a small improvement in the performance of the shortest path procedure can result in big improvements in overall performance.

Very often, in real world applications, shortest path subroutines may be quite complicated due to additional constraints or multiple optimality criterion. Such problems are computationally much more difficult than simple shortest path problems. Simple (or unconstrained) shortest path problems involve only the determination of shortest (least cost) paths with no additional considerations and can be solved in polynomial time, while constrained shortest path problems or multiple-criterion shortest path problems may take exponential time.

To illustrate, consider an ordinary shortest path problem, where the label at a node gives the length of the current shortest path from the source node to the given node. Since each label contains only one cost, it can without ambiguity, dominate or be dominated by another label. Specifically, in determining a cheapest path, a cheaper label dominates a costlier one.

In the case of constrained shortest path problems, at a single node there may be a set of labels, each of which corresponds to a different path and none of which is dominant. Such a set of labels is said to be *efficient* (Desrochers and Soumis,1988,1988b). An efficient path is one that consists only of efficient labels. So one label can dominate another one only if all the costs on the label are respectively better than the costs on the other label. Suppose neither of the labels can dominate each other, then both the labels have to be stored at the node. Hence, it is possible that all the paths into a node result in efficient labels. Since theoretically there are an exponential number of paths in a network, an exponential number of labels may exist and hence, the algorithm can take exponential time.

The following sections discuss simple, multi-criterion and constrained shortest path problems. Constrained and multi-criterion shortest path problems can be solved using methods such as the ones described in Desrochers and Soumis(1988,1988b).



Shortest path from s->t: 1 -> 2 -> 3 -> 4

Figure 3-6: Simple Unconstrained Shortest Paths

Simple Shortest Path Problems

Figure 3-6 provides an example of a simple unconstrained shortest path problem. There are four nodes and four arcs in this network. The number on each node indicates the node number and the number on each arc indicates the cost on the arc. The problem here is to find the shortest path between the source node (1) and the sink node (4). As can be seen from the figure, there are two labels at node 3. One label has a cost of 4 and the other one has a cost of 3. A cost of 3 is better in a cheapest path problem and hence the label which contains a cost of 4 is dominated and discarded. Thus each node is associated with only one label and that label indicates the cost of the current shortest path from the source node to that label.

Denardo and Fox(1979) study the general shortest path problem and describe solution techniques called *reaching* and *pulling*. The algorithms (in pseudocode) are as follows:

```
Pulling
    1. Set $v(1) = 0, v(k) = +\infty$ for $k \neq 1$.
    2. DO for $j = 2, \ldots, N$
        3. DO for $i = 1, \ldots, j-1$
            $v(j) \leftarrow \min\{v(j), v(i) + t_{ij}\}$
```

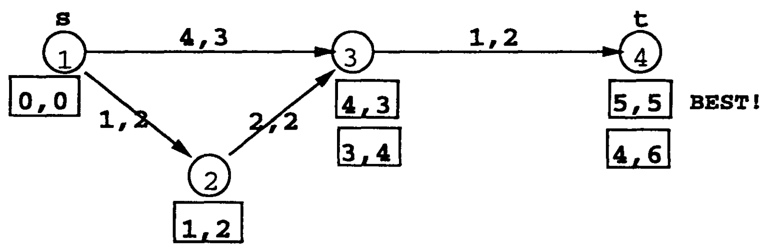


Figure 3-7: Bi-Criterion Shortest Paths

```
Reaching
    1. Set $v(1) = 0, v(k) = +\infty$ for $k \neq 1$.
    2. DO for $i = 1, \ldots, N-1$
        3. DO for $j = i+1, \ldots, N$
            $v(j) \leftarrow \min\{v(j), v(i) + t_{ij}\}$
```

Pulling is also referred to as *dynamic programming.* Dynamic programming is used to solve a lot of shortest path problems (as in Houck, et al.,1980). Reaching is a label setting scheme. Dijkstra's algorithm(1959) is one of the first implementations of reaching. Shepardson and Marsten(1980), using *Forward-Backward* reaching, solve the shortest path problem first using a forward reaching algorithm (Denardo and Fox,1979) and then using a backward reaching algorithm. This results in each node having a forward label indicating the cost of the cheapest path to that node from the source node and a backward label indicating the cost of the cheapest path from that node to the sink node. Given an arc cost, a forward label at the start node of the arc and a backward label at the end node of the arc, it becomes easy to test if the arc can improve the the current best solution of the shortest path subroutine.

Multi-criterion Shortest Path Problems

Multi-criterion shortest path problems arise in crew pairing optimization as follows.

Example 3 *Consider the domestic crew pairing problem (Vance,1993) in which minimum cost pairings are to be determined. The cost of a pairing is the maximum of* *three costs, namely, a prorated time-away-from-base cost, a cost corresponding to the prorated flying time in the pairing, and the sum of minimum guarantees for each duty period in the pairing. This problem can be solved by constructing an appropriate network and using a tri-criterion shortest path solution procedure.*

A bi-criterion shortest path problem is a multi-criterion shortest path problem with two criteria. Its solution procedure is demonstrated using an example given below. Figure 3-7 depicts a network containing two costs per arc. At each network node, there are labels, each label with two costs; the first (second) representing the sum of the first (second) costs on each arc leading from the source node to the node holding the label.

As the algorithm reaches out on an arc from a label, the costs on the arc are added to the respective costs on the label. In this problem, the cost of a path between two points is the maximum of the two cost sums. The progress of the algorithm will be given step by step.

1. **Initiate:** The source node (1) is given a label of $[0, 0]$ and all other nodes are given large labels ($[\infty, \infty]$).

2. **Reach:** Reaching out from node 1 to nodes 2 and 3 generates labels $[1, 2]$ and $[4, 3]$ at the respective nodes.

3. **Dominance check:** Label $[1, 2]$ dominates label $[\infty, \infty]$ at node 2, and label $[4, 3]$ dominates label $[\infty, \infty]$ at node 3.

4. **Reach:** Reaching out from node 2 to node 3 generates a label $[3, 4]$ at node 3.

5. **Dominance check:** Neither label $[4, 3]$ nor label $[3, 4]$ can dominate the other (since the first cost on $[4, 3]$ is greater than the first cost on $[3, 4]$, but the second cost on $[4, 3]$ is less than the second cost on $[3, 4]$) and hence both are efficient. Both labels are stored at node 3.

6. **Reach:** Reaching out from node 3 to node 4 generates 2 labels at node 4, namely $[5, 5]$ and $[4, 6]$.

7. **Dominance check:** Both [5,5] and [4,6] are efficient at node 4 and both dominate label $[\infty,\infty]$.

8. **Reach:** No more reaching can be done. STOP.

When the algorithm terminates, all the labels at the sink node are scanned and the cost corresponding to each label is evaluated. Since, in this example the cost of a label is the maximum of the costs in the label, label [5,5] will have a cost of 5 and label [4,6] will have a cost of 6. Thus label [5,5] corresponds to the shortest path label. The shortest path is $1 \rightarrow 3 \rightarrow 4$.

Constrained Shortest Path Problems

A constrained shortest path problem is one which involves the determination of the shortest path that satisfies some additional constraints. What makes this problem difficult is the fact that the shortest path between two nodes may not satisfy the extra constraints. All possible paths between the two nodes must be implicity evaluated and the shortest feasible path must be obtained.

Example 4 *While solving the long haul crew pairing problem Barnhart, et al.(1991) solve a constrained shortest path problem that requires that*

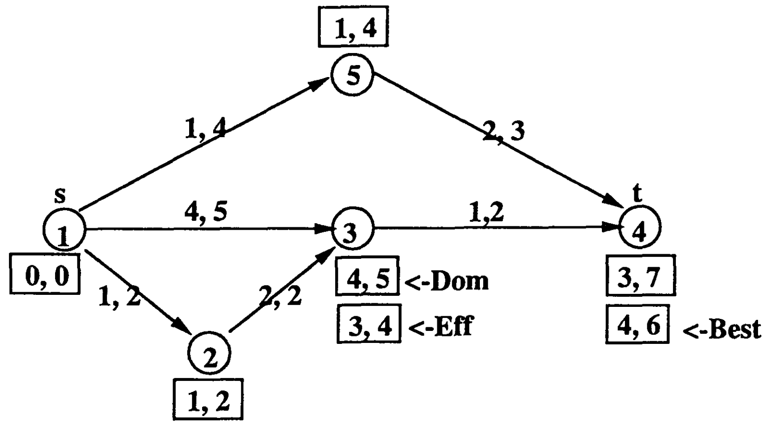
1. *the maximum time-away-from-base must not exceed 15 days;*

2. *a maximum of 8 flying hours must occur in every 24 hours;*

3. *24 hours of consecutive rest must occur in a week.*

Example 5 *The shortest path problem with time windows (SPPTW) is a commonly encountered problem that is solved using constrained shortest path procedures. The objective is to find the shortest path between two or more nodes in a network, while satisfying constraints that some or all of the nodes have to be serviced within their respective time windows (Desrochers and Soumis,1988).*

Figure 3-8 can be used to demonstrate the solution procedure of a constrained shortest path problem. Assume that the cost of a path is the sum of the second costs



Shortest Path from s->t: 1->2->3->4

Figure 3-8: Constrained Shortest Paths

on all the arcs that make up the path, and that the first cost is a resource that is being consumed by using that arc. Suppose the problem is defined as follows.

Example 6 *Find the least cost path (the cost being the sum of the second costs on all the arcs that constitute the path) between nodes 1 and 4 which satisfies the constraint that the sum of the utilized resources (i.e., the first costs on the arcs) on the path is less than or equal to 4.*

The progress of the algorithm is shown below.

1. **Initiate:** The source node (1) is given a label of $[0, 0]$ and all other nodes are given large labels ($[\infty, \infty]$).

2. **Reach:** Reaching out from node 1 to nodes 2, 3 and 5 generates labels $[1, 2]$, $[4, 5]$, and $[1, 4]$ at the respective nodes.

3. **Dominance check:** Label $[1, 2]$ dominates label $[\infty, \infty]$ at node 2, label $[4, 3]$ dominates label $[\infty, \infty]$ at node 3, and label $[1, 4]$ dominates label $[\infty, \infty]$ at node 5.

4. **Reach:** Reaching out from node 2 to node 3 generates a label $[3, 4]$ at node 3.

5. **Dominance check:** Label $[4,5]$ is dominated by label $[3,4]$ since the second cost on the second label ($[3,4]$) is smaller than the second cost on the first label ($[4,5]$) and the second label consumes less resources (3 units) as compared to the first label (which consumes 4 units). Label $[4,5]$ is eliminated.

6. **Reach:** Reaching out from node 3 to node 4 generates a label $[4,6]$ at node 4.

7. **Dominance check:** Label $[4,6]$ dominates label $[\infty,\infty]$ at node 4.

8. **Reach:** Reaching out from node 5 to node 4 generates a label $[3,7]$ at node 4.

9. **Dominance check:** Both $[3,7]$ and $[4,6]$ are efficient at node 4 since $[3,7]$ uses less resources but $[4,6]$ is less expensive.

10. **Reach:** No more reaching can be done. STOP.

Two feasible efficient paths exist between nodes 1 and 4. Of the two labels at node 4, $[4,6]$ is less expensive and hence corresponds to the least cost path among all feasible paths.

As stated earlier, a common example of constrained shortest path is the shortest path problem with time windows (SPPTW). Solomon and Desrosiers (1988) discuss a method to solve the SPPTW. Desrochers and Soumis (1988) solve the SPPTW using a generalized permanent labelling algorithm. Their implementation uses generalized buckets, and extends the work of Denardo and Fox(1979). Desrochers and Soumis (1988b) extend the work of Desrochers and Soumis (1988) and present a reoptimization algorithm for the SPPTW. Their work is motivated by the fact that the SPPTW is solved repeatedly as a subproblem in many larger problems such as vehicle routing problems, pickup and delivery problems, travelling salesman problems, m-travelling salesmen problems, minimum spanning tree problems, multi-period vehicle routing problems and the shoreline problem. For example, Houck, et al.(1980) solve the travelling salesman problem as a constrained shortest path problem.

3.5 Large Scale Integer Programs and Column Generation

Large integer programs can be solved exactly by combining branch and bound and column generation. This section presents the technique, focusing on large scale $0-1$ minimization problems, such as the CPP. The method, however, is general and its steps are as follows (Barnhart, et al.,1993b):

- **Step 0:** Choose the root node of the branch and bound enumeration tree as the current node for exploration.
- **Step 1:** Using column generation, solve the LP associated with the current node. If the current node is the root node, then the optimal LP objective function value is a lower bound on the best integer solution value that can be found.
- **Step 2:** Step 1 has 4 possible outcomes - specifically,
  1. the LP at the current active node is infeasible;
  2. the optimal LP objective function value is higher than the best IP solution objective function value;
  3. the objective function value of the optimal LP solution is integral and better than that of the best IP solution so far; or
  4. the objective function of the optimal LP solution is better than the best IP solution objective function value but the optimal LP solution is not integral.

  As described in section 3.2, the first three outcomes cause the node to be pruned, i.e., further exploration is prevented at that node. In case of the fourth outcome, one of the non-integral variables, x_j^*, is selected and two new nodes are created from the current node, with the additional constraints $x_j \geq \lceil x_j^* \rceil$ and $x_j \leq \lfloor x_j^* \rfloor$ respectively.

- **Step 3:** If no more active nodes remain or if the gap between the best IP objective function value and the lower bound (LP solution at the root node) is "small" enough, the algorithm is terminated. Otherwise, an active node is chosen for exploration. Go to step 1.

An important point to be noted here is that it is not necessary to solve the LP's to optimality in step 1. By using termination criteria based on time, number of iterations or closeness of bounds, the LP's can be terminated to give solutions that are not optimal. The tradeoff, however, is that the tree may not be pruned as much and more nodes may have to be explored.

3.5.1 Challenges

The following points need to be stressed in order to bring out the inherent challenges of this IP solution procedure using column generation (Barnhart, et al.,1993b).

1. In this solution approach, the LP associated with *each* node in the branch and bound tree is solved using column generation. This is because an LP relaxation solved by column generation is not necessarily integral and applying a traditional branch-and-bound technique to the RMP with its existing columns will not guarantee an optimal, or even feasible solution. After branching, there may exist a variable with negative reduced cost that is not present in the RMP. Therefore, to find an optimal solution, it may be necessary to generate additional columns after branching.

2. If a non-integral optimal solution is generated at a node of the branch and bound tree, then a non-integral variable is chosen and new branches and corresponding active nodes are created. Since the LP associated with each new active node is also solved using column generation, *compatibility* of the branching decision and the subproblem solution procedure must be ensured. By compatibility, it is meant that the additional constraints introduced during branching must not destroy the tractability of the subproblem.

Consider, for example, the CPP. Each variable in the crew pairing problem represents a path. If its LP gives a non-integral solution, then a non-integral pairing is chosen and the node that is currently being explored is branched to give two new active nodes - one in which the pairing is set to 0 and the other in which it is set to 1. Since the LP associated with each new active node is also solved using column generation, the compatibility of this branching decision for CPSP must be examined. Consider, for example, the situation when the CPSP is solved using a time-line network and a shortest path procedure (as described in section 3.4.1). If a branching decision sets a pairing p to 1, then

(a) each pairing in RMP is deleted if it covers any flight in p,

(b) each row in RMP is deleted if it represents a flight in p,

(c) each arc in the time-line network is deleted if it contains any flight contained in p, and

(d) the RMP objective function is increased by the cost of P.

