Imperial College of Science Technology and Medicine
(University of London)
Management School

CONTRIBUTIONS TO THE SOLUTION OF THE CREW SCHEDULING PROBLEM

by

GWAN-HO PAEK

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and for the
Diploma of Imperial College

May 1992



ACKNOWLEDGEMENTS

I would like to express my first thanks to my supervisor Professor Nicos Christofides for his kind advice throughout this research. His warm encouragements and invaluable comments have been the main powers to circumvent a lot of bottlenecks throughout the research period.

The sponsoring from the Korean Electronics and Telecommunications Research Institute should be acknowledged. They provided me with the opportunity to carry out this research. Their financial support has been a great help to my study.

A partial sponsoring from the British Council during my first year should also be acknowledged. Even though the sponsoring was limited to one year, it was very helpful in the most difficult days of the first year in London.

I also wish to show my gratitude to the friends and colleagues for their timely help. The discussions with colleagues in the Management School of Imperial College have been very refreshing and conducive to innovative ideas in the course of my work. In particular the friendly advice about computing from Dr Andreas Johannsen is deeply appreciated. I am thankful to Dr Young-gun Kwon for his arduous task of reading the drafts and providing many corrections. Any remaining errors should be attributed only to me.

My deepest gratitude must go to my family both those far away in Korea and those with me in London. They have been much more understanding and dedicated than any other family could. Without the help of my mother and my sister in my country, the study may have been much more difficult. Especially, I cannot thank too much for their enduring patience, my wife Kyong-sook and three daughters Aram, Nari and Yesol. Although this is their first experience in living abroad, they have not uttered a single word of complaint.

ABSTRACT

In this thesis we consider three exact and one approximate procedure for the solution of Crew Scheduling Problems, such as those found in airline crew scheduling.

The first algorithm (called as the Clique Aggregation Technique - CAT) is based on a formulation dependent on the expansion of a graph representing all flights and their possible transitions. We present a method of expressing network-flow-type constraints which when used to surrogate them, produces a strong LP relaxation.

The second algorithm (called as the Iterative Time-constraint Addition algorithm - ITA) is based on the unexpanded flight-transition graph and uses the LP-relaxation solution to identify which constraints to add, as and when needed. These two algorithms are tested computationally and are shown to be able to solve small CSP's optimally.

The third exact algorithm, based on a unique combination of Lagrangean Relaxation and the generation of k-best optimal solutions to associated Assignment Problems, produces excellent lower bounds which, when embedded in a tree search can solve CSP's with more than 100 flights in less than one-hour computing time on an IBM PC/AT.

Finally, we describe a heuristic based on local optimization and using network-flow to search for possible improvements. This heuristic has produced optimal results on all the problems tried and whose optimal solution is known.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ---------- 1

ABSTRACT ---------- 2

TABLE OF CONTENTS ---------- 3

CHAPTER 1 : INTRODUCTION ---------- 7

1-0. Summary ---------- 7

1-1. The Crew Scheduling Problem ---------- 8

1-2. General Assumptions and CSP Extensions ---------- 13

1-3. Application Examples of the Crew Scheduling Problem ---------- 15

1-4. NP-completeness and Computational Complexity ---------- 18

1-5. Major Approaches and Techniques ---------- 21

1-6. General Trends of CSP Methodology in Industrial Practice ---------- 26

1-7. Research Objectives and Thesis Outline ---------- 28

CHAPTER 2 : THE CLIQUE AGGREGATION TECHNIQUE ---------- 30

2-0. Summary ---------- 30

2-1. Introduction ---------- 31

2-1-1. General ---------- 31

2-1-2. The Network Expansion Approach to the CSP ---------- 36
2-2. An Alternative Formulation Based on $\overline{G}$ (CAT) ---------- 44
2-2-1. Objective Function with Crew Costs ---------- 47
2-2-2. Flight-covering Constraints in the CAT Formulation ---------- 50
2-2-3. Flow-conservation Constraints in the CAT Formulation ---------- 52
2-3. Non-integer Solutions ---------- 55
2-4. A Tree Search Procedure ---------- 62
2-5. Computational Results ---------- 66
2-6. Conclusion ---------- 70

CHAPTER 3 : ITERATIVE TIME-CONSTRAINT ADDITION ALGORITHM ------ 72
3-0. Summary ---------- 72
3-1. Introduction ---------- 73
3-2. Variables in ITA Algorithm ---------- 80
3-3. Objective Function of CSP in ITA ---------- 84
3-4. Constraints in ITA Formulation ---------- 84
3-4-1. Generation of Constraints from the LP Solution ---------- 85
3-4-2. Generation of Insurance Constraints ---------- 89
3-5. A Tree Search Algorithm for Obtaining an Integer Solution ---------- 91
3-6. Computational Results ---------- 92
3-7. Conclusion ---------- 97

CHAPTER 4 : A SET-PARTITIONING BASED HEURISTIC ---------- 98

4-0. Summary ---------- 98

4-1. Introduction ---------- 99

4-1-1. The Classical SCP Formulation of the CSP ---------- 99

4-1-2. The Problem Generation ---------- 100

4-1-2. The Problem Solution ---------- 101

4-2. Heuristic Methods ---------- 102

4-2-1. Greedy Heuristics ---------- 103

4-2-2. Reduced Cost Heuristics ---------- 104

4-3. An Initial Heuristic Solution to the CSP ---------- 104

4-3-1. Generation of Columns for the SPP ---------- 104

4-3-2. The Construction of the Initial Solution ---------- 106

4-4. A Local Improvement Procedure ---------- 108

4-4-1. The Search for a Local Improvement ---------- 110

4-4-2. The Choice of $\mathbb{S}'$ ---------- 114

4-5. Computational Results ---------- 118

4-6. Conclusion ---------- 122

CHAPTER 5 : A TREE SEARCH ALGORITHM FOR THE CSP ---------- 123

5-0. Summary ---------- 123

5-1. Introduction ---------- 123

5-2. Lagrangean Relaxation and k-best Optimal Solution ---------- 126

5-2-1. A Test for Optimality ---------- 128

5-2-2. Derivation of a Lower Bound ---------- 129

5-3. Computation of Lower Bound ---------- 129

5-4. A Dual Tree Search ---------- 134

5-5. Computational Results ---------- 138

5-6. Conclusion ---------- 141

CHAPTER 6 : CONCLUSIONS ---------- 143

REFERENCES ---------- 145

Chapter 1

INTRODUCTION

1-0. Summary

This chapter provides the basic concepts of the Crew Scheduling Problem (CSP) with its general development trends, importance and intractability to set up the research objective of the thesis. In the first section, the definition of an ordinary CSP is given as a kind of cost minimisation scheduling problem on a time-constrained graph. With a simple illustration of the CSP network diagram, some terminologies are also given. In Section 2 the underlying assumptions are categorised according to the characteristics of the subsystems such as the number of depots, airplane types, time windows etc. These assumptions are intended to provide the systematic understanding of the whole structure of CSP variants and extensions. Brief comments about CSP extensions for each assumption are also given in this section. Some application examples of the CSP's are put forward in Section 3 to show the wide applicability of this problem in the real world. The CSP is one of the most widely applied combinatorial optimization problems in the fields of airline, bus, truck and railway scheduling. However, solving the CSP is not an easy task. The computational complexity such as the NP-completeness is briefly discussed to understand why it is so difficult to solve the CSP. Most of the methodologies used in practice are based on heuristic algorithms. This situation provides the fundamental raison d'être of our study in this thesis. We set up our research objective as the development of practical algorithms to tackle large size CSP's. More specifically, our research is concentrated on the development of efficient algorithms for the "rotation" problems in airline crew scheduling.

1-1. The Crew Scheduling Problem

The scheduling of crews is a well-known combinatorial problem which has attracted considerable attention from both industries and the operational research community. A tremendous amount of man-hours have been devoted to this classical problem due to its mathematical importance and economic impact. Like the other mathematical problems with "good" properties it is very simple to state but, in contrast, difficult to solve. No mathematical background is needed to understand it. The problem, however, has refuted all the efforts so far to find a good algorithm for it. This continuing inability to solve it in an optimal way has only added to the charm of this problem.

In our straightforward expression, the Crew Scheduling Problem (CSP) is defined as the ***cost minimisation scheduling problem on the time constrained graph.*** The objective is to find the least cost set of paths covering every one of a given set of tasks (flights) only once within the given work-duty-time during the planning period.

From now on, as long as there arises no confusion, we use interchangeably the term "CSP" and "airline crew scheduling problem (ACSP)" as well as their corresponding terminologies because the ACSP is a typical kind of CSP. "***Task***", or "***Flight***" in ACSP, is the work to be done at a certain place during a predetermined time period. The flight is denoted by its starting time, finishing time, starting place and finishing place. "***Path***", or "***Rotation***" in ACSP, is a set of tasks combined on the single route. The duration of each path should not be longer than the given work-duty-time. "***Work-duty-time***" is the maximum allowable duration of a path, which is usually 8 hours a day by the union regulations. "***Planning Period***" is the time duration of the whole CSP scheduling from the earliest starting time to the latest finishing time of the

tasks. It might be one day for a small problem or several weeks for a large one.

Let us illustrate a simple configuration of the CSP using an example from airline crew scheduling. Consider the following example of a 5-flight CSP in Tables 1-1 and 1-2. We assume there is no work-duty-time limit for the time being. Even though this is a basic CSP which can be solved by using a minimum cost network flow formulation, we believe this can provide a general concept for the CSP.