With these changes, every subsequent subproblem solution satisfies the requirement that the flights in p be covered only by pairing p. Now consider the alternate branch in which pairing p is set to zero. In this case, p is deleted from the RMP. It is not possible, however, to delete p from the time-line network. Instead, the subproblem solution procedure is constrained by the requirement that it not generate pairing p. For nodes in which the corresponding LP is constrained by the requirements that greater than one pairing has been set to zero, it is necessary to adjust the CPSP solution process to ensure that none of the disallowed pairings is generated.

The following section reviews the literature in solving large scale integer programs, particularly crew pairing problems. Section 3.5.2 discusses applications of column generation in crew pairing optimization and section 3.5.3 presents applications of combined branch and bound and column generation solution procedures.

3.5.2 Airline Crew Pairing Optimization

Column generation techniques that are used to solve crew pairing problems can be classified as either *explicit* or *implicit.* Explicit methods rely on brute-force enumeration of pairings. Since the total number of pairings is large, subsets of flights or duty periods are taken and pairings are enumerated using only these subsets. Implicit methods on the other hand, usually solve an optimization subproblem to generate minimum reduced cost pairings and hence "implicitly" price out **all** pairings.

Explicit Column Generation

An example of explicit column generation is the work of Rubin (1973), who solves the airline crew pairing problem as a large set covering problem. Set covering solutions are obtained for much smaller matrices extracted from the overall problem. A similar approach is taken by TRIP, a program written at American Airlines(Anbil, et al., 1991) , to determine *heuristically* a schedule for the domestic crew scheduling problem. Crainic and Rousseau (1987) also use an explicit column generation method to solve heuristically the crew scheduling problem. A simple chaining routine is used that puts duty periods together to form legal crew pairings, and then integer solutions are obtained with the use of Salkin's heuristic (Lemke, et al,1971).

Gershkoff (1989) models the American Airlines crew scheduling problem as an integer program. The heuristic solution approach starts from a feasible solution and at every iteration, a subset of the flights are taken and all possible pairings are generated from those flights. Then, the lowest cost set of pairings that cover all the flights are found. If the lowest cost set is the same in two successive iterations, then the algorithm is terminated.

Anbil, et al.(1993) solve the domestic crew pairing problem for American Airlines using an explicit column generation method. Anbil, et al.(1991b) describe a branch-and-bound procedure to determine solutions to the crew pairing problems encountered at American Airlines. Columns are generated using American Airlines crew-pairing optimization system TRIP and solutions are obtained using IBM's Optimization Subroutine Library (Druckerman, et al.,1991). Their branching strategy, based on the work of Ryan and Foster(1981), requires that a particular flight immediately follow another flight in a pairing. The solution procedure, however, does not actually involve branching since, based on the LP solution, a flight is selected to follow another flight and this decision is never altered.

Implicit Column Generation

Implicit column generation methods solve a subproblem which evaluates the reduced cost of all columns without explicitly enumerating all the columns. The subproblems are usually shortest path problems or dynamic programming problems.

Barnhart, et al.(1991,1993) solve the crew pairing problem for long haul carriers. The LP relaxation of the set covering formulation is solved using a column generation technique which repeatedly solves a constrained shortest path problem over a long-haul network. Various branching strategies are discussed, one of them being the strategy to branch on follow-ons as in Anbil, et al. (1991b) and another being a method of branching on day vs. night connects.

Desrosiers, et al. (1991) solve the crew pairing problem exactly using a branch-and-bound procedure in which columns are generated by solving resource constrained shortest path problems(Derochers and Soumis,1988).

Vance(1993) and Vance, et al.(1993b) solve the daily domestic crew pairing problem using an alternate two-phase formulation that partitions flights into duty periods and duty periods into pairings. The LP relaxation is solved using column generation, with two types of columns being generated, one corresponding to duty period sets and the other corresponding to pairings. An integer solution is obtained by embedding the column generation procedure within a branch-and-bound tree, where branching rules based on duty period sets are considered first and later branching is done on the pairing variables.

Minoux(1984) models the long haul crew pairing problem as an integer program and solves the LP relaxation using column generation. However, Minoux does not describe how integer solutions are obtained. Similarly, focussing on the LP solution, Rannou (1986) and Lavoie, et al.(1988) build upon this work with Lavoie, et al.(1988)

solving problems with 329 flights and 1113 duty periods.

Barutt and Hull(1990) solve the domestic problem for Northwest Airlines by formulating it as a set partitioning problem, and solving it using column generation. Methods of designing parallel algorithms are suggested.

3.5.3 Other Applications

The following sections describe applications of combined column generation and branch and bound solution approaches.

Time Window Constrained Routing

A survey of the use of column generation methods in time constrained routing and scheduling is given in Desrosiers, et al. (1993). Desrosiers, et al.(1984) develop algorithms to solve the school bus routing problem with time windows. The problem is formulated as a set covering problem and is solved using a branch and bound algorithm. Columns are generated using a shortest path procedure constrained by time windows on the nodes. Branching is performed on the arc flow variables and not on the routes so as to preserve the shortest path structure of the subproblem.

Haouari, et al. (1990) provide a general framework for modelling and solving complex routing problems. They model the vehicle routing problem as a set partitioning problem and solve it using a column generation approach embedded within a branch-and-bound algorithm. Desrosiers, et al.(1986) solve the problem of determining the number of vehicles to cover a certain set of trips and to determine their routes and schedules, so that each trip begins within its given time interval and that costs are minimized. This problem has been shown to be a generalisation of the m-travelling salesman problem. One of the algorithms used is a column generation algorithm on a set partitioning problem which is solved using branch and bound. Columns are generated by solving a dynamic programming subproblem.

Kolen, et al. (1987) solve a vehicle routing problem with time windows in which a fixed number of vehicles of given capactiy are available at a depot and have to serve a set of clients with given demands such that each client is visited within a time window. A branch and bound algorithm is designed to minimize the total route length. Shortest paths are computed by a labelling method, similar to Dijkstra's method (1959).

Desrochers, et al.(1992) present a new optimization algorithm for the vehicle routing problem with time windows. The problem is formulated as a set partitioning problem and is solved using a branch and bound procedure, generating columns using dynamic programming. A heuristic branching strategy is developed in which branching is performed on arc flow variables with the largest weight. Dumas, et al. (1991) solve the pickup and delivery problem with time windows using an exact algorithm. Column generation is used within a branch and bound scheme and columns are generated using a constrained shortest path procedure.

Urban Transit Crew Scheduling

The urban transit crew scheduling problem involves the determination of bus driver work schedules that minimize total costs and satisfy all labor agreements. This problem is described in detail by Ryan and Foster(1981) and Wren, et al.(1985). Desrochers and Soumis(1989) model this problem as a set covering problem and solve it using a branch and bound procedure. The LP relaxation is solved using column generation and new columns are generated by solving a shortest path problem with resource constraints using dynamic programming. The branching rule, is based on a method suggested by Ryan and Foster(1981), and requires that branching be made on a pair of tasks to be performed in a single workday.

Ship Scheduling

Appelgren(1969) solves a ship scheduling problem by formulating it as a multi-commodity flow model, relaxing integrality constraints, decomposing it using the principle of Dantzig-Wolfe(1960), and solving the LP using column generation. Fortunately, most of the optimal LP solutions were integral.

Communication Systems and Multi-Commodity Flow Problems

Ribeiro, et al.(1989) solve the problem of finding optimal schedules in satellite switching systems. A set partitioning formulation is used and column generation is used in the branch and bound tree search to ensure global integer optimality. A conventional branching rule is employed fixing a variable in the set partitioning formulation to either 0 or 1. If the variable is fixed at 1, then all rows with a coefficient of one in the corresponding column can be deleted and all other variables containing ones in these rows can be set to zero. Alternately, the column is fixed at 0 and is deleted.

Parker and Ryan(1993) solve the bandwidth problem of allocating bandwidth in a telecommunications network to maximize total revenue. The problem is formulated as a multi-commodity network flow problem with a requirement that the flows be integral. The problem is solved using a branch and bound procedure with linear programming providing the bounds. The linear relaxations are solved using column generation. Columns are generated using shortest path problems. A mixed branching strategy is used which fixes at one variables in the path based column generation formulation and fixes at zero the values of one of several variables in the arc-based formulation.

Barnhart, et al.(1991c) solve multi-commodity network flow problems in which a commodity is defined by a single origin and a single destination. The linear programming relaxation is solved using column generation and columns are generated by solving shortest path problems. The branching strategy involves branching on arc-flow variables. One branch requires that commodity k be assigned to arc ij while the other forbids commodity k from being assigned to arc ij. The second branch is easy to implement by just removing the arc from the subproblem network.

Cutting Stock Problems

Vance(1993) and Vance, et al.(1993b) present algorithms for the binary cutting stock problem employing both column generation and branch and bound. The Ryan and Foster(1981) branching heuristic is used to obtain optimal integer solutions.

Generalized Assignment

The generalized assignment problem is to find the maximum profit assignment of jobs to agents such that each job is assigned to precisely one agent and each agent has a capacity restriction. Savelsbergh(1993) formulates this problem as a set partitioning problem. Column generation and branch-and-bound are used to obtain optimal integer solutions. Branching is performed using a heuristic that a job is either forbidden or required to be assigned to an agent.

Chapter 4

Dual Ascent Heuristics

This chapter presents a dual ascent heuristic to speed up the performance of column generation. This is a general heuristic that can be applied to any column generation process. The need for the speed up is discussed and a dual ascent heuristic for column generation (DACG) is presented. An interior point modification of the heuristic is made and the modified heuristic (IDACG) is detailed. Computational experience on randomly generated problems is provided. This chapter also includes a literature review of various dual ascent heuristics. The next chapter shows how the heuristic can be incorporated into the column generation solution procedure for the CPP.

4.1 Motivation

Reduction in computation time for the column generation procedure is an important goal in algorithm refinement for the following reasons:

1. Despite being a very powerful tool, the column generation algorithm has been shown to possess very poor convergence properties. Very often, the algorithm makes rapid advances in the early iterations but then slows down and begins to "tail off" towards optimality (Bradley, et al., 1977).

2. The solution of CPP involves the solution of many LP relaxations of CPP. Specifically, one LP relaxation of the CPP is solved at each node of the branch

and bound enumeration tree. This implies that even small improvements in the performance of the column generation algorithm will lead to substantial savings in time.

3. A faster column generation algorithm allows more nodes in the branch and bound enumeration tree to be explored given a fixed solution time. Hence, a faster column generation procedure is more likely to achieve an optimal (or feasible) IP solution, given time limitations.

The poor performance of column generation could be due to a variety of reasons.

- While the column generation algorithm guarantees monotonic primal improvement, there is no such guarantee for the dual.
- The bouncing about of the dual implies the generation of many columns that may not be a part of the optimal solution. These additional columns, merely by their presence, slow down the column generation algorithm in the long run.

This leads to the hypothesis that if the dual could be controlled in some way, i.e., if an improved dual solution could be generated at each iteration, fewer columns and hence faster solution of the RMP would result. A dual ascent procedure can provide this control, and additionally, it can provide a computationally inexpensive way to compute a lower bound on the optimal RMP solution value (Hearn and Lawphongpanich,1989; Houck, et al., 1980).

4.2 Dual Ascent Heuristics - Literature Review

Subgradient optimization is one of the most popular and generic dual ascent techniques. It is a simple, approximately ascending algorithm for unconstrained or constrained non-differentiable concave programming problems. If certain assumptions are satisfied, the algorithm converges to an optimal solution. The algorithm and its convergence proofs have been covered in great detail in a number of works such as Held, et al.(1974) and Shapiro(1979). Subgradient optimization has been applied to a variety of problems. It has been used by Houck, et al.(1980) to solve the travelling salesman problem, by Shepardson and Marsten (1980) to solve the two duty period bus driver scheduling problem, by Marsten and Shepardson (1981) to solve airline crew scheduling problems and by Carraresi, et al. (1982) to solve large scale bus driver scheduling problems.

Fisher (1981) shows how subgradient optimization can be used in combination with lagrangian relaxation methods to solve integer programs. Sherali and Myers (1988) have conducted extensive tests on various dual formulations and subgradient optimization strategies for linear programming relaxations of mixed-integer programs. Hearn and Lawphongpanich (1989) have used the method to compare and test the performance of a lagrangian dual ascent heuristic.

Subgradient optimization however, has the following drawbacks (Marsten and Shepardson,1981; Sherali and Myers,1988;Hearn and Lawphongpanich,1989):

1. Subgradient optimization works very well for low requirements of accuracy. However, for high accuracy needs, it either fails or takes a large number of iterations to converge to optimality.

2. Subgradient optimization is still a very poorly understood algorithm. It is not very robust and is sensitive to the choice of its parameters like the step size and hence, may need to be fine tuned for each new class of problems.

3. Marsten and Shepardson(1981) report that for the set partitioning problems they solved, LP solutions were almost always integer, while solutions obtained using subgradient optimization were fractional.

Multiplier adjustment methods have also been used to obtain dual ascent. A well known example is the heuristic (DUALOC) for the facility location problem by Erlenkotter(1978). The heuristic is very simple and has been shown to be extremely successful. Wong (1984) has given a dual ascent approach that solves steiner tree problems on a directed graph and generalizes DUALOC. Multiplier adjustment methods have also been used by Guignard and Kim (1987) for lagrangean decomposition, by Guignard (1988) to achieve a dual ascent method for simple plant location problems in combination with Bender's cuts, by Guignard and Opaswongkarn (1990) to compute bounds in capacitated plant location problems, and by Guignard and Rosenwein (1989) to solve the generalized assignment problem. Barnhart (1992) uses dual ascent methods for large-scale multi-commodity flow problems and reports that the dual ascent solutions not only provide good lower bounds, but also provide good starting solutions for primal-based heuristics. Magnanti and Wong (1984) provide an overview and application of dual ascent and multiplier adjustment methods in network design problems.

4.3 The DACG Heuristic

The dual ascent heuristic for column generation (DACG) is based on LP duality theory. Any primal feasible vector has an objective function value greater than or equal to that of any dual feasible vector (in a minimization problem), a shift from a dual feasible vector in the direction of a primal feasible vector will ascend the dual objective function. In order to maintain feasibility of the dual vector, the shift from the dual vector to the primal vector should not be complete, since the primal feasible vector is not likely to be dual feasible. Thus, by performing a line search between the primal and dual solutions, a step size can be found that

- makes the largest ascent in the dual objective function, and
- maintains dual feasibility.

Incorporating these observations into the LP solution process for RMP, gives the DACG heuristic. In every iteration of column generation, after the generation of columns by SP, a line search is carried out between the dual feasible vector and the RMP dual vector (which is the primal feasible vector). This line search yields a step size and altering the dual in the direction of the step size provides

- an improved lower bound for the optimal value of MP, and
- a new dual feasible vector that will be used to generate columns (in addition to the ones generated by SP) and a new dual feasible vector that will be used in the

line search in the next iteration of the column generation solution procedure.

Consider the following optimization problem.

$$\min\ c.x \tag{4.1}$$

$$\mathit{subject\ to}\ A.x = e \tag{4.2}$$

$$x \geq 0 \tag{4.3}$$

where A is a matrix, e is a vector, x is the primal vector and π is the dual vector. The dual to this problem is given by

$$\max\ \pi.e \tag{4.4}$$

$$\mathit{subject\ to}\ \pi.A = c \tag{4.5}$$

$$\pi \geq 0 \tag{4.6}$$

The following notation will be used to describe the DACG heuristic.

π_k:	The dual vector optimal to RMP in iteration k.
π_0:	The initial dual feasible vector.
ρ_k:	an optimal dual vector, referred to as the dual iterate, to the RMP in iteration k.

In the case of the LP relaxation of the CPP, a dual vector is feasible if the reduced cost of **all** columns in the CPP have a non-negative reduced cost with respect to the dual vector. Using this definition, DACG performs the following steps at each iteration of the column generation algorithm:

- **Step 1** Find θ^* given by

$$\theta^* = arg \max_{0 \leq \theta \leq 1} feas[\pi_{k-1} + \theta(\rho_k - \pi_{k-1})] \tag{4.7}$$

where $feas[x]$ is a function whose value is x if x is dual feasible, and 0 otherwise.

- **Step 2** Set

$$\pi_k = \pi_{k-1} + \theta^*(\rho_k - \pi_{k-1}) \tag{4.8}$$

- **Step 3** If $\theta^* > 0$, find the minimum reduced cost column with dual variables π_k and add that column to the current LP.

Step 1 finds the maximum step size that can be taken to maintain dual feasibility while *ascending* the dual.

Proposition 1 *Whenever a line search is carried out (Step 1) and the dual feasible vector is moved i.e., $\theta^* > 0$ (Step 2), dual ascent is achieved.*

From equation 4.8 above,

$$\pi_k.e = \pi_{k-1}.e + \theta^*(\rho_k - \pi_{k-1}).e \tag{4.9}$$

Note that $\pi_k.e$ is the value of the objective function of the dual vector π_k (equation 4.4). Therefore the change in the dual objective function value Δz between iteration k and iteration $k-1$ is given by

$$\Delta z = (\pi_k - \pi_{k-1}).e = \theta^*(\rho_k - \pi_{k-1}).e \tag{4.10}$$

Since $\theta^* > 0$ and $\rho_k.e \geq \pi_{k-1}.e$ (from LP duality), we see that $\Delta z \geq 0$.

The maximum step size θ^* can be found by performing a line search between dual feasible vector π_k, and RMP dual iterate ρ_k and solving a SP for each θ in the line search, using the dual vector $\pi_k^\theta = \pi_{k-1} + \theta(\rho_k - \pi_{k-1})$. If the dual vector π_k^θ yields negative reduced costs columns in SP, π_k^θ is not dual feasible; otherwise π_k^θ is feasible. If $\theta^* > 0$, π_k (equation 4.8) is an improved dual solution. In **Step 2**, a new (improved) dual solution π_k is constructed and in **Step 3**, π_k^θ is used to generate columns to be added to RMP.

Proposition 2 *Optimality of MP is obtained if $\theta^* = 1$.*

If θ^* has a value of 1, the dual feasible vector π_k is equal to the RMP dual iterate ρ_k (from equation 4.8). So $\pi_k.e = \rho_k.e$ and the lower bound on the optimal MP solution value ($\pi_k.e$) equals the upper bound on the optimal MP solution value ($\rho_k.e$). The equality of the bounds indicates optimality.

Proposition 3 *DACG is finite and exact.*

The proof follows from the proof of convergence and optimality of the column generation algorithm, since DACG only modifies the column generation algorithm by generating *additional* columns.

4.3.1 Geometric Interpretation of DACG

Figure 4-1 provides a geometric interpretation of the DACG heuristic. DACG adds columns to the primal problem at each iteration. This corresponds to adding constraints to the dual problem and descreasing the size of the dual feasible region. If all the dual constraints (primal columns) could be enumerated, the feasible region in dual space would be specified completely.

Assume that the rectangle $0abc$ corresponds to the dual feasible region. Let the point 0 correspond to the initial feasible dual vector q_0. Let the restricted master problem in the first iteration correspond to the columns (or constraints in the dual) $c1-c1$ and $c2-c2$. Let $c3-c3$ and $c4-c4$ correspond to the columns (constraints in the dual) added in subsequent iterations. For the sake of simplicity, assume that only one column (constraint) is generated per iteration. Let p_i denote the DACG dual iterate in iteration k, and let q_i denote the feasible dual vector in iteration k. The first three steps are given below.

The following description uses the dual version of the restricted master problem. The columns in the primal problem correspond to constraints in the dual and hence primal column generation corresponds to dual constraint generation.

- **Iteration 1** The master problem is given by constraints $c1-c1$ and $c2-c2$ and the optimal dual iterate is p_1. A line search is performed between the feasible dual vector of the earlier iteration, q_0, and p_1. This line search will find the maximum step size that can be taken while keeping the feasible dual vector

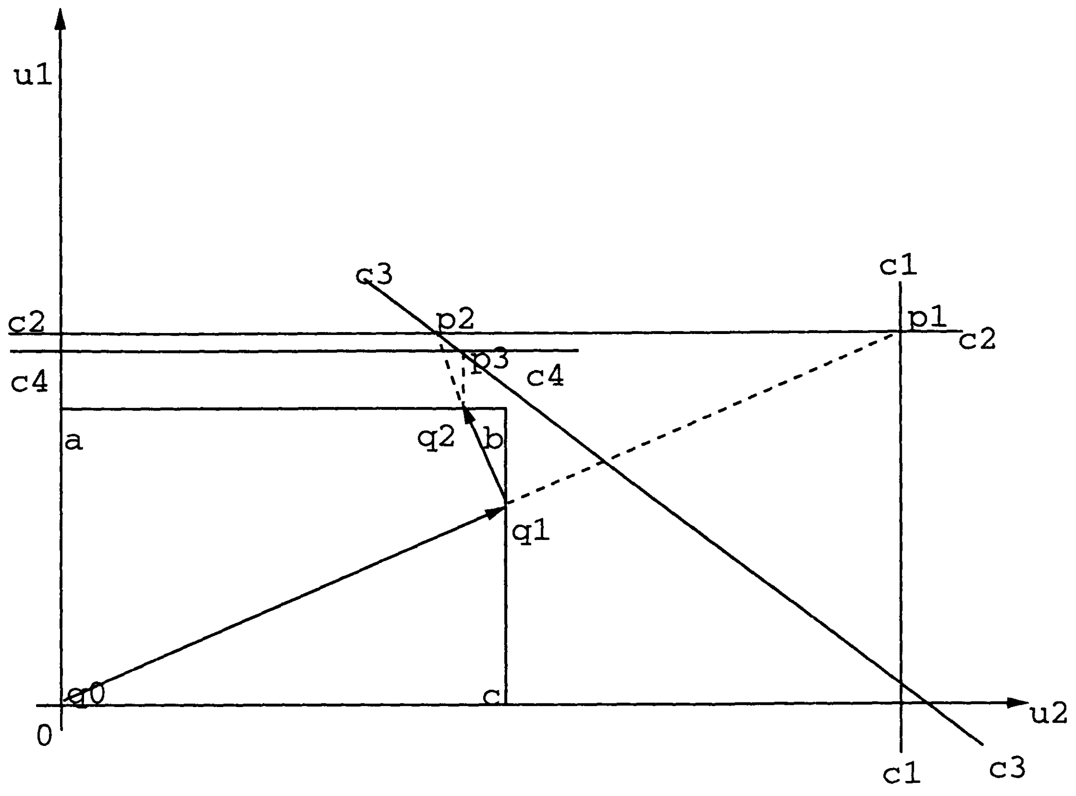


Figure 4-1: Geometric Interpretation of DACG

within the dual feasible region. DACG then moves to the new dual feasible vector, q_1.

- **Iteration 2** Constraint $c3 - c3$ is generated by DACG and the new optimal dual iterate is p_2. DACG performs a line search between q_1 and p_2 and moves to a new point q_2.

- **Iteration 3** DACG generates column $c4 - c4$ and the optimal dual iterate is p_3. DACG performs a line search which converges back to q_2, since the direction of the line search points out of the dual feasible region.

DACG has not been carried out to optimality for the sake of brevity.

4.4 IDACG - Interior Point DACG

Iteration 3 in the above example shows a situation in which the feasible dual vector gets "stuck" at the dual feasible region boundary, and is therefore unable to ascend the dual objective function. To overcome this, DACG is modified by changing step 2 as follows

- **Step 2:** Set

$$\pi_k = \pi_{k-1} + \alpha\, \theta^*(\rho_k - \pi_{k-1}) \tag{4.11}$$

where $0 < \alpha < 1$. Thus, the modification is simply that the maximum step size θ^* is multiplied by a step factor α to keep the dual feasible vector in the interior of the feasible region. The modified heuristic is denoted by IDACG (interior point version of DACG). This computationally simple modification is made based on the two following assumptions:

1. Initializing the dual feasible point to be an interior point, and multiplying the maximum step size by a step factor (< 1) will most likely keep the dual feasible vector in the interior.

2. Keeping the dual vector in the interior of the dual feasible region provides more flexibility in movement and many more line searches produce non-zero step sizes, resulting in faster overall convergence to the optimal dual solution.

Although the dual feasible region is not explicitly known, it is easy to find (by solving an appropriate subproblem) if a given dual solution is feasible. The lack of knowledge of the dual feasible region makes it difficult to find a feasible interior direction in a manner other than that employed in IDACG.

Using the example to describe DACG, Figure 4-2 depicts the behavior of IDACG. The first three steps of IDACG shown in figure 4-2 are as follows:

- **Iteration 1** The dual of the master problem corresponds to constraints $c1 - c1$ and $c2 - c2$ and the optimal dual iterate is p_1. A line search is performed between q_0 and p_1 and the maximum possible step size θ^* is determined. A new feasible dual vector, q_1, is constructed using equation 4.11.

- **Iteration 2** IDACG adds constraint $c3 - c3$ and the new optimal dual iterate is p_2. A line search is carried out between q_1 and p_2 and the maximum step size is determined. Once again, a dual vector, q_2, is constructed.

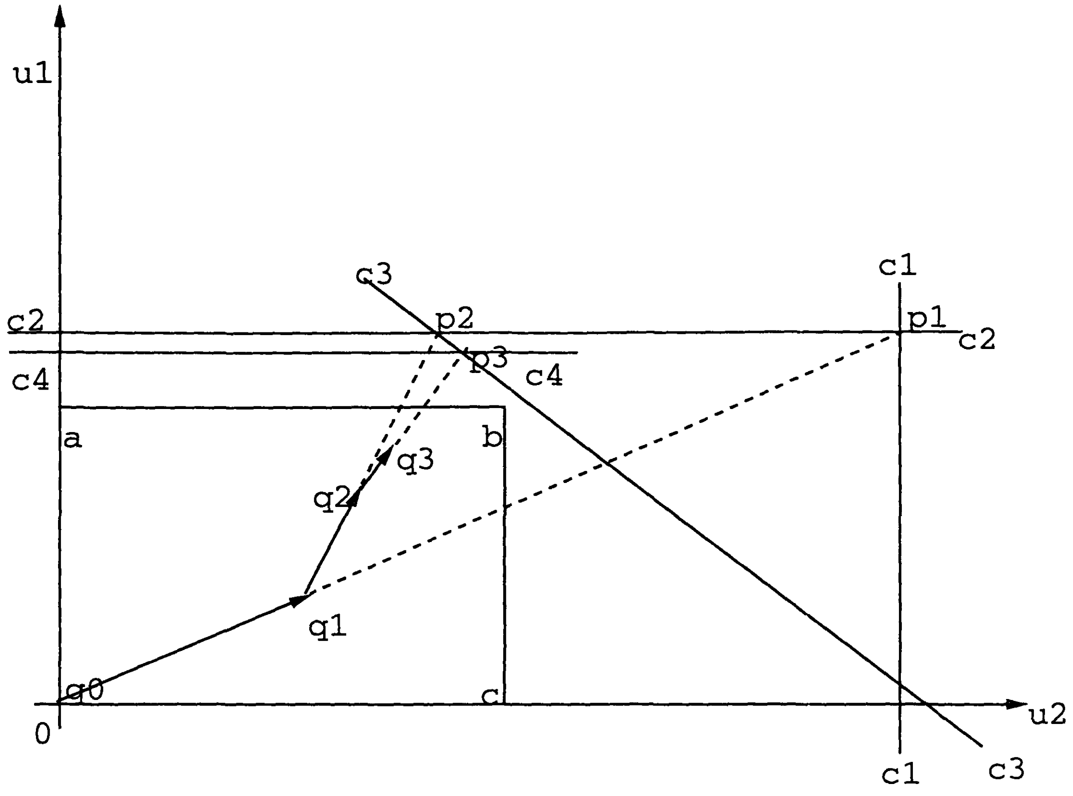


Figure 4-2: Geometric Interpretation of IDACG

- **Iteration 3** IDACG adds constraint $c4 - c4$ giving an optimal dual iterate p_3. The line search between q_2 and p_3 yields another feasible step size and a new feasible dual vector, q_3.

This example of IDACG shows how the dual vector is retained within the interior of the feasible dual region.

Proposition 4 *IDACG is exact and finite.*

This follows from proposition 3.

4.5 Results on Randomly Generated Problems

IDACG was tested on randomly generated problems. The randomly generated problems, constructed using a method proposed by Hearn and Lawphongpanich(1989), are of the form

$$\min\ cx \tag{4.12}$$

Size		*CG*	*IDACG*	*IMP.(%)*
25*x*50	*time*	−−	−−	−−
	ite.	55	28	50
	col.	55	55	*neg.*
	piv.	149	109	27
50*x*100	*time*	9	6	30
	ite.	124	54	57
	col.	124	106	14
	piv.	545	377	45
75*x*150	*time*	40	26	35
	ite.	233	91	61
	col.	233	181	22
	piv.	1618	920	43

Table 4.1: Overall Comparison of Heuristics - Density = 100%

$$subject\ to\ Ax \leq b \tag{4.13}$$

$$x \in X = \{x : 0 \leq x \leq d\} \tag{4.14}$$

where c,b and d are constant vectors and A is an $m \times n$ constraint matrix.

A comparison was made between ordinary (i.e., without dual ascent) column generation (CG) and IDACG on 30 random LP's, the specifications of which are given in table 4.1. The data were randomly generated from uniform integers with the following ranges: $c_j \in [-100, 100]$, $A_{ij} \in [-25, 25]$, $d_j \in [0, 50]$ and $b_i \in [0, 25n]$. The LP's were solved using IBM's Optimization Subroutine Library (Druckerman, et al., 1991). The results obtained are tabulated in table 4.1 in which "time" stands for the run time in seconds on an IBM RS 6000/320 workstation, "ite." refers to the number of iterations of the algorithm, "col." stands for the number of columns generated and "piv." stands for the total number of pivots made by the simplex subroutine in each algorithm. The last column shows the percentage improvement in runtime of IDACG over CG and is calculated using the formula

$$Percentage\ Improvement = 100 \times \frac{(Time\ by\ CG\ -\ Time\ by\ IDACG)}{Time\ by\ CG} \tag{4.15}$$

The results are summarized as:

1. IDACG was faster than CG on all the random problems tested. For the smallest problems (25 rows by 50 columns), the run times were negligible and hence not reported. For the medium problems (50 rows by 100 columns), the percentage improvement in time (given by equation 4.15) is 30%. For the largest problems (75 rows by 150 columns), the percentage improvement is 35%

2. For all problem sizes, IDACG required fewer iterations and fewer pivots in the simplex subroutine to reach optimality.