Table 1-1. Flights of CSP Example

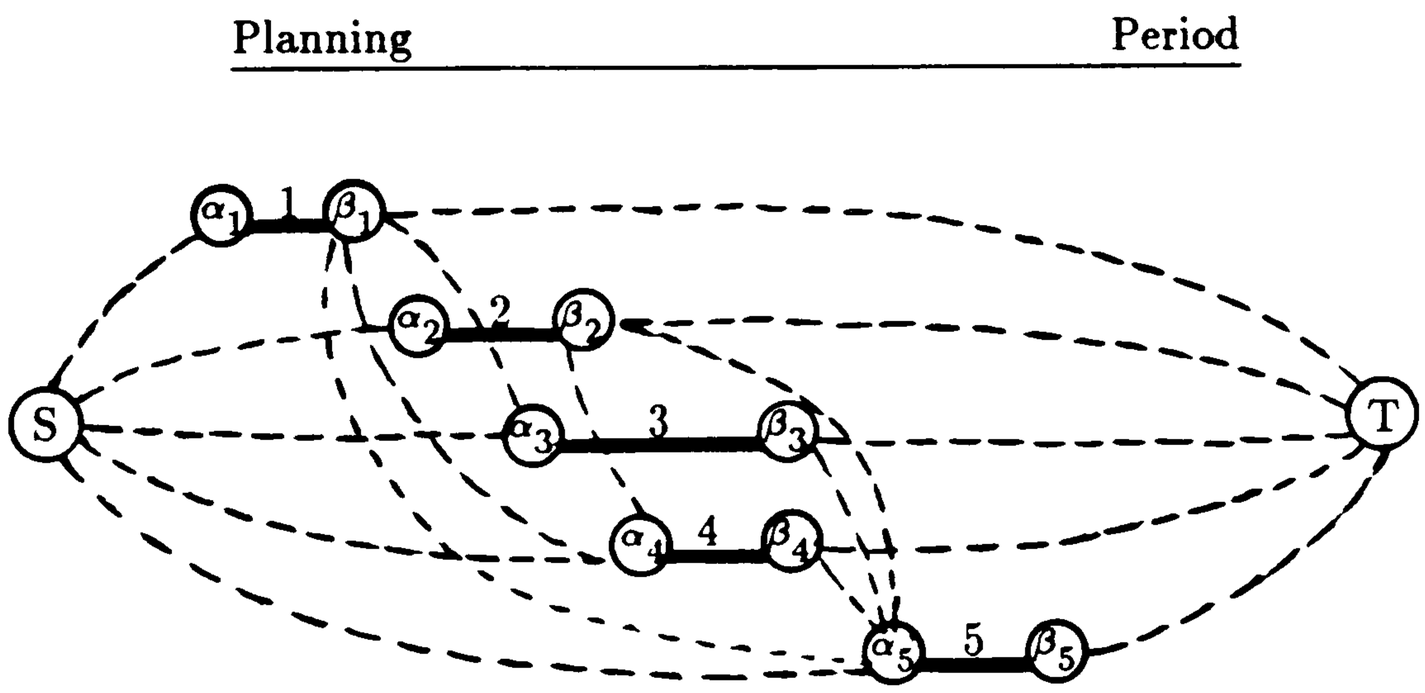
Flight No	Starting Time	Finishing Time	Starting Place	Finishing Place
1	20	70	A	B
2	30	80	A	C
3	120	180	D	A
4	170	210	C	D
5	260	300	B	A

Table 1-2. Transition Time of CSP Example

Place	A	B	C	D
A	0	40	50	60
B	50	0	70	30
C	50	70	0	50
D	60	30	50	0

The above example can be expressed in the following graph-theoretic representation (See Figure 1-1). Consider a pilot starting his work on flight 1 which is defined by the flight between the nodes α_1 and β_1. After working on flight 1, he cannot start working on flight 2 because the finishing time of flight 1 is later than the starting time of flight 2. He can, however, work on flight 3 or 4 or 5 as long as the work-duty-time permits. In case the arriving airport of flight 1, say London, is different from the leaving airport of flight 3, say Paris, he must go to Paris to work on flight 3. Of course this transition needs some time and some cost i.e. transportation cost or lodging cost for any overnight stay. In a similar manner, we can construct several possible paths of flights with corresponding costs. Our attempt is to find the ***set of paths***, among all the possible paths, to cover all the flights with the least total cost.

Figure 1-1. Configuration of CSP



The above example is expressed in the following well-known mathematical formulation which can be solved by using a minimum cost network flow formulation as shown by Ford and Fulkerson *[1962]*.

$$\text{minimise} \quad \sum_{(\beta_i,\alpha_j)\in \overline{\mathrm{N}}} \mathrm{C}_{ij}\mathrm{X}_{ij}$$

$$\text{subject to} \quad \sum_{j\in \mathrm{V}_i^-} \mathrm{X}_{ij} - \sum_{p\in \mathrm{V}_i^+} \mathrm{X}_{pi} = \begin{cases} 0 & \text{for } i = 1, \ldots, n \\ -K & \text{for } i = \mathrm{T} \\ K & \text{for } i = \mathrm{S} \end{cases} \Bigg\} \overline{\mathrm{P\,N}}$$

$$\mathrm{X}_{ij} \in \{0,1\}, \qquad \forall\ i, j \text{ such that } (\beta_i,\alpha_j)\in \overline{\mathrm{N}}$$

where (β_i,α_j) is the transition arc from flight i to flight j

C_{ij}= linking cost between flights i and j [cost of arc (β_i,α_j)]

X_{ij}= 1 if arc (β_i,α_j) is in the optimal solution

0 otherwise

$\overline{\mathrm{N}}$ = { set of all feasible transitions }

V_j^+= { $i \mid (\beta_i,\alpha_j)\in \overline{\mathrm{N}}$ }

V_j^-= { $i \mid (\beta_j,\alpha_i)\in \overline{\mathrm{N}}$ }

S is the super-source

T is the super-sink

K = number of paths

n = number of flights

Problem $\overline{\mathrm{P\,N}}$ is clearly a network flow problem and can be solved quite easily. However, if a maximum work-duty-time restriction exists, the solution of the above problem is not necessarily feasible. It would, for example be infeasible if the time duration of a path in the solution exceeds the given work-duty-time. For example, if we assume work-duty-time is 100 minutes, the path 1 - 3 with the time duration of 160 minutes cannot be included in the solution. Usually these additional work-duty-time constraints are so numerous that we cannot solve the problem easily.

The CSP can also be considered as a kind of public transportation problem *[Mitra et al 1985]*. In the hierarchy of the planning process of public transportation system, the CSP is recognised as the tactical problem of the last phase. This implies that the

structure of the CSP depends heavily on the higher planning process to deal with more global transport problems. The planning process of the public transport system is usually divided into four phases, namely *[Gouvras 1987]*:

1) The planning of the transport network

2) The determination of the time table for each route

3) The routing of the vehicles to meet the time table

4) The scheduling of the crews to run the vehicles

Meanwhile from the theoretical point of view in combinatorial optimization, the CSP can be considered as a special case of the Travelling Salesman Problem (TSP) *[Orloff 1976]* and Quadratic Assignment Problem *[Garfinkel 1972, Connolly 1989]*. The CSP is a task-time-constrained and multi-salesmen-allowed TSP. Based on the methodology for TSP, several algorithms with the same theoretical background have been developed for CSP *[Giannessi 1979]*. The CSP is also similar to a kind of shortest spanning tree (SST) problem in which the length of every tree is limited by the maximum duration; which is equivalent to the work-duty-time in the CSP. In the algorithm of Gavish and Srikanth a heuristic solution of the CSP is produced by exploiting the spanning tree solutions generated during the tree search to obtain upper bounds for the CSP *[Gavish et al 1983]*.

In particular it is generally recognised that the CSP has a very close connection with the Vehicle Routing Problem (VRP) in many respects. The basic structure of the CSP is so similar to that of the VRP, (the former involving more complicated restrictions), that it is not unusual that they are considered in the same papers at the same time. They are often interacting with each other especially in practical solution methodologies. Several major workshops to deal with these similar themes

simultaneously have been held in Chicago 1975, at the University of Maryland 1979, in Leeds 1980 and in Montreal 1983.

These two problems can be considered in the same formulation at the same time. In fact Ball et al suggested a technique to try to schedule vehicles and crews simultaneously *[Ball et al 1983]*. However, this kind of combined approaches are comparatively few. Then the natural question can be raised. Why are there considerably fewer attempts to deal with them in the same formulation, (not just in the same paper), than ones which consider them separately? The answer is quite simple. The combined routing and scheduling problems are extremely difficult to deal with owing to the complexity of the interaction between the two component subproblems. For example the problems of a school bus, subscriber dial-a-ride and full loading with time windows cause a myriad of difficulties when routing and scheduling operations in these problems must be performed simultaneously. The detailed description of this problem can be found in Bodin and Golden *[1981]*. For the general VRP, interested readers can refer to Christofides *[1985]* and Golden et al *[1987]*.

1-2. General Assumptions and CSP Extensions

Every CSP model is built up on some assumptions which are derived inevitably from the simplifying processes of real world situations. Some of them are expressed explicitly in the model and some are hidden implicitly. For the ordinary CSP we suppose main assumptions according to the basic characteristics of sub-components such as number of connections, time windows, number of depots etc. Then the CSP can be classified into various types by relaxing these assumptions. Our intention is, however,

not to discuss the theoretical background of these assumptions but only to illustrate them for the overall understanding of the CSP. The general assumptions of CSP and their corresponding extensions can be listed as follows.

Table 1-3. Assumptions and CSP Extensions

Code	Characteristic	Possible Options
DCSP	depot number	one depot - multiple depots
VCSP	vehicle type	one type - many types
TCSP	time window	fixed time - variable time window
RCSP	rest	no rest - rest during work-period

Assumption 1 : *Single Depot*

In the CSP, it is assumed that there is only one depot from which all paths should start and revert to after work. If we drop this assumption, a CSP extension with multi-depot (DCSP) is generated. This DCSP can be found frequently in the real world problem of airline transportation.

Assumption 2 : *Single Type of Homogeneous Vehicles*

It is assumed also for the CSP that there is only one type of vehicle or that the airline company and the crew have no differences based on the vehicle types. Every crew can work on every airplane. If the flights are to be covered by various vehicle types and certain crews can only work on certain types of vehicles, this assumption is relaxed to get the CSP with vehicles of multi-type (VCSP). This is also, like the DCSP, one of the most frequently encountered practical problem types for airline operations.

Assumption 3 : *Fixed Starting and Finishing Time*

The starting times and flight durations of each flight are assumed to be fixed. It is certain whether two corresponding flights can be linked in this problem. But in the CSP with time windows on the flight starting times (TCSP), we cannot say exactly whether the flights could be linked or not. The possibility of linking them depends on the actual starting time of the previous flight within its allowed time window.

Assumption 4 : *Rest during Work -period*

It is assumed that rest periods are required during the crew work-period in the ordinary CSP. However, in practice the union sets the upper and lower limit on a rest period, for example in a total of 8 hours working; a continuous rest period of 1 hour may be required in such a way that no continuous work-period for the crew exceeds $4\frac{1}{2}$ hours. We must then drop the assumption of "no rest" (RCSP).

1-3. Application Examples of the Crew Scheduling Problem

The CSP and its extensions have various applications in real life problems. A significant number of real world applications, of both theoretical and practical importance, are found to be direct extensions of the CSP. We present here some of these application examples which are, of course, not all the cases but only a small part of the main examples. This presentation is neither meant to be exhaustive, nor meant to describe the subject in a new light.