3. IDACG used fewer columns than CG to reach optimality for the medium and large problem sizes. However, the differences in the number of columns generated for the smallest problems were negligible.

4. IDACG always took fewer pivots in the simplex subroutine to reach optimality.

5. IDACG can potentially generate two columns per iteration in these problems - one based on the RMP dual iterate and one based on the dual feasible vector. However, a column based on the dual feasible vector is generated only if a non-zero step size is obtained in the dual line search. The results show that the number of columns generated is almost equal to twice the number of iterations for all the problem classes. This shows that the line search rarely results in a step size of zero and that the dual feasible vector remains in the interior as IDACG progresses.

Empirical experience indicates that the performance of IDACG is best for step factor values between 0.25 and 0.4. Thus, a step factor of 0.3 is chosen for the remaining computations.

Chapter 5

Solving the Crew Pairing Problem

This chapter presents

1. a column generation algorithm to solve the LP relaxation of CPP;
2. an application of IDACG to the CPP;
3. an efficient procedure to generate lower bounds on the optimal CPP objective function value; and
4. methods of obtaining integer solutions to the CPP.

5.1 LP Solution to the CPP

The first step in solving the CPP is to relax the integrality constraints and solve the LP associated with the root node of the branch-and-bound enumeration tree. The LP is very difficult to solve due to the enormity of the number of columns and hence column generation is used to solve it. The following column generation scheme, consisting of two parts that are executed alternately, can be used to solve the CPP (Minoux,1984;Lavoie, et al,1988; Barnhart, et al.,1991). An *iteration* of the column generation algorithm involves the solution of each of the following two parts:

1. **The restricted master problem** (RMP) which requires the optimal selection of pairings to cover flights, where the set of pairings is restricted to a subset of

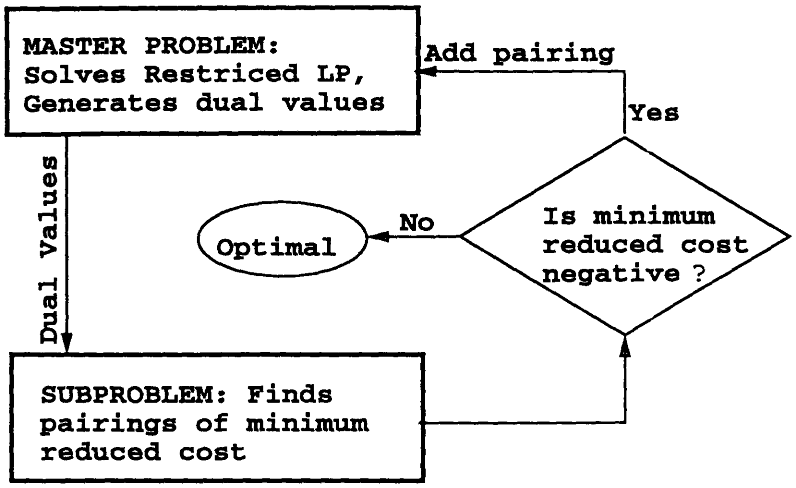

MASTER PROBLEM: Solves Restriced LP, Generates dual values
Add pairing
Yes
Dual Values
No
Optimal
Is minimum reduced cost negative ?
SUBPROBLEM: Finds pairings of minimum reduced cost

Figure 5-1: Column Generation for Crew Pairing Optimization

the total pairings, and integrality of the solution is not required.

2. **The subproblem** (CPSP) which generates one or more pairings with minimum reduced cost, i.e., pairings that can potentially reduce the cost of the current crew pairing solution generated by RMP. New pairings can be generated for some CPP problems using a shortest path procedure as discussed in section 3.4.1.

The column generation algorithm for the crew pairing problem is shown schematically in figure 5-1 (Minoux,1984;Lavoie, et al.,1988).

5.1.1 IDACG for the Crew Pairing Problem

The dual to the LP relaxation of the CPP is

$$(CPP-Dual) \quad \max \quad \sum_{i=1}^{m} \pi_i \tag{5.1}$$

subject to

$$c_j - \sum_{i=1}^{m} \pi_i a_{ij} \geq 0 \quad j = 1, \ldots, n \tag{5.2}$$

$$\pi_i \geq 0 \quad i = 1, \ldots, m \tag{5.3}$$

IDACG can be incorporated into the CPP solution process. Each iteration of IDACG involves performing the following steps:

- **Step 1. Initialization:** Choose an initial set of pairings, denoted by R, that cover all the flights. The starting dual vector π_0, that is feasible to CPP-Dual, is chosen to be a vector of zero's.

- **Step 2. Restricted Master Problem:** Solve RMP to optimality over R. Denote the optimal RMP dual solution for iteration k by ρ_k, referred to as the dual iterate at iteration k.

- **Step 3. Column Generation/Pricing Subproblem:** Solve CPSP and add to RMP any negative reduced cost pairings that are generated.

- **Step 4. Dual Ascent Heuristic:** Perform the following steps and then return to step 2.

    1. **Line Search:** Perform a line search between the dual feasible vector π_{k-1} and dual iterate ρ_k, to obtain the maximum feasible step size, θ^*. At each point θ in the line search, the feasibility of the point is checked. This is accomplished by defining vector π_k^θ as

    $$\pi_k^\theta = \pi_{k-1} + \theta(\rho_k - \pi_{k-1}) \tag{5.4}$$

    and modifying the time-line network costs to equal the reduced costs with respect to vector π_k^θ. Then, as in step 3, the existence of a negative reduced cost pairing can be determined using a (specialized) shortest path procedure. The existence of such pairings indicate dual infeasibility. If $\theta^* = 0$, the dual ascent step has failed in this iteration and the procedure returns to **Step 2.**

    2. **Dual Ascent:** The dual feasible vector is moved to a new point $\pi_k = \pi_{k-1} + \alpha\theta^*(\rho_k - \pi_{k-1})$, with $\alpha = 0.3$ (empirically set).

3. **Auxilliary Column Generation:** The network costs are modified to equal the reduced costs with respect to π_k. SP is solved again but only those pairings that have a negative reduced cost with respect to π_{k-1} are extracted and added to RMP.

5.1.2 IDACG for CPP - Efficiency and Other Issues

The following points can be used to improve the efficiency of IDACG.

1. Because the maximum step size θ^* is multiplied by a step factor α, it is not necessary to determine θ^* precisely. Consider the following proposition.

   Proposition 5 *By using a tolerance of 0.1 in the line search, the search can be restricted to 4 feasibility checks.*

   This follows from the fact that

   $$(\frac{1}{2})^4 = 0.0625 < 0.1 \tag{5.5}$$

   This results in a significant speed up of IDACG.

2. During the feasibility checks made in step 4a of IDACG, it is not necessary to determine the minimum reduced cost pairing between every pair of nodes representing the same crew base. If the shortest path algorithm for a node pair yields a negative reduced cost pairing, it indicates infeasibility and the feasibility check for this point in the line search can be stopped immediately.

3. Using equation (5.1), π_k can be used to construct a lower bound for MP. However, a better bound can be obtained by using vector δ^{bnd} given by

   $$\delta^{bnd} = \pi_{k-1} + \theta^*(\rho_k - \pi_{k-1}) \tag{5.6}$$

   δ^{bnd} is feasible to CPP-Dual, and hence $\sum_{i=1}^{m} \delta_i^{bnd}$ a lower bound on the optimal objective function value of MP (using equation 5.1).

5.1.3 Termination of LP solution using Farley's Bound

The branch and bound algorithm is basically an enumeration scheme that is enhanced by fathoming based on bound comparisons. The size of the branch and bound tree can be controlled by using strong bounds. However, there is a tradeoff between the computational efforts involved in computing strong bounds and evaluating small trees as compared to computing weaker bounds and evaluating larger trees (Barnhart, et al., 1993b). This trade off can be explored in branch and bound algorithms in which column generation is used to solve the resulting LP's, by terminating the column generation before LP optimality is achieved and working with non-optimal solutions. In some cases, this may not affect the branch and bound tree size. As an example, when the optimal CPP objective function value is known to be integral and the value of the objective function of the RMP is less than the round up of the lower bound provided by the root node RMP, column generation can be terminated since nothing can be gained by solving the RMP to optimality. This method has been used by Vance (1993) in solving binary cutting stock problems.

A lower bound on the optimal CPP objective function value can be obtained by determining the optimal value of the root node LP. However, this may be very painful to achieve since column generation often has a tailing effect and may take a very long time to achieve optimality. This motivates the need for a scheme to generate bounds that will permit the early termination of the column generation algorithm. One such bound, obtained by a method proposed by Farley(1990), is based on (non-optimal) LP solutions associated with the root node of the branch and bound tree.

Farley's Bounding Procedure

Since it is very difficult to achieve optimality for MP, researchers have used several rules of thumb to terminate the column generation algorithm. Gilmore and Gomory (1963) suggest that the column generation algorithm be terminated when the change in the objective function does not exceed a certain tolerance over a specified number of iterations. Farley(1990) points out however, that this rule has the disadvantage of stopping at what he terms a *stall point.* He suggests a better bounding rule defined on the following linear programming model and its dual.

Primal	P	**Dual**	D
Minimize	$Z = \mathbf{c}^\mathbf{T}\mathbf{x}$	Maximize	$U = \mathbf{b}^\mathbf{T}\mathbf{u}$
subject to	$\mathbf{Ax} \geq \mathbf{b}$	subject to	$\mathbf{A}^\mathbf{T}\mathbf{u} \leq \mathbf{c}$
	$\mathbf{x}, \mathbf{c} \geq \mathbf{0}$		$\mathbf{u}, \mathbf{c} \geq \mathbf{0}$

Using column generation, let $(\bar{x},\bar{u})$ be the current solutions to P and D, and define $\lambda_j = \mathbf{A}_\mathbf{j}^\mathbf{T}\bar{\mathbf{u}}$ for each column j. If $\lambda_j \leq 0$ for all j, then $\bar{u}$ is a feasible dual solution and $\bar{x}$ is optimal; Otherwise, let

$$\frac{c_k}{\lambda_k} = \min_{j:\lambda_j>0} \frac{c_j}{\lambda_j} \tag{5.7}$$

Note that $c_k/\lambda_k \geq 0$, given that $c_k \geq 0$.

Farley proves that for P, $\mathbf{c}^T\bar{\mathbf{x}}c_k/\lambda_k$ is a lower bound on the optimal objective function value Z. Various forms of cutting stock algorithms have been terminated using this procedure. The following section shows how this bounding procedure can be efficiently applied to the CPP.

Application of Farley's Bound to CPP

Note that the number of columns is exponential in the crew pairing problem and hence, finding c_k/λ_k requires an exponential number of evaluations. The determination of this ratio can be performed efficiently however, for long haul crew pairing problems in which pairing cost is assumed proportional to time-away-from-base. Hence, given any node pair (a, b) in a time-line network, the costs of all valid pairings between nodes a and b are equal. Let C_{ab} denote this cost, and let P_{ab} denote the set of all valid pairings between the nodes a and b. Then,

$$\begin{aligned} \min_{j:\lambda_j>0} \frac{c_j}{\lambda_j} &= \min_{(a,b)} \min_{j:\lambda_j>0, j\in P_{ab}} \frac{c_j}{\lambda_j} && (5.8) \\ &= \min_{(a,b)} \frac{C_{ab}}{\max\limits_{j:\lambda_j>0, j\in P_{ab}} \lambda_j} && (5.9) \end{aligned}$$

$$= \min_{(a,b)} \frac{C_{ab}}{\{-\min_{j:\lambda_j>0, j\in P_{ab}} -\lambda_j\}} \tag{5.10}$$

$$= \min_{(a,b)} \frac{C_{ab}}{\lambda^*_{ab}} \tag{5.11}$$

where $\lambda^*_{ab} = \{-\min_{j:\lambda_j>0, j\in P_{ab}} -\lambda_j\}$. Recall that the CPSP finds a pairing with minimum reduced cost (denoted by lab_{ab}), i.e.,

$$lab_{ab} = \min_{j \in P_{ab}} (c_j - \lambda_j) \tag{5.12}$$

Since,

$$\min_{j \in P_{ab}} (c_j - \lambda_j) = C_{ab} + \min_{j \in P_{ab}} (-\lambda_j) \tag{5.13}$$

it follows that

$$\lambda^*_{ab} = C_{ab} - lab_{ab} \tag{5.14}$$

Thus, the values λ^* are a by-product of the CPSP solution procedure. Hence, the above implementation of Farley's bounding procedure for long haul crew pairing problems is computationally no more complex than the CPSP solution procedure.

Generality of the Bounding Procedure

Consider now the effect on the generation of Farley's lower bound for CPP's where pairing cost is not proportional to the time-away-from-base cost. For example, as described in section 2.3.1, Vance(1993) studies a typical U.S. domestic crew pairing problem in which crew pairing cost is a non-linear function of several different costs. Even for these complex cost structures, a lower bound on pairing cost is provided by the time-away-from-base component, i.e.,

$$c_j \geq C_{ab} \, \forall j \in P_{ab} \tag{5.15}$$

and hence, Farley's lower bound is finally computed as:

$$\min_{j:j>0} \frac{c_j}{\lambda_j} = \min_{(a,b)} \min_{j:j>0, j\in P_{ab}} \frac{c_j}{\lambda_j} \tag{5.16}$$

$$\geq \min_{(a,b)} \frac{\min_{j \in P_{ab}} c_j}{\max_{j:j>0, j \in P_{ab}} \lambda_j} \tag{5.17}$$

$$= \min_{(a,b)} \frac{C_{ab}}{\{-\min_{j:j>0, j \in P_{ab}} -\lambda_j\}} \tag{5.18}$$

$$= \min_{(a,b)} \frac{C_{ab}}{\lambda^*_{ab}} \tag{5.19}$$

5.1.4 LP for CPP - Implementation Issues

Several implementation issues must be considered in solving CPP. These are discussed in the following sections.

Deadheading and Coverage

Crew deadheading refers to repositioning of crews to increase their utilization (Barnhart, et al., 1991). Sometimes it is essential to add deadheads to ensure coverage of flights. For example, in dated problems (ones in which the times and dates of flight departures are given), early (late) flights in the schedule which do not start (end) at a crew base do not get covered. This is because there are no earlier (later) flights in the schedule that can get them to or from a crew base. Deadhead flights are therefore added so as to allow crews to depart (return) from (to) their crewbases and fly the first (last) flights in a schedule. These deadheads can be selected by hand or by using an analytical tool such as that of Barnhart, et al(1991b) .