The application examples can be found widely in many distribution and

transportation fields. Since Dantzig and Fulkerson have tried to minimise the number of tankers to meet a fixed schedule *[Dantzig et al 1954]*, the application cases of CSP have been increasing in great numbers. Applications are also being extended to the general mass transit systems such as:

- airlines *[Arabeyre 1969, Giannessi 1979, Baker 1980, Ball et al 1985, Crainic 1987, Hong 1990]*
- ship *[Dantzig et al 1954, Bellmore et al 1971, Fisher et al 1985, Psarafis et al 1990]*
- bus *[Orloff 1973, Krolak 1978, Bodin 1979, Mitra et al 1981, Swersey 1984, Paixão 1989]*
- truck *[Bodin 1978]*
- train *[Florian 1976, Assad 1980]*
- general mass transit systems *[Ball et al 1980, Ward 1981, Ball et al 1983, Desrochers 1989]*

First as its name denotes, the CSP methodology can be applied directly to the general crew scheduling problems in transportation or distribution. It is generally recognised that, from a practical point of view, the effective scheduling of crews has saved and will continue to save government and industry many millions of dollars a year. The distribution cost added to purchased goods is about 400 billion dollars in the United States and some 15 billion pounds in the U.K. per year. Among these amounts, the cost associated with operating vehicles and crews forms an important component *[Bodin 1983]*.

The most frequently applied example, however, is found in the airline crew scheduling problem (ACSP) for airline operations. Air crews are amongst the most valuable resources of airlines and their efficient utilisation is obviously an important consideration. The airline crew cost is usually said to be $10 \sim 15\%$ of the total cost of airline operations, which is the second highest component of direct operating cost after

fuel *[Gershkoff 1989]*. The large U.S. airlines each pay several million dollars each month for hotel rooms to lodge their crews for the next flight. Even an improvement of 1% in the current solution of ACSP in Alitalia corresponds to a profit of more than half million dollars per year *[Giannessi 1979]*. It was also reported that approximately quarter million dollars per year on crew costs have been saved for the Flying Tiger Line, the world's largest air cargo carrier, by a computer based crew scheduling package *[Marsten et al 1981]*.

The CSP methodology can be applied to the scheduling of military air force operations as well. However, there are several major differences between the scheduling for a military airlift operation and the scheduling for a commercial airline. First, an airlift operation does not require restrictions on the possible routes for a crew. A second difference is the lack of a fixed schedule for the airlift operation. For detailed discussions, one can be referred to Sklar et al *[1990]*.

In the general mass transit organisation, the new crew schedule will usually be used for a long time, about half a year, and manpower costs account for a large share of the total expenses, roughly speaking around 40%. It is therefore very important to build an optimal or near optimal crew schedules so as to save the total cost over a long period. In fact, the saving of 1% in a medium sized city such as Montreal would have represented nearly 2 million dollars in 1986 *[Desrochers 1989]*. Another example shows that the RN, a large mass transit transport company in Portugal, saved about 2 million Portuguese escudos annually, applying the CSP method to the transportation management in the Lisbon district *[Paĩxao 1986]*.

The application case of CSP is in no way limited to the transportation sector. The idea and methodology can be applied to general distribution system. For example the basic structure of the CSP is similar to that of the job scheduling problem. The

cases of applying CSP for job shop scheduling are found in Lenstra *[Lenstra 1979, Lenstra 1981]*. The rotation of workforce is another main application of the CSP methodology *[Balakrishnan et al 1990]*. Similar examples are also found in the problem of garbage collection and street sweeping *[Bodin et al 1978]*.

1-4. NP-completeness and Computational Complexity

From a mathematical point of view, the CSP is one of the computationally untractable ***combinatorial problems in the NP-complete family***. Despite the vast amount of work devoted to this problem, both from the theoretical and practical points of view, this problem remains as one of the unsolved combinatorial problems. The optimal solution to this combinatorial problem is composed of selecting several paths of nodes over a set of feasible paths. The problem is very easy to describe and to understand, but very difficult to solve. Interested readers in combinatorial optimization problems are referred to Lawler *[1979]*, Christofides et al *[1979]*, Klee *[1980]*, Papadimitriou et al *[1982]*.

The computational complexity of an algorithm may be defined as the number of elementary operations necessary to solve an instance of a problem. The size of an instance of a problem is measured by the length of the shortest coding necessary to specify the data completely. Given an instance of size n and a real function $g(n)$ of n, we say that the complexity of an algorithm is $O(g(n))$, if $f(n)$, the maximum time required to execute the algorithm is such that : $|f(n)| < c|g(n)|$ where c is constant depending on the type of computer. Algorithms in $O(n)$ are called linear, those in $O(n^k)$ are called polynomial and those in $O(2^n)$ are called exponential *[Papadimitriou et al 1982]*.

All problems can be categorised into two groups, undecidable or decidable. Some undecidable problems are proved so hard that no algorithm, from a computing point of view, can be given for them at all. The decidable problems, in turn, have two subgroups. The problems of the first subgroup cannot be solved in polynomial time even with a non-deterministic machine. As far as the level of difficulty in solving the problems is concerned, this category of problems comes directly after the class of undecidable problems. Since Hartmanis constructed this categorization *[Hartmanis1965]*, Fisher and Meyer have found natural problems of this category *[Fisher 1974, Meyer 1972]*. In contrast to the first subgroup, the second subgroup of decidable problems consists of all the problems to be solved in polynomial time with a non-deterministic machine. These problems are said to be included in the *Non-deterministic Polynomial class* (NP-class). All the problems of class P that can be solved in polynomial time by deterministic algorithms, which are said to belong to class P, also belong to the NP-class. There is a class in NP in which every problem of that subgroup has the property so that every other problem in the same subgroup can be polynomially converted to it. Cook proved that the algorithm for one problem of this subgroup can be applied to all the other problems, or vice versa *[Cook 1971]*. These problems in a sense are the hardest in NP. They are called NP-complete problems. Cook showed a list of NP-complete problems *[Cook 1973]*. Karp also presented a collection of NP-complete problems of well known combinatorial problems *[Karp 1972]*. For the general computational complexity, the interested reader may consult the works of Karp*[1975]*, Tarjan *[1978]*, Garey and Johnson *[1979]*, Lenstra and Rinnooy Kan *[1979]*, Sandi *[1979]*, Lewis and Papadimitriou *[1981]* and Papadimitriou and Steiglitz *[1989]*.

The problems of combinatorial optimization are consequently grouped into two categories. The first type is the NP-complete problem which, from a complexity point of

view, is hard to solve. The second type is the polynomial problem for which efficient solution techniques already exist *[Lenstra 1979]*. The CSP is an NP-complete problem *[Ball 1980, Ball 1980a, Bodin 1981]*. For the detailed discussion about the computational complexity of vehicle routing and scheduling problems one can be referred to Lenstra and Rinnooy Kan *[Lenstra et al 1979, Lenstra et al 1981]*. The CSP is so large and complex that only the small to medium size problems of up to 150 flights per week have been solved by the exact mathematical techniques so far (using mainframe computers) *[Marsten 1978, Marsten 1981]*. Therefore the large size airline companies which have some thousands of flights per week have no option but to use practical algorithms based on heuristics *[Baker 1979]*.

For example by the general IP formulation for the CSP we need the following number of variables and number of constraints to solve the n-flight CSP in the worst case. The worst case means that all flights are disjoint with randomly arranged starting times with very large transition times among them and that the work-duty-time is large enough to cover all the flights in a single path. In this case we must consider as variables, all the combinations between two corresponding flights (the first term ${}_nC_2$ in the below expression). The combinations between starting and finishing depots and flights, (the second term $2n$), must be considered also to get all the necessary variables. The number of variables V_g then becomes:

$$V_g = {}_nC_2 + 2n$$

At least there must be two variables, in other words three flights, in a constraint. Therefore, roughly speaking, the maximum number of constraints C_g, which is also the number of time constraints, becomes:

$$C_g = \sum_{i=3}^{n} {}_nC_i$$

The above expressions give an indication of the CSP size in a mathematical programming formulation which might give us a simple reason why the CSP is so difficult to solve. As shown in Table 1-4, the number of constraints is increasing so drastically that we cannot deal with those CSP's with more than 20 flights which may have about one million constraints in the worst case.

Table 1-4. Example of CSP Size in IP-based Formulation

Flights	Variables	Constraints
10	65	968
15	135	32,647
20	230	1,048,365
25	350	33,554,106
30	495	1,073,741,358

The argument that faster computers may reduce the computing time does not hold water. Even if we have a computer one million times faster than the present ones, we cannot expect to solve even 25-flight CSP in a routine way (without algorithmic sophistication) even with a year's computing time.

1-5. Major Approaches and Techniques

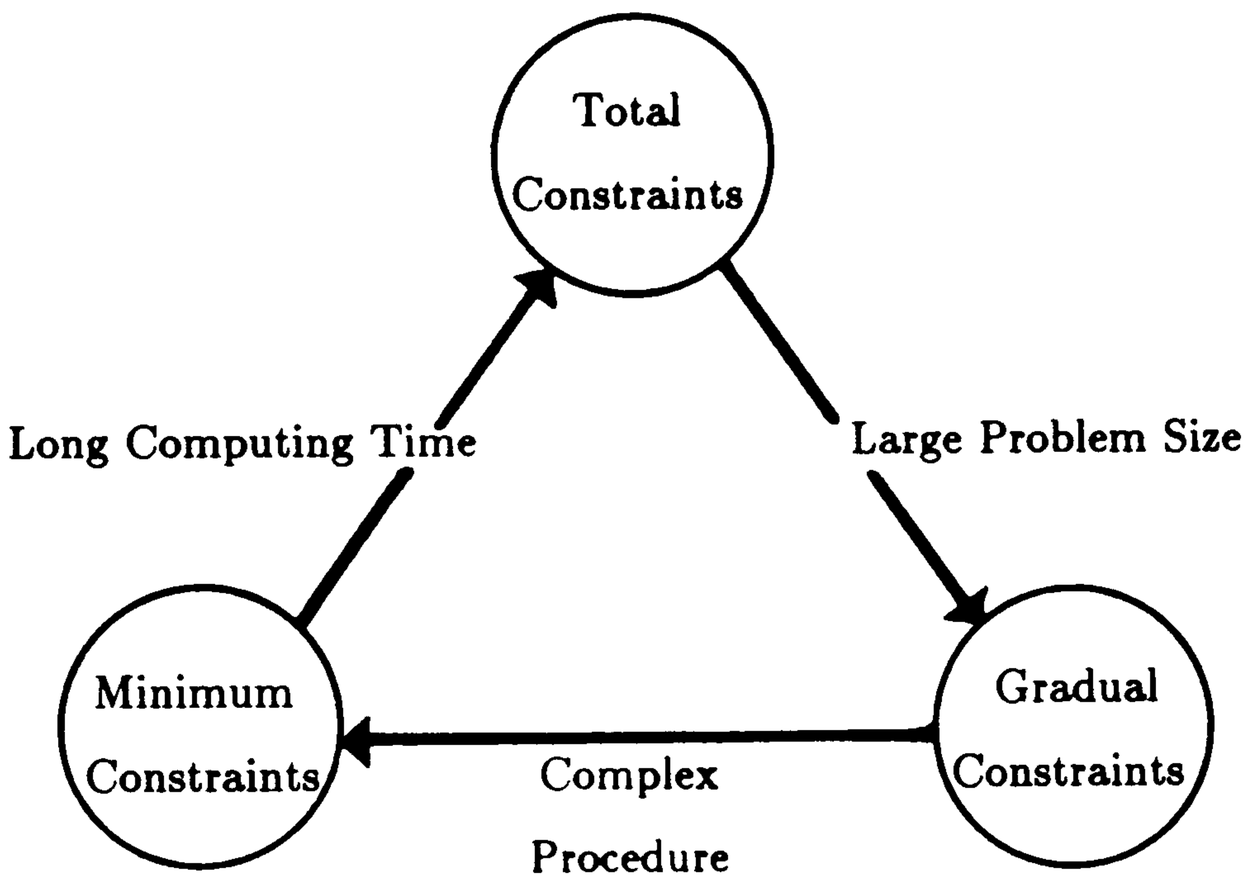
This section gives simply an overview of the major approaches found in the