Starting the Algorithm

The big-M method is used to obtain the initial solution for the column generation algorithm. In the big-M method, a starting solution consisting of high cost artificial columns is used. Each column represents an artificial pairing that covers exactly one flight and has a cost equal to the sum of the cost of the flight, the cost of deadheading a crew from its crewbase to the flight's origin, and the cost of deadheading the crew from the flight's destination to the crewbase. There are as many such columns as there are flights. Since these columns are expensive and cover only one flight each, they are unattractive and will be driven out of the basis if a feasible solution exists.

Column Management

Column management refers to the rules of thumb used to restrict the number of RMP columns to a reasonable number. It may be advantageous to delete RMP columns since, an increase in the number of columns increases not just the number of pivots, but also the time taken for each pivot. It is important to note that deleting columns will not affect the optimality of the column generation procedure. Two sample column management techniques are:

1. After each RMP solution, all columns that have reduced costs greater than some small positive threshold value are discarded.

2. After each RMP solution, all non-basic columns are discarded.

Managing round off errors

In order to handle round off errors, only pairings with reduced costs less than some small (negative) threshold are added to RMP.

The Overall Algorithm

Incorporating the above implementation issues, Figure 5-2 gives a schematic sketch of the solution method for the linear relaxation of the CPP.

5.2 IP Solution to CPP

One method of solving big integer programs like CPP is with the use of column generation embedded within a branch-and-bound framework, as described in section 3.5. However, this approach has several practical difficulties, such as:

1. Branch and bound is a computationally difficult and memory intensive approach even without the added requirements of a sophisticated solution technique like column generation.

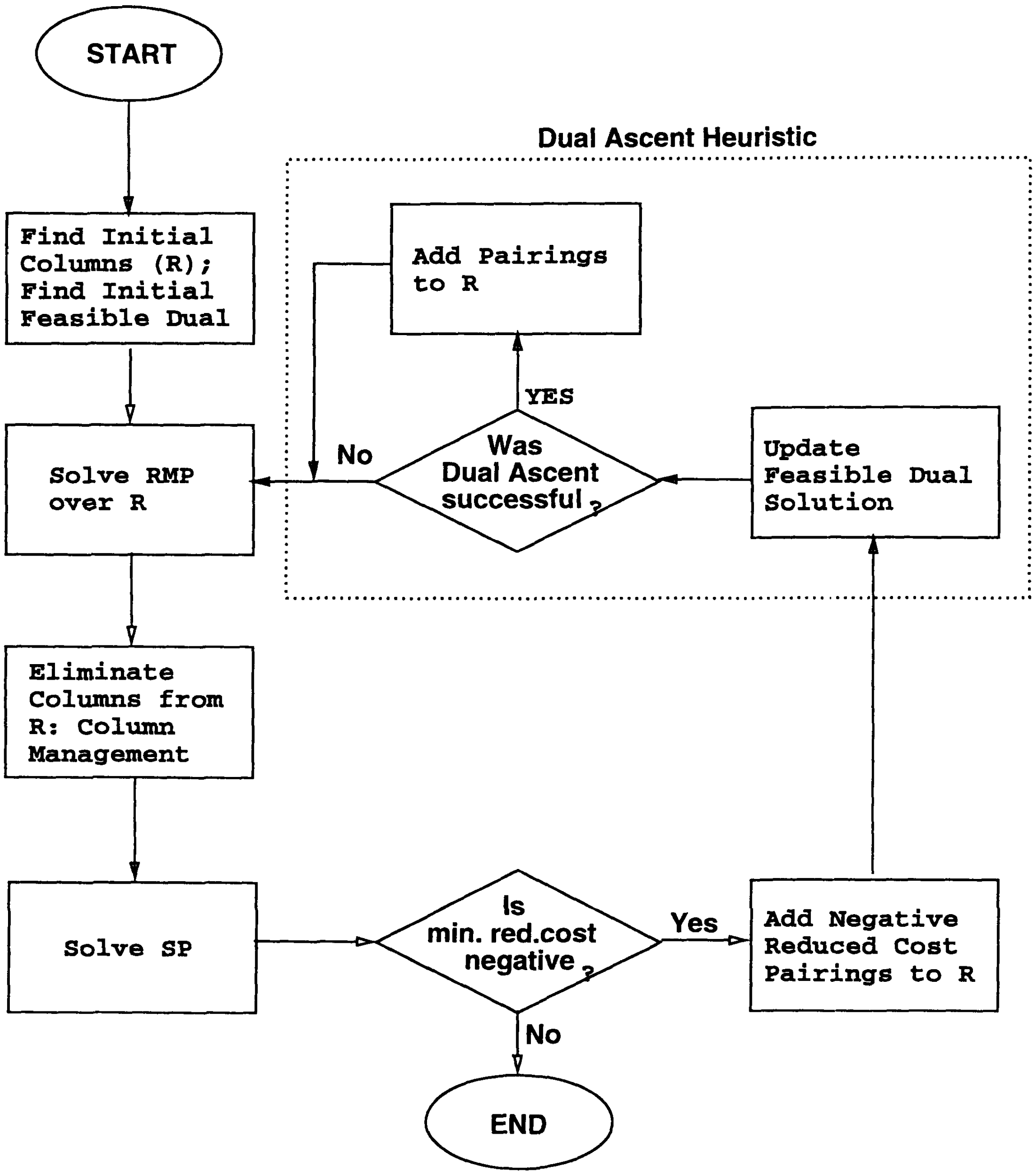

START
Dual Ascent Heuristic
Find Initial Columns (R); Find Initial Feasible Dual
Add Pairings to R
YES
Solve RMP over R
No
Was Dual Ascent successful ?
Update Feasible Dual Solution
Eliminate Columns from R: Column Management
Solve SP
Is min. red.cost negative ?
Yes
Add Negative Reduced Cost Pairings to R
No
END

Figure 5-2: LP Solution Procedure for CPP

2. Column generation complicates the approach further since the solution of *each* node involves solving an LP with additional constraints using column generation.

A simplified solution approach, one which is not guaranteed to produce an optimal CPP solution, is to solve only the root node LP with column generation and to solve all subsequent LP's with a fixed set of columns. This set may be (a subset of) the columns generated in solving the root node. Practical experiences of Anbil, et al. (1991b) and Barnhart, et al. (1993) show that this simplified method yields near optimal solutions for CPP's. Two strategies to solve are discussed below. One uses the strategy provided by OSL directly and is referred to as the black box strategy; the other uses a specialized branching rule tailored after that of Ryan and Foster (1981) and is referred to as the follow-on fix strategy.

5.2.1 The Black Box Strategy

This method uses the OSL subroutines EKKMPRE and EKKIMDL. EKKMPRE is a pre-processing subroutine that analyzes the 0-1 structure of the constraint matrix in order to reduce the size of the branch and bound enumeration tree. Some of the steps carried out by EKKMPRE are as follows.

1. A heuristic approach is used to set all 0-1 variables to 0 or 1 and hence determine if a valid solution can be obtained.

2. Fix each 0-1 variable in turn, first to 0 and then to 1, and determine the effect on all other variables. This analysis helps in pruning the branch and bound tree.

3. Add constraint rows called *cuts* to make the solution to the LP closer to the IP solution.

The branching rule used by EKKIMDL sets a variable to 0 or 1. The variable is chosen from among those with the highest priority where the variable priorities are provided by the user (all the priorities are set equal to 1 in this study). Among those variables with equal priority, the variable with the least *degradation* is chosen. The degradation of a variable is a measure of how much the objective function will worsen before an integer solution is reached if the variable is used in branching. More details of the above subroutines and related terms can be obtained from the OSL User's Guide.

5.2.2 Follow-on Fix Strategy

This branching strategy is based on the branching rule developed by Ryan and Foster(1981) and applied by Anbil, et al. (1991b) to CPP's. If the optimal LP solution at the current active node is non-integral, one of two branching rules is chosen - flight a should be followed by flight b, or flight a should not be followed by flight b. The choice of flights a and b can be made by computing from the basic RMP solution for the current branch and bound node, which pair of flights appear sequentially most often. In the method adopted by Anbil, et al. (1991b), a branching decision that is made is never changed (i.e., backtracking is not done). However, good solutions to the CPP were obtained.

5.2.3 IP - Efficiency Issues

Branch and bound algorithms consume a lot of memory. Hence, it is desirable to reduce the problem size whenever possible. The heuristic used to achieve this is:

- **Step 1 Elimination of Columns:** Discard all columns in the root node LP which have reduced costs greater than some tolerance.
- **Step 2 Elimination of Rows:** Each column in the root node LP solution with a value of 1 is discarded along with each row it covers.

After the root MP is either solved to optimality or terminated before optimality is reached, the heuristic is run on the constraint matrix to reduce its size to manageable proportions. This procedure will not affect the feasibility of the CPP solution.

Chapter 6

A Case Study

6.1 Introduction

This chapter presents computational experience in solving crew pairing problems of a typical long haul carrier. The model was coded using the IBM's OSL and the C programming language. All computational tests were run on an RS 6000/370 workstation. The following definitions will be used in the analysis of the results.

1. **Bound Gap** refers to the gap between the upper and lower bounds on the optimal MP objective function value, during the solution of RMP using column generation, and is defined as

$$\textit{Bound Gap} = \frac{(\textit{Current RMP Solution} - \textit{Current Lower Bound})}{(\textit{Current RMP Solution} + \textit{Current Lower Bound})/2} \tag{6.1}$$

2. **Optimality Gap** is used to measure the difference between the optimal solution of the current RMP and the optimal MP solution, and is defined as

$$\textit{Optimality Gap} = \frac{(\textit{Optimal RMP Solution} - \textit{Optimal MP Solution})}{\textit{Optimal MP Solution}} \tag{6.2}$$

3. **Duality Gap** is used to measure the gap between the best CPP (IP) solution and the optimal MP solution, and is defined as

$$Duality\ Gap\ (\%) = 100 \times \frac{(Best\ CPP\ Solution\ -\ Optimal\ MP\ Solution)}{Optimal\ MP\ Solution} \tag{6.3}$$

4. **Line Search Failures** refers to the number of line searches that end with a maximum feasible step size (θ^*) of zero.

5. **Column Generation Failures** refers to the number of column generation iterations which do not produce any negative reduced cost columns using reduced costs computed with the current dual feasible solution. This includes those iterations in which the line search has failed.

6. **Average Step Size** refers to the average of the dual ascent feasibility step sizes. Therefore, if θ_i^* is the maximum step size in iteration i, the average step size over N iterations is given by

$$\frac{\sum_{i=1}^{N} \theta_i^*}{N} \tag{6.4}$$

7. **Iterations to Optimality** refers to the number of iterations that the column generation procedure takes to reach primal optimality.

8. **Percent Integrality** refers to the percentage of the non-zero variables in an optimal LP solution for CPP that are equal to one.

6.2 Data used in the Case Study

The following definitions are used to characterize the datasets.

1. **No_Seg** refers to the number of flights that should be covered.

2. **No_Dhd** refers to the number of deadheads used in solving the problem.

3. **No_Dps** refers to the total number of duty periods generated from the flights and deadheads.

4. **No_Arcs** refers to the number of arcs in the time-line network (section 3.4.1).

5. **No_Nds** refers to the number of nodes in the time-line network.

6. **No_Cbs** refers to the number of crew bases in each dataset.

7. **No_Cb_Nds** refers to the number of crew base nodes in the time-line network.

8. **No_Fl_Pr** refers to the average number of flights per pairing.

Five full datasets, numbered P1 through P5, were used in the computational study. Each dataset represents a six to seven week schedule of flights to be flown by a typical long haul carrier. Three smaller datasets, numbered S1, S2, and S3, were created by extracting small portions from the full dataset P1. These smaller datasets were created to test the effect of problem size on algorithmic performance. S1 and S2 represent a flight schedule of one week each, while S3 represents a two week flight schedule. The problems were solved using the integer programming and column generation techniques described in chapters 3 and 5. The column generation subproblem is a simple shortest path problem and it is implemented using a time line network as described in section 3.4.1. The characteristics of all the problems and the corresponding subproblems are given in table 6.1.

6.3 Solution of Small Problems

The performance of problems S1, S2 and S3 using conventional column generation (CG) is tabulated in table 6.2, i.e., no special dual ascent or column management strategies were tested on these problems. Similar performance of CG is achieved for problems S1 and S2. For example, table 6.2 compares the number of iterations, the amount of time, and the number of flights per pairing for S1, S2, and S3. Observe that both S1 and S2 have integral LP optimal solutions.

Problem	Problem Characteristics				Subproblem Characteristics		
No.	No_Seg	No_Dhd	No_Dps	No_Cbs	No_Nds	No_Arcs	No_Cb_Nds
P1	1138	2368	4551	2	9102	13616	2318
P2	1128	1025	3221	2	6442	9629	1078
P3	943	724	2478	2	4956	7402	898
P4	1012	1628	4226	2	8452	12645	2194
P5	1023	1862	4703	1	9406	14074	2884
S1	163	1342	2007	2	4014	5988	1032
S2	160	1570	2318	2	4636	6917	1211
S3	325	1767	2799	2	5598	8360	1442

Table 6.1: Case Study Data Characteristics

Problem	No_Fl_Pr	Performance Criterion	Percentage Integrality	Duality Gap
S1	4.47	20 ite. 349 sec.	100%	0
S2	4.43	17 ite. 358 sec.	100%	0
S3	4.20	25 ite. 976 sec.	48%	0.08%

Table 6.2: Performance of Small Problems

The optimal solution to problem S3 has only 48% integrality. It is solved to IP optimality by using the constraint matrix of the optimal MP solution as input to the "black box" IP solver in IBM-OSL (section 5.2.1). A feasible integer solution with a duality gap (equation 6.3) of 0.08% was achieved in 426 seconds.

6.4 LP Solution of Large Problems

The following solution strategies have been used to find optimal LP solutions for problems P1, P2, P3, P4, and P5.

6.4.1 Solution Strategies

In addition to conventional CG, three dual ascent variants of IDACG were used to solve problems P1 through P5. These were used in combination with two column management schemes.

Column Management Strategies

The three column management strategies that were tested are:

1. After each iteration of CG (or IDACG), all columns are retained. However, each time the number of columns exceeds a threshold (chosen as 50,000), only the basic columns are retained and all other columns are eliminated. This strategy is denoted by CMS1.

2. All columns are retained after each iteration of CG (or IDACG). However, each time the number of columns exceeds a threshold (chosen as 35,000), all columns with reduced costs greater than a specified value (chosen to be a quarter of the maximum pairing cost) are eliminated. This strategy is denoted by CMS2.

3. All columns are retained after each iteration of CG (or IDACG). However, each time the number of columns exceeds a threshold (chosen as 35,000), all columns with reduced costs above the median reduced cost are eliminated. This strategy is denoted by CMS3. The median reduced cost is calculated approximately by first determining the maximum and minimum reduced costs among all columns in the constraint matrix and then using an interval reduction method with a large tolerance.

Dual Ascent Variants

Conventional column generation, that is column generation not using dual ascent, is denoted as DAS0. The three dual ascent variants of IDACG tested are

1. Columns are generated (in step 3 of IDACG) using the interior point dual feasible vector given by equation 4.11. This strategy is denoted DAS1.

2. Columns are generated using the boundary point dual feasible vector, instead of the interior dual feasible vector. This strategy is denoted DAS2.

DAS1 and DAS2 carry out four evaluations for every line search in step 1 of IDACG. Since each evaluation in the line search is time consuming, an attempt is made to reduce the time taken by the line search by reducing the number of evaluations to three. This strategy is denoted by DAS3.

A comparison of the performance of dual ascent heuristics DAS2 and DAS3, using column management strategy CMS1, is given in table 6.3. The performance of DAS3 is, in general, worse than the performance of DAS2. The reason for the poor performance of DAS3 is the fact that, compared to DAS2, a greater percentage of the line searches end in failure, i.e. yield a maximum step size of zero. Since DAS3 performs 3 evaluations in every line search as compared to 4 evaluations per line search by DAS2, the accuracy of the line search in DAS3 is poorer and hence, small step sizes will not be captured by DAS3, resulting in many more line search failures by DAS3. A zero step size (i.e., a failed line search) means that the dual cannot be ascended and no dual ascent columns can be generated and hence, a line search is wasted. Failed line searches slow down the algorithm. Owing to its poor performance, DAS3 has not been investigated further.

6.4.2 Solving to LP optimality

Table 6.4 compares the performance of DAS1 and DAS2 for the five datasets, P1 through P5.

"I." refers to the number of iterations,

"T." refers to the time taken in seconds, and

"B." refers to the bound gap (equation 6.1).

If an inordinate amount of time was required to achieve an optimal LP solution, the procedure was terminated prematurely. The bound gap is reported for such instances. Only 13 of 15 problem/column management strategy combinations were considered because problems P1 and P4 did not converge for column management strategy CMS3. This yields 13 conventional CG cases and 26 IDACG cases.

Problem No.	Performance Criterion	DAS2	DAS3
P1	Iterations	44	41
	Time (seconds)	17129	17854
	Fail. Line Searches(%)	4.6	12.2
P2	Iterations	35	40
	Time (seconds)	7316	7995
	Fail. Line Searches(%)	11.4	22.5
P3	Iterations	22	27
	Time (seconds)	2393	2697
	Fail. Line Searches(%)	4.6	11.1
P4	Iterations	41	47
	Time (seconds)	14213	15403
	Fail. Line Searches(%)	9.8	21.3
P5	Iterations	25	30
	Time (seconds)	6472	6942
	Fail. Line Searches(%)	8.0	16.7

Table 6.3: Performance of DAS3

1. **Comparison of CG and IDACG:** In 16 of the 26 cases, IDACG took fewer iterations than CG. However, only 8 cases of IDACG proved to be faster in terms of time than CG. This is because even though IDACG requires fewer iterations, each IDACG iteration takes longer due to the dual ascent step.

2. **Comparison of DAS1 and DAS2:** Neither DAS1 nor DAS2 outperforms the other, in terms of time or number of iterations.

3. **Comparison of Column Management Strategies:** A comparison of column management strategies shows that CMS3 is clearly the worst since 5 problems had to be terminated due to poor convergence when it was used. In general, CMS1 seems to perform better than CMS2.

4. **Effect of Subproblem Complexity:** IDACG solves the subproblem many times - it solves a subproblem to generate columns as in CG; it then carries out a line search in which it runs the subproblem 4 times (in DAS1 and DAS2); and provided the maximum step size θ^* is non-zero, it runs the subproblem

once more to generate additional columns. This implies that if the subproblem is computationally expensive to solve, the performance of IDACG will be poor when compared to that of CG. Table 6.5 shows that the proportion of time spent by the subproblem is the largest in P5. This is one of the reasons for the poor performance of IDACG on problem P5.

Table 6.1 shows that the P5 subproblem has the largest number of nodes, the largest number of arcs and the largest number of crew base nodes. Moreover, P5 has only one crew base, implying greater subproblem solution time. This can be explained with an example.

Example 7 *Consider two subproblems A and B, both of which have 200 crew base nodes. Suppose that subproblem A has only one crew base while subproblem B has two crew bases, named JFK and BOS respectively, each with 100 nodes. Suppose also that every node pair belonging to the same crew base can be connected by one valid pairing, going from the node with earlier time to the node with later time.*

Although the shortest path algorithm used is a one-to-many algorithm (i.e., from a single origin to multiple destinations), the number of shortest path pairings that one can extract from a set of n nodes is given by the formula

$$\frac{n \times (n-1)}{2} \tag{6.5}$$

From equation 6.5, subproblem A has 19900 node pairs and subproblem B has 4950 node pairs for each crew base and hence a total of 9900 node pairs. This implies that subproblem A would possibly have twice as many pairings as subproblem B and hence would possibly take twice as long.

This may explain why the P5 subproblem is the most difficult to solve computationally.

5. **Better Bounds by IDACG:** IDACG produces better bounds than CG. This is evident from the problems which are terminated before achieving an optimal

LP solution. The bound gaps are smaller for the problems when IDACG is used.

6. **Early Primal Optimality:** In solving problems P1 through P5, the optimal primal objective function value is reached relatively early in the solution process, but a feasible (optimal) dual solution is not achieved for several additional iterations. This results because the right hand side in the primal problem is a vector of 1's which causes degeneracy in the dual, i.e., it may take the dual a substantial number of iterations to stabilize and reach optimality. Table 6.6 gives the percentage of the number of iterations in which the optimal primal objective function value is attained. As an example, consider the solution of P4 using DAS0 and CMS2. An optimal primal solution value is achieved in just 51.4% of the iterations needed to prove LP optimality. This justifies the strategy of terminating the solution procedure when the primal-dual gap (bound gap) is small. Although the lower bound will not be as tight, the run time will be reduced.

6.4.3 Solving with Termination Criterion

The criteria that are used to terminate the column generation algorithm is based on the bound gap (equation 6.1). The results for two values of the bound gap - 0.05 and 0.01, are given in tables 6.7 and 6.8 respectively. The two termination criteria will be referred to as TC1 and TC2 respectively. Some entries in the two tables are empty since they were terminated before reaching the termination criterion. In tables 6.7 and 6.8,

"I." refers to the number of iterations,

"T." refers to the time in seconds,

"O." refers to the optimality gap at termination,

"L." refers to the fraction of the failed line searches,

"C." is the number of failed dual ascent column generation iterations,

"S." refers to the average step sizes upto termination.

No.		CMS1			CMS2			CMS3		
		DAS0	DAS1	DAS2	DAS0	DAS1	DAS2	DAS0	DAS1	DAS2
P1	I.	48	51	44	48	49	58	313	163	134
	T.	18479	19237	17129	20569	24732	24475	52566	53036	45114
	B.							0.204		0.009
P2	I.	51	36	35	100	54	165	72	59	39
	T.	12040	12539	7316	15524	12401	19145	12201	11589	7086
	B.									
P3	I.	32	24	22	33	35	38	44	26	27
	T.	2122	2095	2393	2138	2822	3653	2436	2075	2412
	B.									
P4	I.	26	38	41	37	47	27	369	383	225
	T.	7778	13090	14213	14575	24356	19745	57450	74106	38556
	B.							0.138	0.004	0.007
P5	I.	38	39	25	31	23	24	29	24	27
	T.	4518	8269	6472	3673	5826	6391	3152	5314	6403
	B.									

Table 6.4: Overall LP results

Problem	% time spent on	CMS1	CSM2	CSM3
P1	Master problem	93.27	94.16	88.47
	Shortest paths	5.68	4.90	11.55
P2	Master problem	96.12	95.30	95.29
	Shortest paths	3.07	4.00	3.89
P3	Master problem	90.10	90.01	89.61
	Shortest paths	7.02	7.06	7.88
P4	Master problem	82.87	89.59	74.27
	Shortest paths	14.90	9.22	25.72
P5	Master problem	44.25	48.79	39.47
	Shortest paths	51.15	45.69	54.09

Table 6.5: Analysis of Time Consumption in Conventional CG

No.	CMS1			CMS2			CMS3		
	DAS0	DAS1	DAS2	DAS0	DAS1	DAS2	DAS0	DAS1	DAS2
P1	0.646	0.4901	0.682	1	0.469	0.466		0.411	0.388
P2	0.941	0.556	0.686	0.58	0.519	0.176	0.986	0.441	0.795
P3	0.938	0.792	0.864	0.909	0.543	0.526	0.682	0.731	0.741
P4	0.923	0.5	0.596	0.514	0.277	0.593		0.068	0.218
P5	0.79	0.87	1	0.839	0.913	0.833	0.931	0.958	0.889

Table 6.6: Proportion of Iterations to Reach Primal Optimality

Termination Criterion 1 (TC1)

The following results are observed for TC1 (refer table 6.7).

1. **Comparison of IDACG and CG:** IDACG (DAS1 and DAS2) always took fewer (or equal) iterations than CG (DAS0). The performance of DAS1 was generally better than DAS0 in terms of time. The same however, cannot be said of DAS2. This shows that DAS1 is generally the best strategy. One of the reasons for this could be that the average step size is generally larger in DAS1. A greater step size in the IDACG dual ascent step implies that the dual is kept further in the interior and this allows a greater flexibility in the movement of the feasible dual and hence better performance.

2. **Comparison of Bounds:** The reason for the better performance of dual ascent is the better quality of the bounds generated by IDACG. The relatively tighter bounds generated by IDACG allow faster termination of the algorithm and hence savings in both the time and number of iterations. A point to be noted is that IDACG (DAS1 and DAS2) always satisfy the termination criterion while CG (DAS0) fails to do so in some cases.

3. **Small Optimality Gaps:** Despite using a large bound gap of 5% (0.05), the primal objective function value is quite close to the optimal LP solution value. This can be seen from the optimality gaps (equation 6.2 reported in table 6.7).

4. **Column Management Strategies:** A comparison of the column management strategies shows that CMS3 is the least desirable since two problems do not reach the termination criterion TC1. In general, CMS1 performs better than CMS2.

5. **Interior Point Performance:** The number of line search failures and dual ascent column generation failures is identical in almost all cases for DAS1 and DAS2. This shows that both the methods keep the dual in the interior.

It is interesting to see that P5 defies any trend. For example, dual ascent in general improves the performace of CG except for P5 where CG is far superior to IDACG.

Termination Criterion 2(TC2)

The following results are observed in table 6.8.

1. **Comparison of IDACG and CG:** Although the performance of DAS1 and DAS2 is generally better than DAS0 in terms of number of iterations, their performance is poor relative to CG, in terms of time. One of the reasons could be the fact that a bound gap of 0.01 is relatively small and due to the tailing effect of column generation, more iterations are taken to reach that gap. Since IDACG takes more time than CG per iteration, it will perform badly when the number of iterations are increased.

2. **Comparison of DAS1 and DAS2:** The performance of DAS1 is generally better than that of DAS2. One reason could be the fact that the average step size is generally larger in DAS1 than in DAS2. Moreover, the number of line search failures and dual ascent column generation failures is usually larger in DAS2, contributing to its poor performance.

3. **Optimality Gaps at termination:** The optimality gap at termination using TC2 is always very small. Table 6.8 shows that for a bound gap of 0.01, the problems are very close to the optimal LP solution.

No.		CMS1			CMS2			CMS3		
		DAS0	DAS1	DAS2	DAS0	DAS1	DAS2	DAS0	DAS1	DAS2
P1	I.	26	16	16	44	21	26		36	41
	T.	6622	5285	5795	20043	9582	14242		9488	11294
	O.	2e-4	4e-4	2e-4	2e-06	4e-05	9e-06		3e-05	2e-05
	L.		1	1		1	1		2	2
	C.		1	1		1	1		2	2
	S.		0.36	0.32		0.34	0.31		0.21	0.19
P2	I.	23	14	21	77	25	28	27	19	18
	T.	4332	3280	5249	13495	5163	6614	4231	3582	3428
	O.	1e-4	5e-05	1e-05	0	5e-05	2e-06	4e-05	5e-05	5e-05
	L.		2	4		1	2		2	2
	C.		2	4		4	2		2	2
	S.		0.39	0.34		0.293	0.3		0.37	0.36
P3	I.	20	14	15	20	14	16	20	14	18
	T.	1429	1304	1791	1419	1307	1875	1420	1308	1854
	O.	2e-4	2e-4	9e-05	2e-4	2e-4	0	2e-4	2e-4	3e-05
	L.		1	1		1	1		1	1
	C.		1	1		1	1		1	1
	S.		0.41	0.4		0.41	0.39		0.41	0.34
P4	I.	24	17	23	30	20	21		40	43
	T.	7186	6360	8171	11868	11048	14005		13146	12349
	O.	0	1e-4	2e-05	0	0	0		0	1e-06
	L.		2	2		2	2		3	3
	C.		2	2		5	2		3	3
	S.		0.38	0.32		0.36	0.36		0.2	0.19
P5	I.	24	18	24	21	16	15	23	17	21
	T.	3235	4420	6055	2679	4063	4085	2572	3885	5268
	O.	3e-4	2e-4	2e-06	1e-4	3e-4	2e-4	6e-05	4e-4	1e-06
	L.		2	2		2	2		2	1
	C.		2	2		2	2		2	1
	S.		0.33	0.292		0.36	0.35		0.346	0.27

Table 6.7: Results for Termination Criterion 1

4. **Dual Ascent Failures:** As compared to TC1, the number of failed iterations increases dramatically when TC2 is used to terminate the algorithm. As an example, P4 using DAS2 and CMS3 has 80 iterations (out of 173) in which the line search converges to zero. This is one of the reasons for the relatively poor performace of IDACG using TC2.

6.4.4 Conclusions

1. **Effect of Subproblem Complexity:** The success of IDACG (in any variant form) depends on the proportion of time spent in the master problem and subproblem. IDACG uses the subproblem much more often than does CG. Thus, the greater the proportion of time spent in the suproblem or the more intractable the subproblem is, the less likely will it be for IDACG to improve the performance of CG.

2. **Early Primal Optimality:** In the column generation solution of the CPP LP, primal optimality is reached well before dual optimality. This motivates the need for termination criteria other than the dual feasibility requirement of the LP optimality termination criterion.

3. **Comparison of IDACG strategies:** A comparison of dual ascent strategies shows that DAS1 (IDACG with generation of columns using the interior point dual feasible vector given by equation 4.11) is superior to DAS2 (IDACG with generation of columns using the boundary point dual feasible vector) more often than not. Moreover, DAS1 frequently performs better than DAS0 (conventional CG).

4. **Column Management Strategies:** It is not necessary to retain all the columns that are generated. By deleting some or all of the non-basic columns after each run of the restricted master problem, the optimality of the final solution will not be affected. Moreover, by keeping the problem size down, the solution of each restricted master problem can be achieved faster.