literature. Detailed literature surveys are found in each chapter of this thesis dealing with the technique that is the central theme of that chapter.

The approaches to the CSP vary widely in the basic philosophy and mathematical sophistication. Our objective in this section is, therefore, to assemble these approaches as far as possible and regroup them into a few generic categories so that similarities and differences may be easily seen.

First the major approaches to the CSP might be categorised basically into three groups according to their basic philosophies concerning the management of the time constraints. The size of constraints is believed to be the most critical bottleneck in solving the CSP and also for the general combinatorial optimization problems obtained from CSP extensions. Let us label them the ***Total Constraints*** approach, the ***Gradual Constraints*** approach and the ***Minimum Constraints*** approach.

Figure 1-2. Basic Approach Pattern in CSP



The *Total Constraints* approach is formulating the problem with all possible

constraints. This seems the most straightforward and intuitive approach, but the resulting complexity usually makes the formulation too large to be handled as a single entity. Accordingly this untreatable size of the problem is relaxed in the hope that the relaxed problem should be significantly easier to solve than the original one. Most techniques including mathematical programming and the graph-theoretic method fall under this category. We try this "*once and for all*" approach in the algorithm CAT in Chapter 2.

The *Gradual Constraints* method is starting from the problem at first with the minimum required constraints, usually the network flow constraints. If the solution from this formulation is not feasible, other necessary constraints are added. The problem is solved on the extended formulation again perhaps using incremental techniques. These procedures are iterated until the feasible optimum is obtained. Normally the processes of selecting suitable constraints and adding them to the primal problem are complex and time consuming. Our algorithm ITA in Chapter 3 is a member of the family using this approach.

The third approach, *Minimum Constraints*, is formulating the problem with minimum required constraints like the *Gradual Formulation* approach except that what is being introduced gradually are not "complete constraints", but constraint "elements" which when finally taken together, describe a large number of constraints. The other constraints are not added to the problem but considered only when they are necessary for the feasibility test of a solution. The direct tree search is a typical example of this kind. The conspicuous drawback of this approach might be the heavy burden of computing time for the generation of appropriate lower bounds and feasibility test on them.

There can be another classification of the approaches from a different point of

view. For example all approaches can also be grouped into two types, *exact algorithms* and *heuristic algorithms* according to the type of the final solution provided by them. Let us review briefly the general trend of this classification and its corresponding techniques.

The exact algorithms are aiming to get the exact solutions. Most theoretical studies are concentrated on this approach. The main stream of this sort of CSP researches is devising mathematical techniques to solve larger problems *[Arabeyer 1969, Graham et al 1970]*. But by this approach it is difficult to solve the large size CSP's of the real world. Until now this approach can provide solutions only for small to medium size CSP's (up to 150 flights per week) *[Marsten et al 1978, Marsten et al 1981]*.

The impotence of exact algorithms has expedited the heuristic methods to solve the large size CSP. The concurrent scheduler *[Bodin et al 1978a]*, the interchange heuristic *[Smith et al 1981]*, the set covering based heuristic *[Toregas 1972]*, Orloff's heuristic *[Orloff 1973]* and the two step approaches for multiple depot CSP *[Bodin et al 1978a]* are under this category of heuristic approaches. Even though they cannot guarantee optimal solutions, it cannot be denied that a good heuristic solution is better than no solution at all. Most commercial packages have been developed on this heuristic concept. These approaches have been studied vigorously in the late 1960's and in the mid 1970's to yield voluminous works. Most of these algorithms are based on greedy principles and its modifications. More recently considerable progresses have been made in improving the efficiency of the heuristics *[Baker 1979, Baker 1979a, Baker 1980, Hinson 1975]*. It was reported that CSP's with over 3000 flights have been solved by set covering based heuristics *[Baker 1981]*.

The approaches to CSP can be classified also by their theoretical backgrounds such as *linear programming approach*, *graph theoretic approach* and *tree search* and so

on. This is the usual methodological classification.

From the theoretical point of view, the mathematical programming approaches, which usually take the form of IP, look fancier than any other approaches. This formulation, however, does not guarantee the integrality of the solution. The focal point of this approach therefore has been laid on the development of the method of getting integer solutions in a reasonable period of time. For this purpose, two main methodologies, cutting plane methods and branch & bound procedures have been developed. Since the pioneering work of Gomory *[1958]*, Salkin and Kendall *[1973]* and Delorme *[1974]* have studied the cutting plane methodology to get integer solutions for LP's. It is difficult, however, to get the appropriate cuts for the large problems. In this case the branch & bound method is applied concurrently to get the integer solution. Most commercial IP packages are programmed on this basis. The critical point of this approach is how to determine the lower bound for which several techniques have been proposed such as Lagrangean Relaxation and State Space Relaxation. More detailed literature survey and discussion will be presented in Chapter 2. Introduction and extensive literature for this approach can be found in Jeroslow *[1979]*. The group theoretic algorithms for integer programming by Shapiro, White and Glover are also of the mathematical programming family *[Shapiro 1968]*.

The next approach, of equal importance and prevalence, is the graph theoretic approach such as the application of the Set Covering Problem (SCP), Set Partitioning Problem (SPP) and the Shortest Spanning Tree (SST). Usually these approaches can produce integer results faster, but produce large number of corresponding sets instead with the result that their formulations are much larger. These approaches therefore have been concentrating on the development of reducing the number of possible sets. The works of Applegreen *[1971]*, Balas, Padeberg *[1975]*, Parker*[1981]*, Smith*[1981]*,

Ward*[1981]* and Mitra*[1981, 1985]* are for this purpose.

The branch & bound (B & B) or tree search approach is the most intuitive. Implicit enumeration methods can be included in this category, but the efforts to find exact solutions using this kind of algorithm is frustrated by the number of tree nodes which becomes astronomical unless sophisticated and time consuming computations of upper and lower bounds are performed at each tree node. Most of the present tree search algorithms are designed with a view to provide a practically useful heuristic solution. For the detailed procedures, several papers by Balas *[1965]*, Beale *[1965]*, Dakin *[1965]*, Land et al *[1960]* are referred to. The practices of implicit enumeration method are found also in Pierce *[1968]*, Foster *[1976]*, Marsten *[1980]*, Shepardson *[1981]*, Chan Bean Yano *[1987]*, and Ryan & Falkner *[1988]*.

1-6. General Trends of CSP Methodology in Industrial Practice

In a commercial airline the flight schedule is a central element of the planning process. Before the development of special computer packages, and also for some of the present small airlines, manual process is the main method to set up crew schedules. This process is very lengthy, requires several weeks, and cannot respond to the frequent changing schedules in a real sense. The real schedules change almost everyday due to the status of marketing, resources, weather and other causes. During the 1950s, recognising the dynamic nature of the scheduling problem, OR professionals both within and outside the airlines have been working on the development of efficient methods to cope with this situation *[Brough 1962]*.

The consensus, however, is that the problem of aircraft crew scheduling is

unsolvable by only quantitative optimization techniques. The problem is recognised to be solved through a structural planning process in which all parts of the airline participate. Much of the work of this type emerged in the 1960s under the name of schedule development and evaluation systems *[Tobin and Butfield 1970]*. The work has been discussed extensively in the symposia and study group meetings of AGIFORS (Airline Group of the International Federation of Operational Research Societies), a kindred society of the IFORS, and at other professional meetings *[Etschmaier 1984]*. Almost without exception the approaches taken by most airlines have been the use of integer programming. Only a few airlines used other approaches such as Monte Carlo techniques *[Loo et al 1970]*.

A number of events such as the oil shock during the 1970s caused dramatic changes in the nature of airline operations. The rapid change of the environment and competition in discount fares forced the airlines into different patterns of operation. Schedules were no longer prepared a long time ahead of the implementation. In many airlines numerical programming was replaced by heuristics of user-friendly man-machine interactive procedures. The main interest has turned from "*generating an optimal crew schedule*" to "*improvement of an existing crew schedules*" *[Tingley 1979, Griesshaber 1980, Lubow 1981]*. From a practical point of view, heuristics save much time and cost than the exact solution approaches. For example, at the American Airlines the savings through the change from the exact solution oriented integer programming approach to the enumeration heuristic method was estimated to be 18 million dollars per year *[Gershkoff 1989]*.

The rapid development of computer technology in the 1980s has revived the mathematical approach in the CSP again. The problem size solvable by optimization methods has expanded greatly. Numerous new methods, especially exact solution based

methods but with heuristic terminations have emerged. It is evident, however, that the airlines do not yet have the efficient solution methods at hand for their full satisfaction.

1-7. Research Objectives and Thesis Outline

In this thesis we deal with the airline crew scheduling problem (ACSP). The ACSP is the most typical CSP containing all the important characteristics of CSP variants. It is also the most widely used application example in the real world situation.

In the scheduling problems of ACSP, there are phases to complete the whole scheduling for crew. They are: the "*rotation*" problem of daily or weekly crew scheduling; and the "*rostering*" problem to form monthly schedules with several rotations from the first phase. The second phase rostering problems, (which are also called bid-line problems), are not as cost-significant, so they attract relatively little consideration. In 1970, however, Alitalia airlines began to study a new system in the area of crew management. The crew rostering problem was immediately spotted as one of the functions with greater need of an automatic support. Crew rostering can be cast into the framework of the classical Assignment Problem for which exact algorithms are already in existence. However, in case of crews, the large dimensions of the problem and the human nature of the resources to be assigned make the problem particularly difficult to be handled by automatic procedures *[Nicoletti 1973]*. It has also been considered to combine the rotation and the rostering problems together. In this attempt the crew members are allowed to construct their own blocks of flights, to avoid the inequitability within the crew rank. Bids are usually accepted in the order of decreasing seniority in the crew rank. So many junior crews are seldom being allocated the

preferred line of work *[Moore et al 1978]*. Interested readers about rostering problems can refer to Nicoletti *[1973]*, Finnegan *[1977]*, Antosik *[1978]*, Giannessi *[1979]*, Mayer *[1980]*, Middleton *[1984]*, Glarnet *[1984]*, Jones *[1989]* and Ryan *[1989]*. In the remainder of this thesis we will consider only the rotation problems of the first phase.

In summary, we set up our research objective as ***the development of practical algorithms for the rotation problems of airline crew scheduling***. Our main consideration is to find efficient algorithms applicable to large size CSP's. We present two algorithms based on the LP formulation in Chapters 2 and 3 and one set partitioning-based heuristic in Chapter 4. Chapter 5 deals with an algorithm based on tree search technique. Final conclusions and comments are given in Chapter 6.

Chapter 2

THE CLIQUE AGGREGATION TECHNIQUE

2-0. Summary

In this chapter we consider an LP formulation with the focus on the reduction of the time constraints, and ultimately on the size of the whole problem. The *clique aggregation technique* (CAT), an improved version of the graph expansion technique (GET), is developed for this purpose. In our CAT algorithm, although we use the same technique to get the expanded graph as in GET, we give an alternative formulation of the time constraints which is much better than the one in GET when these constraints are surrogated to reduce their number. In other words our motivation in the CAT algorithm is to provide a formulation which is "*tight*" when constraints are surrogated. The crew cost is (additionally) considered in the objective function for increased "realism" of the model. The network flow-conservation constraints for the starting and finishing depots are a little different from those of the original GET formulation. These are discussed in Section 2. The formulation of the flight-covering constraints, which is the key part in the IP formulation of the CSP, are thus improved by our new method. If the solution to the LP relaxation of the IP model is not integer, we use a Transportation Problem (TP) and a tree search to find the integer optimal solution. The TP is formulated using the variables with non-integer values from the initial results. In case the cost of TP is the same as the objective value from LP relaxation of the initial formulation, we stop the algorithm because the solution is optimal . When the solution of the TP is still not optimal, we trace the optimal solution through the tree search starting from the upper bound obtained by the TP. The computational results and conclusions are presented in Sections 5 and 6 respectively. It is seen from these section that larger CSP's can be solved by this algorithm than those solved by the formulation used in the GET algorithm.

2-1. Introduction

2-1-1. General

The CSP can be easily formulated as an IP with zero-one variables but this formulation is not easy to solve. The two focal points of this approach are how to get the integer solution in a reasonable period of time and how to tackle problems of large size which require a large number of constraints. The main techniques for obtaining integer solutions are cutting plane methods, branch & bound methods and group theoretic approaches. The most prevalent techniques for deriving bounds in branch and bound methods are: Lagrangean Relaxation and State Space Relaxation.

The cutting plane method is, as its name implies, based on cutting off the feasible region without affecting the optimal integer solution until the integer solution can be obtained. Gomory cuts, Garfinkel cuts, convexity cuts, intersection cuts and logical cuts fall under this category. Since Gomory proposed the cutting plane concept in 1958 *[Gomory 1958]*, several algorithms and commercial packages have been developed *[Beale 1965, Garfinkel et al 1972, Taha 1975, Jeroslow 1975]*. Gomory and Cook proved that their cutting plane algorithm guarantees the optimal solution after a finite number of iterations *[Gomory 1963, Cook 1990]*. Many authors have noticed, however, that this algorithm does not converge fast enough for large size integer programs (IP's). Rubin suggested that the cutting plane algorithm should be stopped and embedded into a branch and bound procedure when the cutting planes become redundant *[Rubin 1973]*. This method has proved to be quite efficient for some problems *[Balas et al 1980, Crowder et al 1983]*.

Balas suggested the intersection cuts *[Balas 1971, Balas 1974]*, and Young

hypercylindrically-deduced cuts based on similar logical concepts *[Young 1971]*. Glover's surrogate constraint is also another variant of the cutting plane method. This concept was then developed into the convexity cuts *[Glover 1968, Glover 1973]*. Martin's accelerated Euclidian algorithm has been shown to be quite satisfactory *[Martin 1963]* in some cases of ACSP's and very special structure problems with up to 900 flights were reported to have been solved to optimality *[Giannessi 1979]*.

The idea of logical cuts was proposed by Owen in 1973 *[Owen 1973]*. Jeroslow set up the theoretical basis for this disjunctive method for logical cuts *[Jeroslow 1977, Jeroslow 1979]*. Balas and Padeberg applied the logical cut method to the Set Partitioning Problem and suggested that this logical cut has proved to be very effective among IP techniques *[Balas et al 1979]*.

Branch and bound (B & B) or tree search procedures might be the most successful and most widely used technique in integer programming, especially when the cutting plane method cannot provide satisfactory solutions in a reasonable period of time. Balas suggested an additive algorithm for the zero one programming in 1965 *[Balas 1965]*. Dakin proposed a tree search algorithm for mixed integer programming problems *[Dakin 1965]*. Beale presented a general survey of mathematical programming with the B & B method for IP *[Beale 1965, Beale 1979]*. Geoffrion and Marsten devised a special implicit enumeration procedure for IP *[Geoffrion, Marsten 1974]*. Interested readers are referred to Garfinkel who described the detailed procedures of general B & B for IP *[Garfinkel 1979]*.

There are now several commercial packages for IP based on the B & B technique. Since Land and Powell proposed a computer code for IP in 1960 *[Land and Doig 1960]*, IP techniques based on the B & B concept have been greatly improved. Most IP packages have been developed using this method *[Roy et al 1970]*. Forrest, Hirst and

Tomlin developed the commercial package UMPIRE for large mixed IP *[Forrest et al 1974]*.

The critical point in B & B is the determination of the lower bound at each node. The following algorithms have been developed with the focus on obtaining better lower bounds.

- Lagrangean Relaxation *[Held and Karp 1971]*
- State Space Relaxation *[Christofides et al 1981]*

Lagrangean Relaxation is one of the most widely used techniques for obtaining bounds for IP's to embed in B & B procedures *[Geoffrion et al 1972, Geoffrion 1974, Shapiro 1979, Fisher 1981]*. To briefly discuss Lagrangean Relaxation let us consider the following integer programming problem (P) *[Fisher 1985]*:

$$
\left.
\begin{array}{ll}
 & Z = \min\ cx \\
\text{subject to} & Ax \geq b \\
 & Dx \geq e \\
 & x \geq 0 \text{ and integer} \\
 & \\
\text{where} & x \text{ is } n \times 1,\ b \text{ is } m \times 1,\ e \text{ is } k \times 1 \text{ vector.}
\end{array}
\right\} \text{(P)}
$$

To create the Lagrangean problem with the constraints $Ax \geq b$ relaxed we define an m vector of non-negative multipliers u and add the term $u(b\text{-}Ax)$ to the objective function of (P) to obtain the following problem (LR_u):

$$\left.\begin{aligned} & Z_D(u) = \min\ cx + u(b\text{-}Ax) \\ \text{subject to}\quad & Dx \geq e \\ & x \geq 0 \text{ and integer} \end{aligned}\right\} (\mathrm{LR}_u)$$

For all applications, the relaxed Lagrangean problem has been chosen so as to be solvable in polynomial or pseudo-polynomial time. It is well known that $Z_D(u) \leq Z$. This fact allows (LR_u) to be used to provide lower bounds in a B & B algorithm for (P). This is the most obvious use of (LR_u). Good feasible solutions to (P) can, however, also frequently be obtained by perturbing "nearly feasible" solutions to (LR_u).

Since Agmon, Motzkin and Schoenberg suggested special algorithms to solve the relaxed problem in 1954, Gavish studied the method to get the best multipliers *[Gavish 1978]*. Bazaraa developed a method based on the duality concept *[Bazaraa et al 1977, Bazaraa 1979]*. Among these techniques, the most widely used is the subgradient method *[Crowder 1974, Held et al 1974, Sandi 1977]*. This method was successfully applied to the Travelling Salesman Problem *[Balas et al 1981]*, capacitated facility location problem *[Geoffrion et al 1978]*, and two duty period scheduling problem *[Shepardson et al 1980]*.

The subgradient method is a gradient-based method used to maximise $Z_D(u)$ with some adaptation at the points where $Z_D(u)$ is nondifferentiable. At the points where $Z_D(u)$ is nondifferentiable, the subgradient method chooses arbitrarily a solution from the set of alternative optimal Lagrangean solutions and uses the $b\text{-}Ax$ for this solution as though it was the gradient of $Z_D(u)$. The result is a procedure that determines a sequence of values for u, beginning at an initial point u^0 and applying the formula (*) *[Fisher 1985]*. The stepsize in the subgradient method should converge to 0, but not too quickly. These observations have been confirmed by Held, Wolfe and Crowder who prove that if $k \to \infty$, $t_k \to 0$ and $\sum_{i=1}^{k} t_k \to \infty$ *[Held et al 1974]*, then the following

procedure for updating u^k solves problem LR_u.

$$u^{k+1} = u^k + \max\{0,\ t_k(b\text{-}Ax^k)\} \quad \text{----------} (*)$$

where x^k is an optimal solution to LR_u^k and t_k is a scalar step size

A formula for t_k that has proven effective in practice is:

$$t_k = \frac{\lambda_k\{\mathrm{Z}^* - \mathrm{Z}_D(u^k)\}}{\sum_{i=1}^{m}\left(\ b_i - \sum_{j=1}^{n} a_{ij}X_j\right)^2}$$

where λ_k is a scalar chosen between 0 and 2

Z^* is the objective value of the best known feasible solution to (P)

$\mathrm{Z}_D(u^k)$ is the optimum value of problem LR_u^k

In most cases λ_k is initially set to 2 and is halved if there is no increase in the optimal value of LR_u^k after a given number of iterations.