No.		CMS1			CMS2			CMS3		
		DAS0	DAS1	DAS2	DAS0	DAS1	DAS2	DAS0	DAS1	DAS2
P1	I.	32	24	34	44	32	35		106	110
	T.	9735	8363	11076	20043	18785	17330		32818	35772
	O.	0	5e-06	0	2e-06	0	0		0	0
	L.		1	2		1	1		22	27
	C.		1	2		6	1		23	27
	S.		0.43	0.34		0.36	0.34		0.12	0.11
P2	I.	43	23	25	77	41	44	62	25	32
	T.	10689	6878	5810	13495	9500	8975	10789	4763	5202
	O.	2e-06	0	0	0	0	0	2e-06	0	2e-06
	L.		2	4		1	2		2	2
	C.		4	4		15	3		2	2
	S.		0.467	0.38		0.29	0.29		0.41	0.34
P3	I.	27	20	18	27	22	21	27	26	23
	T.	1802	1754	2025	1795	1859	2234	1789	2075	2127
	O.	2e-06	0	9e-06	3e-06	0.0002	0	3e-06	0	0
	L.		1	1		1	1		1	1
	C.		1	1		1	1		1	1
	S.		0.5	0.46		0.48	0.41		0.42	0.4
P4	I.	24	32	35	30	28	27		140	173
	T.	7186	10670	11467	11868	16503	19745		40199	31497
	O.	0	0	0	0	0	1e-06		0	0
	L.		3	4		1	2		51	80
	C.		3	4		9	2		51	80
	S.		0.37	0.328		0.4	0.38		0.09	0.07
P5	I.	30	22	25	28	21	22	24	22	24
	T.	3684	5241	6268	3239	5209	5683	2636	4739	5632
	O.	0	2e-06	0	0	0	0	3e-05	1e-06	0
	L.		2	2		2	2		2	1
	C.		2	2		2	2		2	1
	S.		0.39	0.318		0.45	0.48		0.41	0.34

Table 6.8: Results for Termination Criterion 2

Three column management strategies are tested. CMS1 is the column management strategy in which all non-basic columns are eliminated each time the total number of columns exceeds a threshold. In CMS2, each time the number of column exceeds a threshold, all columns with a reduced cost above some threshold value are eliminated. CMS3 is similar to CMS2 except that the threshold reduced cost used to eliminate columns is an approximately evaluated median reduced cost. A comparison of column management strategies puts CMS3 last since some problems could not be terminated using it. From the results, CMS1 is preferred to CMS2. Moreover, CMS1 is attractive because it is non-parametric, i.e., it does not have any threshold values that have to be arbitrarily fixed. One of the reasons for the relatively poor performance of CMS2 is that the threshold value chosen to eliminate columns is very important in its performance. A high threshold value will cause the retention of a large number of columns and slow down the RMP solution time, while a low threshold value might cause the elimination of a large number of columns that later have to be regenerated. This results in a large number of iterations with small improvements per iteration. The reason for the poor performance of CMS3 is that the median reduced cost cannot be calculated easily by an approximate method, and the exact calculation is computationally expensive.

5. **Termination Criterion:** Since the tailing effect is very common in column generation, it is often not useful to run the algorithm to optimality. Premature termination of the algorithm using a good bounding technique gives close to LP optimal solutions. This is mainly due to the fact that during the LP solution process of CPP, primal optimality is reached much faster than dual optimality and hence early termination yields solutions that are close to the LP optimal solution. Dual ascent methods give tighter bounds when applied to column generation for CPP as compared to conventional column generation methods. This allows early termination of the algorithm when dual ascent is applied.

Two termination criterion - TC1 and TC2, are used. A comparison of the results using TC1 (termination at a bound gap of 0.05) and TC2 (termination at a bound gap of 0.01) shows that although the problem reaches very close to optimality using TC2 (optimality gaps are lower than 10^{-5}), the optimality gap is not too big using TC1 (optimality gaps are lower than 10^{-3}). Moreover, by using TC1, the algorithm can be terminated very quickly compared to TC2. As an example, consider the soluton of P1 using DAS1 and CMS1. Under the termination criterion TC1, the algorithm is terminated in 16 iterations, taking 5285 seconds. Using TC2 instead, the algorithm terminates in 24 (50% more) iterations, taking 8363 seconds (58.2% more). The smaller optimality gaps using TC2 do not warrant the extra effort required to achieve them. This favours the use of TC1 in solving such problems.

6. **Effect of Number of Flights per Pairing:** The degree of integrality of the optimal LP solution seems to depend on the average number of flights per pairing in the constraint matrix. The lesser this average number, the greater is the degree of integrality.

6.5 IP Solution of Large Problems

After solving the problems to LP optimality (or using a termination criterion), the constraint matrix is preprocessed before it can be used to find integer solutions.

6.5.1 Preprocessing the Constraint Matrix

The constraint matrix is processed in three stages, namely

1. **Duplicate Column Elimination** in which identical columns are eliminated, and

2. **Fixing the One's** in which all columns which are at one in the optimal LP solution are fixed at one and are eliminated. The rows which they cover are also eliminated.

No.	Percentage Integrality	Criterion	Starting Matrix	Duplicate Elimination	Fixing 1's	Red. Cost Reduction
P1	46.9%	rows	1138	1138	536	536
		columns	42085	37224	32998	1937
P2	22.4%	rows	1128	1128	744	744
		columns	23988	21509	20887	3301
P3	12.7%	rows	943	943	771	771
		columns	47272	29994	29626	4505
P4	14.9%	rows	1012	1012	834	834
		columns	36016	33813	33490	5432
P5	36.9%	rows	1023	1023	613	613
		columns	41802	32602	31493	1425

Table 6.9: Results of Data Preprocessing for IP

3. **Reduced Cost based Elimination** in which an LP is solved over the new compressed matrix and all columns with reduced cost greater than a threshold value are eliminated. The threshold value is chosen to be 10.

This preprocessing helps to reduce the size of the constraint matrix and makes it easier to solve the IP. Table 6.9 shows the results of this preprocessing for the constraint matrices obtained by solving to optimality problems P1 through P5 using DAS1 and CMS1. Note that the greater the percentage integrality of the starting solution (given in the second column of table 6.9), the greater is the compression.

6.5.2 Comparison of Various IP methods

Three versions of the Black Box Strategy (refer to section 5.2.1) are used to solve the IP's.

1. In the first strategey (denoted BB1), the matrix is passed only through the first two stages of compression. One pass of EKKMPRE is made, and then the IP solver is called with a search limit of a 1000 nodes in the branch and bound enumeration tree.

2. In the second strategy (denoted BB2), the matrix is once again passed through stages 1 and 2 of compression. One pass of EKKMPRE is made before it is

input into the IP solver with a search limit of 10000 nodes.

3. The third strategy (denoted BB3) processes the matrix through all three stages of the compression strategy. After passing through two stages of EKKMPRE, it is input into the IP solver with a search limit of 10000 nodes.

Table 6.11 compares the performance of the three methods. Clearly, BB3 is the best method. One of the reasons for the better performance of BB3 as compared to BB1 and BB2 is that the IP is run on a smaller matrix in case of BB3. There is a tradeoff in reduction of matrix size. By reducing the size of the matrix used by the IP solver, there are fewer solutions to choose from and hence the cost of the best IP solution may not be very good. Larger matrices allow a relatively larger number of solutions. However, smaller matrices have the advantage of having smaller run times using an IP solver as compared to larger problems. This decrease in run time is due to two reasons:

1. The smaller the constraint matrix, the lesser time taken per pivot.

2. The smaller the matrix, the fewer the number of variables and the smaller the branch and bound tree. Thus, given a limitation on the number of nodes that one can search, one can cover a larger portion of the branch and bound tree if the problem is smaller.

In this case the run time factor seems to be more dominant and hence the smaller matrices proved to have better results. An important observation is that as the number of flights per pairing increases, it becomes increasingly difficult to obtain good IP solutions. This is due to two effects.

1. From the results, it has been seen that the fewer the number of flights per column, the greater is the percentage integrality of the optimal solution. The percentage integrality of the optimal LP solutions are given in table 6.10.

2. A larger percentage integrality results in greater compression in the data and increased tractibility of the IP.

Problem	Percentage Integrality
P1	47
P2	22
P3	11
P4	14
P5	36

Table 6.10: Percentage Integrality of Optimal LP Solutions

Problem	Flight per Pairing	BB1	BB2	BB3
P1	6.00	1.64% 48 minutes	1.64% 276 minutes	0.69% 89 minutes
P2	7.09	3.49% 58 minutes	3.11% 277 minutes	0.30% 169 minutes
P3	7.83	3.86% 138 minutes	3.69% 578 minutes	1.24% 510 minutes
P4	9.60	17.53% 241 minutes	15.08% 692 minutes	7.66% 851 minutes
P5	5.54	6.4% 52 minutes	5.79% 237 minutes	4.13% 168 minutes

Table 6.11: Performance of IP methods

6.5.3 Conclusions

Problems P3 and P4 were terminated after 3000 and 6750 nodes respectively when BB3 was used due to excessive run times. Tests on the IP's yield the following conclusions:

1. The branch and bound approach is more effective when the problem size is controlled and kept small.

2. Methods BB1 and BB2 are identical except for the number of nodes searched in the branch and bound tree. Table 6.11 shows that good solutions are obtained relatively quickly and there is a diminishing rate of return as the number of nodes searched is increased.

3. The smaller the average number of flights per pairing, the greater is the degree of integrality of the optimal LP solution, the better the compression of the data, and the better the IP solutions that are obtained.

It should be mentioned that the above method may not yield optimal IP solutions. The achievement of optimal IP solutions will require the generation of new columns at each node as discussed in section 3.5. This has however, not been attempted in this thesis and will be a topic of future research.

Chapter 7

Further Work

This thesis concentrates on speeding up the column generation algorithm since it is used many times during the solution of large integer programs using branch and bound. Although computational experience is provided on obtaining IP solutions, further work has to be done in the area of combining column generation and branch and bound into a single framework. Barnhart, et al.(1993b) address the issues associated with combining the two procedures. Additionally, there are several other ideas that should be tested, namely

1. While running the shortest path subroutine, all the valid pairings that have a negative reduced cost are picked from the shortest path spanning tree. In the early stages of the algorithm, the dual variables are far away from optimality and hence the columns generated may not be used in the optimal LP solution. This might justify the generation of fewer columns in the early iterations using rules of thumb such as picking out only those columns with reduced costs less than a certain (negative threshold) and changing the threshold with each iteration.

2. The initial feasible dual solution for IDACG has been chosen to be a vector of zero's. There are however, several ways of choosing the initial solution. For example, an initial dual vector could set the dual variable corresponding to each flight equal to the flying time of the flight.

3. The step factor in the dual ascent heuristic is fixed at 0.3 based on early computational results. The effect of this step factor on the performance of IDACG could be tested.

4. In the dual ascent step, after a new dual feasible vector is found and the shortest path problem is run with respect to this vector, the shortest path spanning tree is examined and only negative reduced cost pairings (with respect to the current dual iterate) are extracted. Instead, all valid pairings could be extracted from the spanning tree or less extreme, pairings with positive reduced costs but less than some threshold could be extracted.

5. In the present implementation, artificial columns are used to initiate the algorithm. The algorithm could be modified to begin with an initial feasible solution obtained using deadheads, if necessary. Work on deadhead selection has been completed (Barnhart, et al., 1991b) and this research could be incorporated.

Appendix A

Problem Parameters

Relevant problem parameters are given below:

1. **Brief Time:** 60 minutes

2. **Debrief Time:** 30 minutes

3. **Minimum Connect Time:** 30 minutes

4. **Maximum Sit Time:** 420 minutes

5. **Maximum Pairing Length:** 21600 minutes (15 days)

6. **Time-Away-From-Base Proration Factor:** 1

An assumption is made that the time-away-from-base cost is the dominant cost component. Thus the other costs can be ignored and this component can be given a weight of one.

Appendix B

Typical Rest Rules for Long Haul Carriers

Before discussing the rules for the rest times, some definitions are in order.

B.1 Definitions

1. **Short Overnight:** This is the shortest period of rest. This has a value of 480 minutes.

2. **Long Overnight:** This is the longest period of rest. It has a value of 960 minutes.

3. **International Overnight:** This has a value of 720 minutes.

4. **Duration of a Duty Period:** This is equal to the total elapsed time including the brief and debrief times.

5. **Real Flying Time:** This is equal to the total flying time of the duty period excluding deadhead flying times.

6. **Short Duty Period:** A duty period whose duration is less than a specified number SHORTDP and a real flying time of less than a specified number SHORTFLY.

A deadhead has zero real flying time. SHORTDP has a value of 720 minutes and SHORTFLY has a value of 480 minutes.

B.2 Rules of Rest

The rules for the rest times for duty periods are discussed below.

1. **Duties with two or more flights:** If such a duty period has an international segment, then the duty requires an international overnight. If not, if the duration of the duty is less than SHORTDP and the real flying time is less than SHORTFLY, then a short overnight will suffice. If not, a long overnight has to be provided.

2. **Single flight duties:** Deadhead segments normally need only short overnights, unless it is flown to an international city in which case it needs an international overnight. If however, the deadhead segment has a duration larger than SHORTDP, it requires a long overnight. If the single flight is not a deadhead, the same rules apply as for duties with more than 1 flight.

References

[1] ABARA, J. Applying integer linear programming to the fleet assignment problem. *Interfaces 19*, 4 (July-August 1988), 20–28.

[2] AGARD, J. Optimal selection of aircraft. In *10th AGIFORS Symposium* (Sydney, Austrailia, November 1970). J. Agard worked for Air France.

[3] AHUJA, R. K., MAGNANTI, T. L., AND ORLIN, J. B. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall Inc., Englewood Cliffs, New Jersey, 1993.

[4] ANBIL, R., BARNHART, C., HATAY, L., JOHNSON, E. L., AND RAMAKRISHNAN, V. Crew-pairing optimization at american airlines decision technologies. In *Optimization in Industry* (1993), T. Ciriani and R. Leachman, Eds., John Wiley & Sons Ltd., pp. 31–36.

[5] ANBIL, R., GELMAN, E., PATTY, B., AND TANGA, R. Recent advances in crew-pairing optimization at american airlines. *Interfaces 21*, 1 (January-February 1991), 62–74.

[6] ANBIL, R., TANGA, R., AND JOHNSON, E. L. A global optimization approach to crew scheduling. Tech. Rep. COC-9105, Georgia Institute of Technology, 1991b.

[7] APPELGREN, L. H. A column generation algorithm for a ship scheduling problem. *Transportation Science 3* (1969), 53–68.

[8] ARABEYRE, J., FEARNLEY, J., STEIGER, F., AND TEATHER, W. The airline crew scheduling problem: A survey. *Transportation Science 3* (1969), 140–163.

[9] BAKER, E., AND FISHER, M. Computational results for very large air crew scheduling problems. *OMEGA, The International Journal of Management Science 9*, 6 (1981), 613–618.

[10] BAKER, E. K. Efficient heuristic algorithms for the weighted set covering problem. *Computers and Operations Research 8*, 4 (1981b), 303–310.

[11] BAKER, E. K., BODIN, L. D., FINNEGAN, W. F., AND PONDER, R. J. Efficient heuristic solutions to an airline crew scheduling problem. *AIIE Transactions 11*, 2 (June 1979), 79–85.

[12] BALAKRISHNAN, A., CHIEN, W. T., AND WONG, R. T. Selecting aircraft routes for long-haul operations: A formulation and solution method. *Transportation Research-B 24B*, 1 (1990), 57–72.

[13] BALL, M., AND ROBERTS, A. A graph partitioning approach to airline crew scheduling. *Transportation Science 19*, 2 (May 1985), 107–126.

[14] BARNHART, C. Dual-ascent methods for large-scale multi-commodity flow problems. Tech. Rep. COC-91-14b, Georgia Institute of Technology, 1992.

[15] BARNHART, C., HATAY, L., AND JOHNSON, E. L. Deadhead selection for the long haul crew pairng problem. Tech. Rep. COC-91-02, Georgia Institute of Technology, 1991b.