Christofides proposed state space relaxation to be used for obtaining bounds in B & B which is an alternative to Lagrangean Relaxation *[Christofides et al 1981]*. This algorithm has proved to be effective for Travelling Salesman Problems *[Christofides 1979]* and Vehicle Routing Problems *[Christofides et al 1982]*. Despite the fact that most routing problems are easy to formulate by Dynamic Programming (DP), DP recursions are

difficult to solve because of the high dimensionality of the state space. So in this method the DP recursions are used not for solving the problem but for obtaining lower bounds. The lower bounds are achieved by relaxing the state space to a lower dimensionality. The general procedures of this algorithm are:

(1) formulate the problem as a dynamic program

(2) choose a mapping function to map into a lower-dimensional state space

(3) solve the relaxed recursion to obtain a bound

(4) use a state space ascent to improve the bound

(5) embed the bound in a branch and bound

Shapiro and Thiriez proposed the group theoretic approach to IP. Since Shapiro initially proposed the extended general group theoretic algorithm for the integer programming problem in 1968, Thiriez applied it to airline crew scheduling and extended it again to the set covering problem *[Shapiro 1968, Thiriez 1969]*.

2-1-2. The Network Expansion Approach to the CSP

All the above algorithms have not succeeded in solving even medium size CSP's in a real sense. Although it is sometimes said that the largest CSP size ever solved by the IP formulation so far is 150 flights, the usual size of CSP which can be solved in this way is much less than 100 flights. The bottleneck in IP formulations is the enormous number of constraints most of which are derived from the time restrictions in the CSP. Even the CSP's of small size can produce a formulation with so many constraints that it cannot be solved by general algorithms in a reasonable time.

The basic approaches to the CSP are grouped into three categories, the *total constraints*, *gradual constraints* and *minimum constraints [See Chapter 1]*. One algorithm in the first category is the graph expansion technique (GET). The focal point of the GET algorithm is to expand the flights into a network according to their starting time status and to introduce "transition arcs" into this network in order to incorporate the time restrictions. In the GET formulation, therefore, only the paths satisfying the work-duty-time are considered. This formulation succeeds in reducing the number of time constraints of the CSP. The critical drawback of the GET formulation, however, is that the number of variables increases drastically instead. The increase in the number of variables is so rapid that this formulation can only deal with small to medium size problems. Another drawback is that the solution by GET has not been proven to be integer in all cases. Cheddad *[1987]* showed that all his test problems had integer solutions, but he could not prove algorithm GET always guarantees integer solutions.

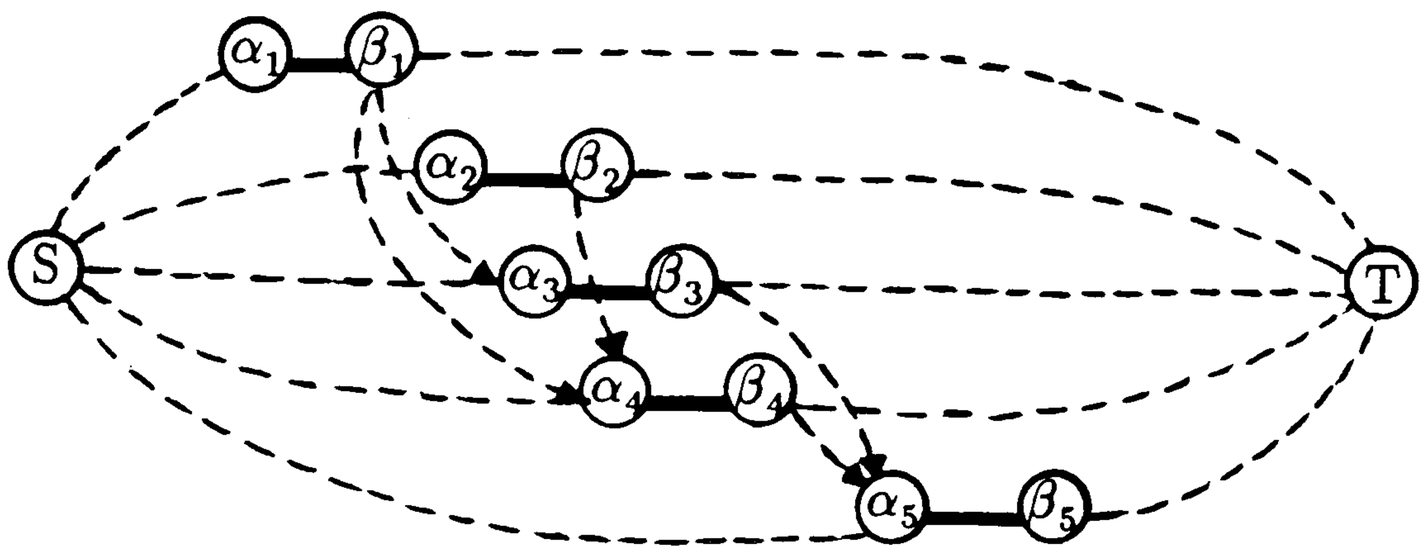
In this chapter we try to improve this approach in spite of the above shortcomings just because it is one of the few algorithms which provide an idea of how to reduce the initial problem size. Our attempt is to improve the GET algorithm to tackle larger size CSP's.

Let us first overview the formulation procedures used in the GET algorithm. In the GET formulation, each flight of the original CSP graph G is duplicated according to its starting position in the possible path combinations to produce an expanded graph $\widetilde{G}$. The sub-flights with the same destinations in this expanded graph $\widetilde{G}$ will be coalesced to produce the final graph $\overline{G}$.

For more detailed explanation, consider again the 5 CSP example in Chapter 1. We assume the work-duty-time is 200 minutes. Then the original graph G becomes (Note that the connection β_1 - α_5 and β_2 - α_5 are removed because their time durations

are greater than the work-duty-time):

Figure 2-1. Original Graph G of Example A



We define the set P(i) for the flight i:

$$P(i) = \{ \text{ a set of all feasible paths starting with flight } i \}$$

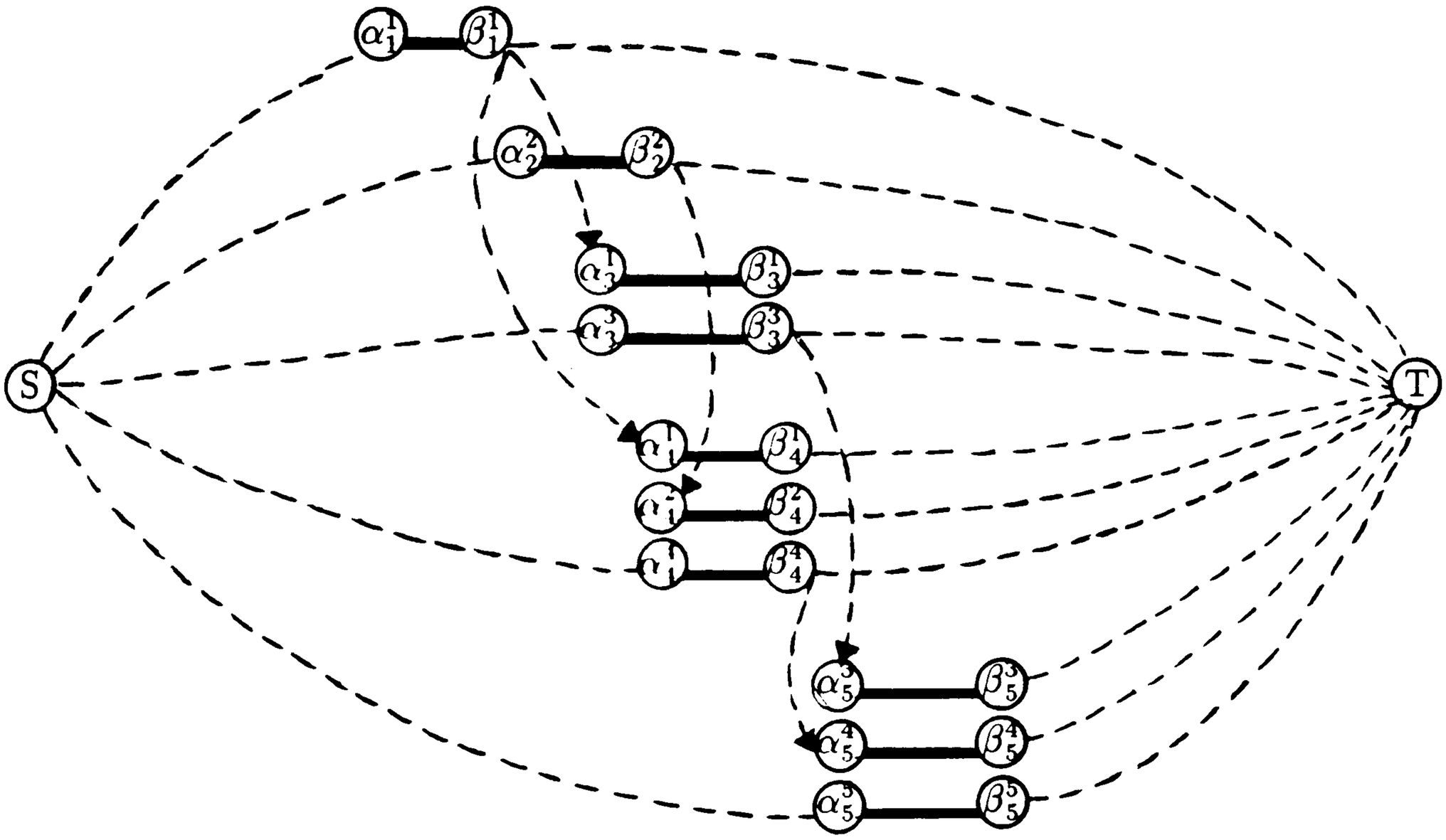
For example, in graph G, the path (α_1, β_1, α_3, β_3) is feasible because its time duration of 160 minutes is less than the work-duty-time but the path (α_1, β_1, α_4, β_4, α_5, β_5) with the time duration of 280 minutes is infeasible. Note that we omit the super-source S and super-sink T in the path for simplicity. In this manner, we obtain:

$$P(1) = \{(\alpha_1, \beta_1) \cup (\alpha_1, \beta_1, \alpha_3, \beta_3) \cup (\alpha_1, \beta_1, \alpha_4, \beta_4)\}$$

Then the flight (α_2, β_2) is included in P(1) or P(2) according to its position in a given path from S to T. We can expand the flight (α_2, β_2) into sub-flights (α_2^1, β_2^1) and (α_2^2, β_2^2) having the same starting and finishing times as (α_2, β_2). If (α_2^1, β_2^1) is chosen as

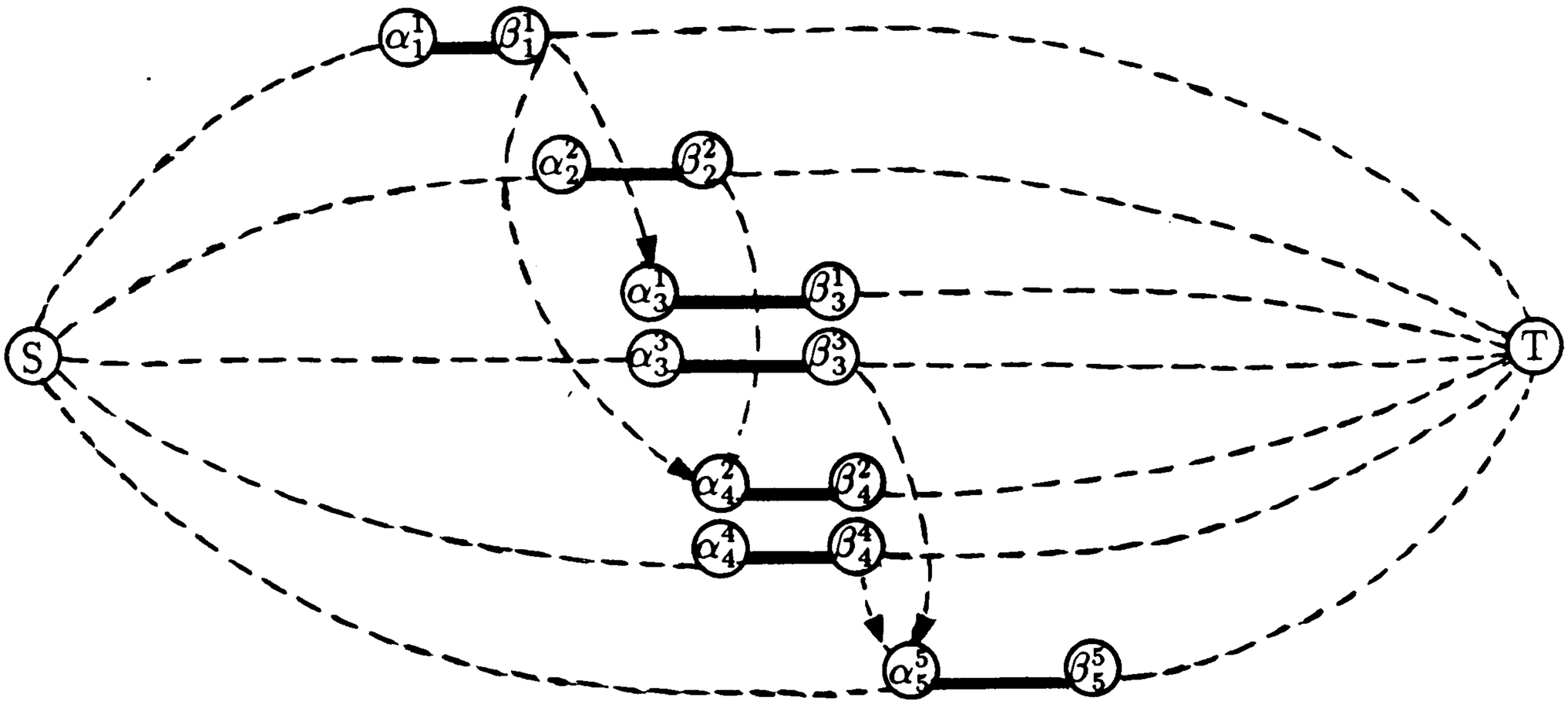
a flight in the solution, this implies that flight 2 is covered by the path starting with the flight 1. Clearly if (α_2^1, β_2^1) is included in a path starting with flight 1, it must not be included in any other path, i.e. subflight (α_2^2, β_2^2) must not be covered by any other path. Similarly, if (α_2^1, β_2^1) is not included in the path starting with flight 1, it must be included in another path (in this case the only other possibility is the path starting with flight 2). In particular note that if (α_2^1, β_2^1) cannot follow flight (α_1^1, β_1^1) because of time overlap, then flight (α_2^1, β_2^1) need not be considered at all in the graph $\widetilde{G}$. Repeating the above argument, we get the following expanded graph $\widetilde{G}$ of example A:

Figure 2-2. Expanded Graph $\widetilde{G}$ of Example A



It should be noted at this stage that, in the graph $\widetilde{G}$, the expanded flight (α_4^1, β_4^1) and (α_4^2, β_4^2) can be coalesced because they have the same successor set (in this case {T}). We choose (α_4^2, β_4^2) as the copy of (α_4, β_4) with the higher superscript to represent both of these two flights. The coalesced graph $\overline{G}$ is given in the following graph.

Figure 2-3. Coalesced Graph $\overline{G}$ of Example A



It is easily seen that the time duration of any S to T path in $\overline{G}$ is not greater than the work-duty-time. Based on this graph $\overline{G}$, the CSP can be formulated as follows *[Cheddad 1987]*.

$$\text{minimise} \quad \sum_{(\beta_i^p,\alpha_j^q)\in\overline{N}} C_{ij}X_{i_pj_q}$$

$$\text{subject to} \quad \sum_{i_p\in V_{j_q}^+} X_{i_pj_q} - \sum_{k_r\in V_{j_q}^-} X_{j_qk_r} = \begin{cases} 0, & j=1,\ldots,n,\ q\in Q_j \\ K, & j=T,\ q=1 \\ -K, & j=S,\ q=1 \end{cases} \Big\}(*)$$

$$\sum_{q\in Q_j}\ \sum_{i_p\in V_{j_q}^+} X_{i_pj_q} = 1 \quad \text{for } j=1,\ldots,n \quad \text{----------} \quad (**)$$

$$X_{i_pj_q} \in \{0,1\} \qquad \forall\ i_p, j_q \text{ such that } (\beta_i^p,\alpha_j^q)\in\overline{N}$$

$\Big\}$ PN_1

where (β_i^p,α_j^q) is the transition arc from flight i_p to flight j_q

C_{ij}= linking cost between flights i and j [cost of arc (β_i,α_j)]

$X_{i_p j_q} = 1$, if (β_i^p,α_j^q) is in optimal solution

$= 0$, otherwise

$\overline{N}$ = { set of all feasible transitions }

$V^+_{j_q} = \{\ i_p \mid (\beta_i^p,\ \alpha_j^q) \in \overline{N}\ \}$

$V^-_{j_q} = \{\ i_p \mid (\beta_j^q,\ \alpha_i^p) \in \overline{N}\ \}$

Q_j = { set of subflights (copies) for flight j }

S is the super-source

T is the super-sink

K = number of paths

n = number of flights

Note that Constraints (*) in PN_1 are the usual flow conservation constraints applied to the acyclic graph $\overline{G}$, whereas Constraints (**) in PN_1 are additional constraints expressing the fact that exactly one copy of the subflights representing each flight must be covered by the path represented by the variables X_{ij}.

How large is the problem size in the GET formulation in the worst case? The worst case arises when all the flights are disjointed and the transition times among them are very large. In this case all possible combinations of flights must be considered in the formulation. Expression (2-1a) below gives the number of variables in the GET formulation in the worst case. Note that there are $\frac{n^2+\ n}{2}$ expanded flights in total. These generate the n variables for the super-source, $\frac{n^2-\ n}{2}$ for connections between flights and $\frac{n^2+\ n}{2}$ variables for the super-sink. In Expression (2-1b) these flights also generate $\frac{n^2+\ n}{2}$ constraints for each of the expanded flights, 2 for the super-source and super-sink

and n for Constraints of type $(**)$ in PN_1. The GET formulation, therefore, produces the following number of variables V_t and number of constraints C_t in the worst case.

$$V_t = n^2 + n \quad \text{----------(2-1a)}$$

$$C_t = \frac{n^2+3n+4}{2} \quad \text{----------(2-1b)}$$

As shown in the following Table, (and comparing it to Table 1-4 of Chapter 1), the GET formulation produces much fewer constraints than the general IP formulation, but this reduced number of constraints is still too large to be solved easily in an IP. In practice it is very difficult to deal with the enormous number of constraints in problem with more than 100 flights in the worst case. Moreover algorithm GET produces a drastic increase in the number of variables.

Table 2-1. Problem Size in the GET Formulation in the Worst Case

Flights	Variables	Constraints
10	110	67
100	10,100	5,152
1000	1001,000	501,502

The great advantage of the network expansion approach to the CSP as proposed by Cheddad, is the fact that when formulation PN_1 is solved as an LP, with the integrality constraints removed, its solution is almost always naturally integer and,

therefore, rarely does one have to go into a branch-and-bound. *[Cheddad 1987]*. The great disadvantage is, as stated above, that the number of constraints (in particular) is too large for solution by standard LP codes, even for modest sized CSP's, although exploitation of the fact that some constraints are of the network-flow variety may alleviate this problem to some extent.

One thought that immediately comes to mind is to investigate the effects on the solution to problem PN_1, if constraints from that formulation one "relaxed" in same way (either ignored or surrogated). In particular a most useful investigation would concern the effects of surrogating the "flow-conservation" Constraints $(*)$ in PN_1, by summing them over the index $q \in Q_j$, for every given j. The relaxed problem $\overline{PN_1}$ would then contain only $2n$ constraints.