[16] BARNHART, C., AND JOHNSON, E., March-May 1993. Personal communication.

[17] BARNHART, C., JOHNSON, E. L., ANBIL, R., AND HATAY, L. A column generation technique for the long haul crew assignment problem. Tech. Rep. COC-91-01, Georgia Institute of Technology, 1991.

[18] BARNHART, C., JOHNSON, E. L., ANBIL, R., AND HATAY, L. A column generation technique for the long-haul crew assignment problem. In *Optimization in Industry, Volume II* (1993), T. Ciriano and R. Leachman, Eds., John Wiley and Son.

[19] BARNHART, C., JOHNSON, E. L., HANE, C. A., AND SIGISMONDI, G. An alternate formulation and solution strategy for multi-commodity network flow problems. Tech. Rep. COC-9102, Georgia Institute of Technology, 1991c.

[20] BARNHART, C., JOHNSON, E. L., NEMHAUSER, G. L., SAVELSBERGH, M. W., AND VANCE, P. H. Branch-and-price: Column generation for solving huge integer programs, 1993b. In progress.

[21] BARUTT, J., AND HULL, T. Airline crew scheduling: Supercomputers and algorithms. *SIAM NEWS* (November 1990).

[22] BAUM, S., AND TROTTER JR., L. Integer rounding for polymatroid and branching optimization problems. *SIAM Journal of Algebra and Discrete Methods 2*, 4 (December 1981), 416–425.

[23] BELGRAY, D. C. Discussion of : Optimal selection of aircraft by j. agard. In *10th AGIFORS Symposium* (Sydney, Austrailia, November 1970). David C. Belgray worked for Eastern Airlines.

[24] BEN-AKIVA, M., AND LERMAN, S. R. *Discrete Choice Analysis:Theory and application to travel demand.* MIT press, Cambridge, Massachusetts, 1985.

[25] BRADLEY, S. P., HAX, A., AND MAGNANTI, T. L. *Applied Mathematical Programming.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1977.

[26] CARRARESI, P., GALLO, G., AND ROUSSEAU, J. Relaxation approaches to large scale bus driver scheduling problems. *Transportation Research-B 16B*, 5 (1982), 383–397.

[27] Chandrasekaran, R., and Tamir, A. On the integrality of an extreme solution to pluperfect graph and balanced systems. *Operations Research Letters 3*, 4 (October 1984), 215–218.

[28] Christofides, N., Mingozzi, A., and Toth, P. State-space relaxation procedures for the computation of bounds to routing problems. *Networks 11* (1981), 145–164.

[29] Crainic, G. T., and Rousseau, J.-M. The column generation and airline crew scheduling problem. *INFOR 25*, 2 (1987), 136–151.

[30] Dantzig, G. B. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1963, ch. 22.

[31] Dantzig, G. B., and Wolfe, P. Decomposition principles for linear programs. *Operations Research 8* (1960), 101–111.

[32] Dantzig, G. B., and Wolfe, P. The decomposition algorithm for linear programs. *Econometrica 29*, 4 (October 1961), 767–778.

[33] Denardo, E. V., and Fox, B. L. Shortest-route methods: 1. reaching, pruning and buckets. *Operations Research 27* (1979), 161–196.

[34] Deo, N., and Pang, C.-y. Shortest-path algorithms:taxonomy and annotation. *Networks 14* (1984), 275–323.

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

[36] Desrochers, M., and Soumis, F. A generalised permanent labelling algorithm for the shortest path problem with time windows. *INFOR 26*, 1 (1988), 191–212.

[37] DESROCHERS, M., AND SOUMIS, F. A reoptimization algorithm for the shortest path problem with time windows. *European Journal of Operations Research 35* (1988b), 242–254.

[38] DESROCHERS, M., AND SOUMIS, F. A column generation approach to the urban transit crew scheduling problem. *Transportation Science 23*, 1 (Feb 1989), 1–13.

[39] DESROSIERS, J., DUMAS, Y., DESROCHERS, M., SOUMIS, F., SANSO, B., AND TRUDEAU, P. A breakthrough in airline crew scheduling. *Working Paper 6* (March 1991).

[40] DESROSIERS, J., DUMAS, Y., SOLOMON, M. M., AND SOUMIS, F. Time constrained routing and scheduling, June 1993. Draft.

[41] DESROSIERS, J., SOUMIS, F., AND DESROCHERS, M. Routing with time windows by column generation. *Networks 14* (1984), 545–565.

[42] DESROSIERS, J., SOUMIS, F., DESROCHERS, M., AND SAUVE, M. Methods for routing with time windows. *European Journal of Operational Research 23* (1986), 236–245.

[43] DIJKSTRA, E. A note on two problems in connexioon with graphs. *Numerische Mathematik 1* (1959), 269–271.

[44] DRUCKERMAN, J., SILVERMAN, D., AND VIARAPULOS, K. Optimization Subroutine Library, Guide and Reference, Release 2, July 1991. Document Number SC23-0519-2, IBM, Kingston, NY.

[45] DUMAS, Y., DESROSIERS, J., AND SOUMIS, F. The pickup and delivery problem with time windows. *European Journal of Operational Research 54* (1991), 7–22.

[46] ELCE, I. The development and implementation of air canada's long range planning model. In *10th AGIFORS Symposium* (Sydney, Austrailia, November 1970). Ivan Elce worked for Air Canada.

[47] Erlenkotter, D. A dual-based procedure for uncapacitated facility location. *Operations Research 26*, 6 (November-December 1978), 992–1009.

[48] Etschmaier, M. M., and Mathaisel, D. F. X. Airline scheduling : An overview. *Transportation Science 19*, 2 (1985), 127–138.

[49] Farley, A. A note on bounding a class of linear programming problems, including cutting stock problems. *Operations Research 38*, 5 (1990), 922–924.

[50] Fisher, M. L. The lagrangian relaxation method for solving integer programming problems. *Management Science 27*, 1 (January 1981), 1–18.

[51] Garfinkel, R., and Nemhauser, G. The set-partitioning problem:set covering with equality constraints. *Operations Research 17* (1969), 848–856.

[52] Gershkoff, I. Optimizing flight crew schedules. *Interfaces 19*, 4 (July-August 1989), 29–43.

[53] Gilmore, P., and Gomory, R. A linear programming approach to the cutting stock problem. *Operations Research 9* (1961), 849–869.

[54] Gilmore, P., and Gomory, R. A linear programming approach to the cutting stock problem, part ii. *Operations Research 11* (1963), 863–888.

[55] Girard, D. The airlines operations model: A schedule development and evaluation tool. In *13th AGIFORS Symposium* (Acapulco, Mexico, September 1973), pp. 195–209. Denis Girard worked for Air Canada.

[56] Guignard, M. A lagrangean dual ascent algorithm for simple plant location problems. *European Journal of Operational Research 35* (1988), 193–200.

[57] Guignard, M., and Kim, S. Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Mathematical Programming 39* (1987), 215–228.

[58] Guignard, M., and Opaswongkarn, K. Lagrangean dual ascent algorithms for computing bounds in capacitated plant location problems. *European Journal of Operational Research 46* (1990), 73–83.

[59] GUIGNARD, M., AND ROSENWEIN, M. B. An improved dual based algorithm for the generalized assignment problem. *Operations Research 37*, 4 (1989), 658–663.

[60] HALL, R. W. Configuration of an overnight package air network. *Transportation Research-A 23A*, 2 (1989), 139–149.

[61] HAOUARI, M., DEJAX, P., AND DESROCHERS, M. Modelling and solving complex vehicle routing problems using column generation. Tech. Rep. G-90-22, Les cahiers du GERAD, May 1990.

[62] HARA, M., AND KOYAMA, R. Short term planning model for domestic operation. In *13th AGIFORS Symposium* (Acapulco, Mexico, October 1973), pp. 183–194. The authors worked for Japan Air Lines.

[63] HEARN, D. W., AND LAWPHONGPANICH, S. Lagrangian dual ascent by generalised linear programming. *Operations Research Letters 8*, 4 (August 1989), 189–196.

[64] HELD, M., WOLFE, P., AND CROWDER, H. P. Validation of subgradient optimization. *Mathematical Programming 6* (1974), 62–88.

[65] HOUCK JR., D. J., PICARD, J. C., QUEYRANNE, M., AND VEMUGANTI, R. The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Opsearch 17*, 2&3 (1980), 93–109.

[66] JONES, R. D. Development of an automated airline crew bid generation system. *Interfaces 19*, 4 (July-August 1989), 44–51.

[67] KELLEY, A., AND POHL, I. *A Book on C.* The Benjamin/Cummings Publishing Company, 1990.

[68] KERNIGHAN, B. W., AND RITCHIE, D. M. *The C Programming Language.* Prentice Hall, 1988.

[69] KOLEN, A. W. J., KAN, A. H. G. R., AND TRIENEKENS, A. W. J. M. Vehicle routing with time windows. *Operations Research 35*, 2 (March-April 1987), 266–273.

[70] KONIG, P. Short term aircraft assignment. In *16th AGIFORS Symposium* (July 1976), pp. 169–188. Peter Konig worked for Swiss Air.

[71] LAMPORT, L. LaTeX*, A Document Preparation System: User's Guide and Reference Manual.* Addison-Wesley Publishing Company, 1986.

[72] LASDON, L. S. *Optimization Theory for Large Systems.* Macmillan Publishing Co., Inc., New York, 1970.

[73] LAVOIE, S., MINOUX, M., AND ODIER, E. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research 35* (1988), 45–58.

[74] LEMKE, C., SALKIN, H., AND SPIELBERG, K. Set covering by single branch enumeration with linear-programming subproblems. *Operations Research 19* (1971), 998–1022.

[75] MAGNANTI, T. L., SHAPIRO, J. F., AND WAGNER, M. Generalized linear programming solves the dual. *Management Science 22*, 11 (July 1976), 1195–1203.

[76] MAGNANTI, T. L., AND WONG, R. T. Network design and transportation planning: Models and algorithms. *Transportation Science 18*, 1 (February 1984), 1–55.

[77] MARCOTTE, O. The cutting stock problem and integer rounding. *Mathematical Programming 33*, 1 (1985), 82–92.

[78] MARCOTTE, O. An instance of the cutting stock problem for which the rounding property does not hold. *Operations Research Letters 4*, 5 (February 1986), 239–243.

[79] MARSTEN, R. E., AND SHEPARDSON, F. Exact solution of crew scheduling problems using the set partitioning model: Recent successful applications. *Networks 11* (1981), 165–177.

[80] McDOWELL, E. 2 airlines struggle to revive their glory days, Thursday, Septermber 2 1993. Business Day - New York Times.

[81] McDOWELL, E. U.s. rivals join northwest air in steep discounts, Tuesday, Septermber 14 1993b. Business Day - New York Times.

[82] MINOUX, M. Column generation techniques in combinatorial optimization - a new application to crew pairing. *XXIV AGIFORS Symposium* (Sept. 1984), 15–29.

[83] ODIER, E., LASCAUX, F., AND HIE, H. Medium haul trip pairing optimization. *XXIII AGIFORS symposium* (October 1983), 81–109.

[84] PARKER, M., AND RYAN, J. A column generation algorithm for bandwidth packing. *Telecommunications Systems* (1993). To appear.

[85] RANNOU, B. A new approach to crew pairing optimization. *XXVI AGIFORS Symposium* (Oct. 1986), 153–167.

[86] RAPLEY, K. Short haul fleet planning models. In *15th AGIFORS Symposium* (Rotorua, New Zealand, October 1975), pp. 307–338. Keith Rapley worked for British Airways.

[87] RIBIERO, C. C., MINOUX, M., AND PENNA, M. C. An optimal column-generation-with-ranking algorithm for very large set partitioning problems in traffic assignment. *European Journal of Operational Research 41* (1989), 232–239.

[88] RICHETTA, O., AND ODONI, A. Dynamic solution to the ground holding problem in air-traffic control. *Transportation Research* (1992). Working paper accepted for publication.

[89] RICHETTA, O., AND ODONI, A. Solving optimally the static ground holding policy problem in air-traffic control. *Transportation Science 27* (1993). Forthcoming.

[90] RUBIN, J. A technique for the solution of massive set covering problems, with applications to airline crew scheduling. *Transportation Science 7*, 1 (Feb 1973), 31–48.

[91] RYAN, D., AND FOSTER, B. An integer programming approach to scheduling. In *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (1981), A. Wren, Ed., North Holland Publishing Company, pp. 269–280.

[92] RYAN, D. M., AND FALKNER, J. C. On the integer properties of scheduling set partitioning models. *European Journal of Operational Research 35* (1988), 442–456.

[93] SAVELSBERGH, M. A branch-and-price algorithm for the generalized assignment problem. Tech. Rep. COC-9302, Georgia Institute of Technology, 1993.

[94] SHAPIRO, J. F. *Mathematical Programming: Structures and Algorithms.* Wiley-Interscience, John Wiley and Sons, 1979.

[95] SHENOI, R. G. Another dual ascent method for generalized linear programming, 3rd May 1993. Term paper submitted for 1.966 - Large Scale Optimization in Transportaion.

[96] SHEPARDSON, F., AND MARSTEN, R. E. A lagrangean relaxation algorithm for the two duty period scheduling problem. *Management Science 26*, 3 (March 1980), 274–281.

[97] SHERALI, H. D., AND MYERS, D. C. Dual formulations and subgradient optimization strategies for linear programming relaxations of mixed-integer programs. *Discrete Applied Mathematics 20* (1988), 51–68.

[98] SIMPSON, R. W., AND BELOBABA, P. 16.74, air transportation economics, Fall 1993. Class notes.

[99] SOLOMON, M., AND DESROSIERS, J. Time window constrained routing and scheduling problems. *Transportation Science 22*, 1 (February 1988), 1–13.

[100] SOLOMON, S. D. American airlines: Going, going ..., Sunday, Septermber 5 1993. The New York Times Magazine.

[101] SUBRAMANIAN, R., SCHEFF, R. P., QUILLINAN, J. D., WIPER, S. D., AND MARSTEN, R. E. Coldstart: Fleet assignment at delta airlines. *Interfaces* (1993). To appear.

[102] TEODOROVIC, D., AND GUBERINIC, S. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research 15* (1984), 178–182.

[103] THIRIEZ, H. Airline crew scheduling:a group theoretic approach. *Working Paper R69-1* (October 1969).

[104] VANCE, P. *Crew Scheduling, Cutting Stock, and Column Generation:Solving Huge Integer Programs.* PhD thesis, Georgia Institute of Technology, August 1993.

[105] VANCE, P., BARNHART, C., JOHNSON, E. L., AND NEMHAUSER, G. L. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications* (1993b). To appear.

[106] VASQUEZ-MARQUEZ, A. American airlines arrival slot allocation system (asas). *Interfaces 21*, 1 (January-February 1991), 42–61.

[107] WONG, R. T. A dual ascent approach for steiner tree problems on a directed graph. *Mathematical Programming 28* (1984), 271–287.

[108] WREN, A., SMITH, M., AND MILLER, A. Complementary approaches to crew scheduling. In *Computer Scheduling of Public Transport 2* (1985), J.-M. Rousseau, Ed., North Holland Publishing Company, pp. 263–278.