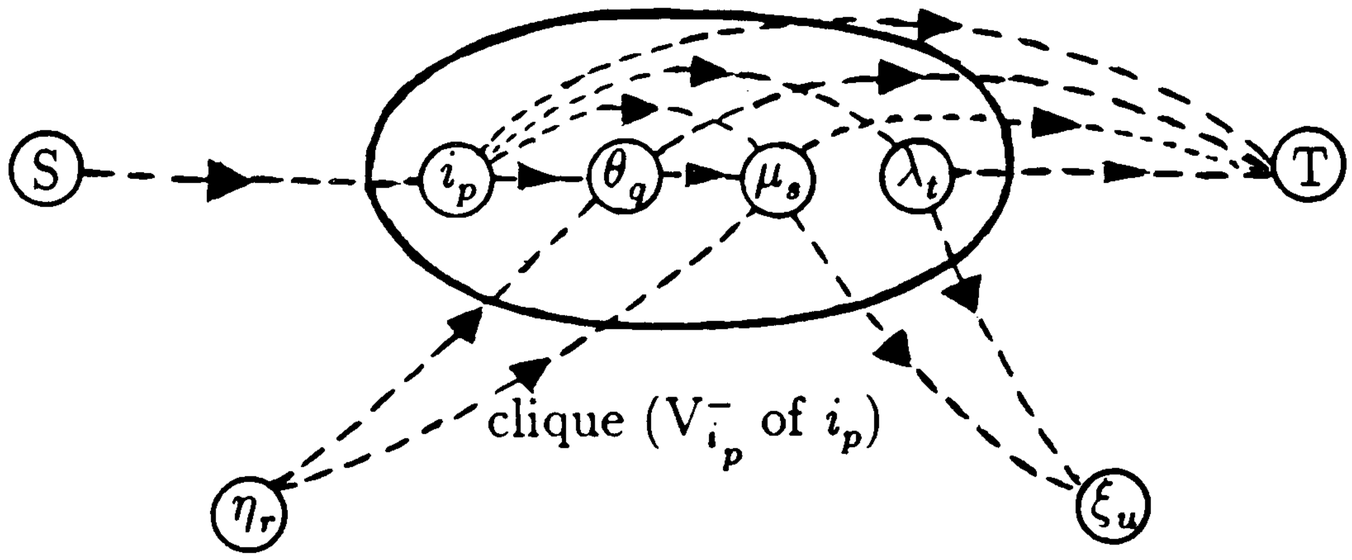
Experimental investigations that we performed on more than a dozen CSP's of medium size (up to 50 flights), show that, unfortunately, such surrogation has a dramatic effect on the solution to problem PN_1; so much so that solutions of the relaxed problem are almost always infeasible to the CSP and that (more importantly) the value of such a relaxed solution is a very poor bound to the original CSP. This conclusion, is not very surprising since, others *[Jeroslow 1977]* have also noted the fact that surrogating network-flow-conservation constraints produces dramatic disturbances to the corresponding integer convex hull of the integer program (as well as the linear program).

In view of the above comments we investigate alternative formulations to the CSP based on the same graph $\overline{G}$ but with a view towards a better behaviour of the model under constraint surrogation.

2-2. An Alternative Formulation Based on $\overline{G}$ (CAT)

Define the ***clique*** of a given flight i_p, as the set of flights $V^-_{i_p}$ (defined earlier as the set of vertices in the expansion network $\overline{G}$ which are terminal vertices of arcs emanating from the vertex β^p_i). The clique of i_p is shown in Figure 2-4 below, enclosed by solid lines. Note that in the figure below we use a vertex to represent a flight (instead of an arc as used in the derivation of graph $\overline{G}$). Also we use a subscript to denote the "copy" of that flight. In other words we now use μ_s (say) as a shorthand notation for flight $(\alpha^s_\mu, \beta^s_\mu)$.

Figure 2-4. Clique of Flight i_p



The vertices of clique $V^-_{i_p}$ have a "natural" order within the clique $i_p \prec \theta_q \prec \mu_s \prec \lambda_t$ imposed by the order of the starting times of the corresponding flights. Let us call the order of a given vertex $j_q \in V^-_{i_p}$ by $R_{i_p}(j_q)$. This order imposes an order on the subsets:

$$\S_1 \subset \S_2 \subset \S_3 \cdots\cdots \subset \S_m$$

of the clique containing i_p (where $m = |V_{i_p}^-|$).

These subsets, for the above example, are:

$$\mathbb{S}_1 = \{ i_p \}$$

$$\mathbb{S}_2 = \{ i_p, \theta_q \}$$

$$\mathbb{S}_3 = \{ i_p, \theta_q, \mu_s \}$$

$$\mathbb{S}_4 = \{ i_p, \theta_q, \mu_s, \lambda_t \}$$

The family of such subsets for the clique of i_p will be denoted by F_{i_p}(in the above example $F_{i_p} = \{\mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \mathbb{S}_4\}$.

An alternative formulation of the CSP based on the network expansion is now stated and explanations of it are given in separate sections that follow.

$$\text{minimise} \quad \sum_{(\beta_i^p, \alpha_j^q) \in \overline{N}} C_{ij} X_{i_p j_q} + C_K X_K \qquad \text{(2-2a)}$$

$$\text{subject to} \quad \sum_{q \in Q_j} \sum_{i_p \in V_{j_q}^+} X_{i_p j_q} = 1 \quad \text{for } j = 1, \ldots, n \qquad \text{(2-2b)}$$

$$= X_K \quad \text{for } j = T$$

$$\sum_{q \in Q_j} \sum_{k_r \in V_{j_q}^-} X_{j_q k_r} = 1 \quad \text{for } j = 1, \ldots, n \qquad \text{(2-2c)}$$

$$= X_K \quad \text{for } j = S$$

$$\sum_{\substack{k_r \in \mathbb{S} \\ j_q \in (V_{k_r}^+ \setminus \mathbb{S}) \cup \{S\}}} X_{j_q k_r} - \sum_{\substack{k_r \in \mathbb{S} \\ j_q \in (V_{k_r}^- \setminus \mathbb{S}) \cup \{T\}}} X_{k_r j_q} = 0 \qquad \text{(2-2d)}$$

$$\forall\, \mathbb{S} \in F_{i_p}, \quad i_p = 1, \ldots, n$$

$$X_{i_p j_q} \in \{0,1\} \quad \forall\, i_p, j_q \text{ such that } (\beta_i^p, \alpha_j^q) \in \overline{N} \qquad \text{(2-2e)}$$

} PN_2

where (β_i^p,α_j^q) is the transition arc from flight i_p to flight j_q

C_{ij}= linking cost between flights i and j [cost of arc (β_i,α_j)]

C_K = crew cost per unit crew

$X_{i_p j_q}$ = 1, if (β_i^p,α_j^q) is in optimal solution

= 0, otherwise

X_K = number of crew to work in the flight schedule

$\overline{N}$ = { set of all feasible transitions }

$V_{j_q}^+ = \{\ i_p \mid (\beta_i^p,\ \alpha_j^q) \in \overline{N}\ \}$

$V_{j_q}^- = \{\ i_p \mid (\beta_j^q,\ \alpha_i^p) \in \overline{N}\ \}$ (called the clique of j_q)

Q_j = { set of subflights (copies) for flight j }

S is the super-source

T is the super-sink

n = number of flights

Note that:

(i) In this formulation the objective function (2-2a) includes an additional term discussed separately in Section 2-2-1.

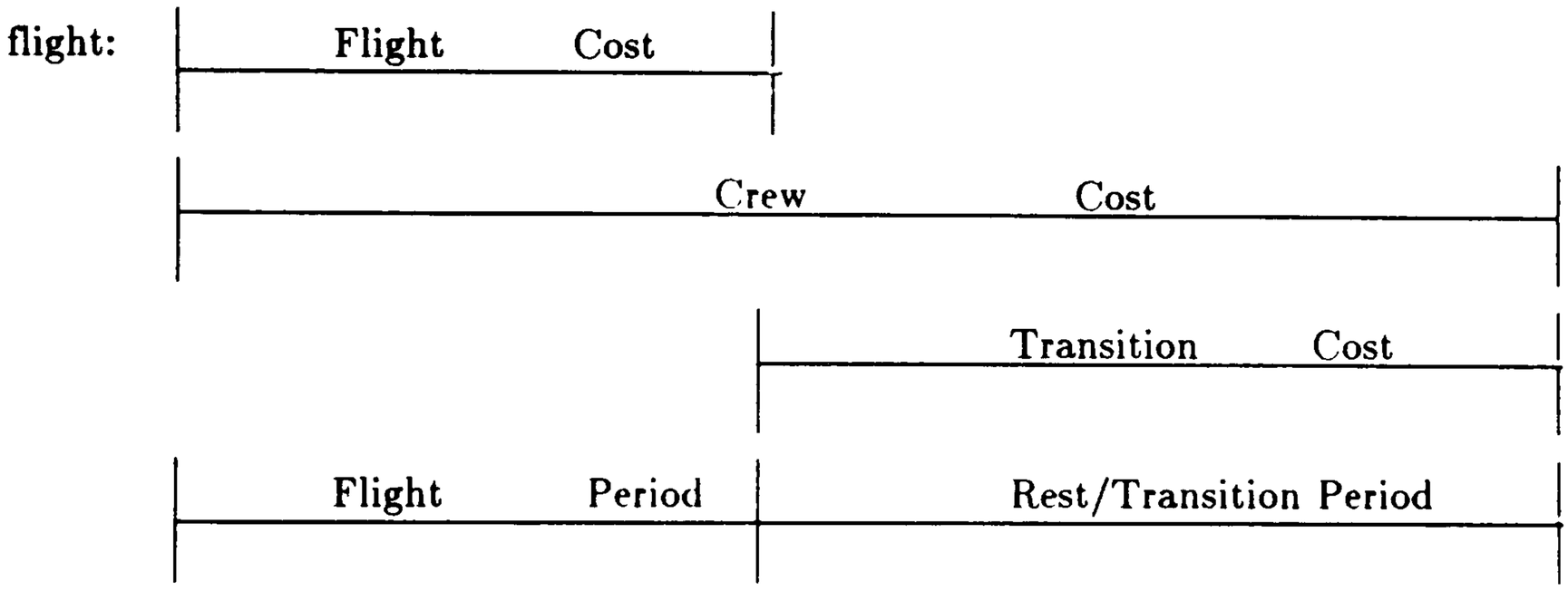
(ii) Constraints (2-2b) and (2-2c) express the fact that every flight must be covered by the set of paths in the solution. They are discussed separately in Section 2-2-2.

(iii) Constraints (2-2d) are generalised network-flow conservation constraints and are easily shown to be equivalent (in an IP sense) to the classical network-flow conservation Constraints (*) of formulation PN_1. They are discussed separately in Section 2-2-3.

2-2-1. Objective Function with Crew Costs

In the practical CSP there are three kinds of costs to be considered for obtaining an optimal solution. They are the flight costs (FC), the crew costs (CC) and the transition costs (TC) as shown in the following Figure 2-5.

Figure 2-5. Cost Structure of CSP



The flight cost is the cost related to the flight itself, such as the cost for fuel and insurance of the plane and so on. This cost usually is approximately proportional to the total flight time of the airplane. In the CSP formulation we do not need to consider this cost since the summation of these flight costs is always constant, because all the flights are covered once and only once.

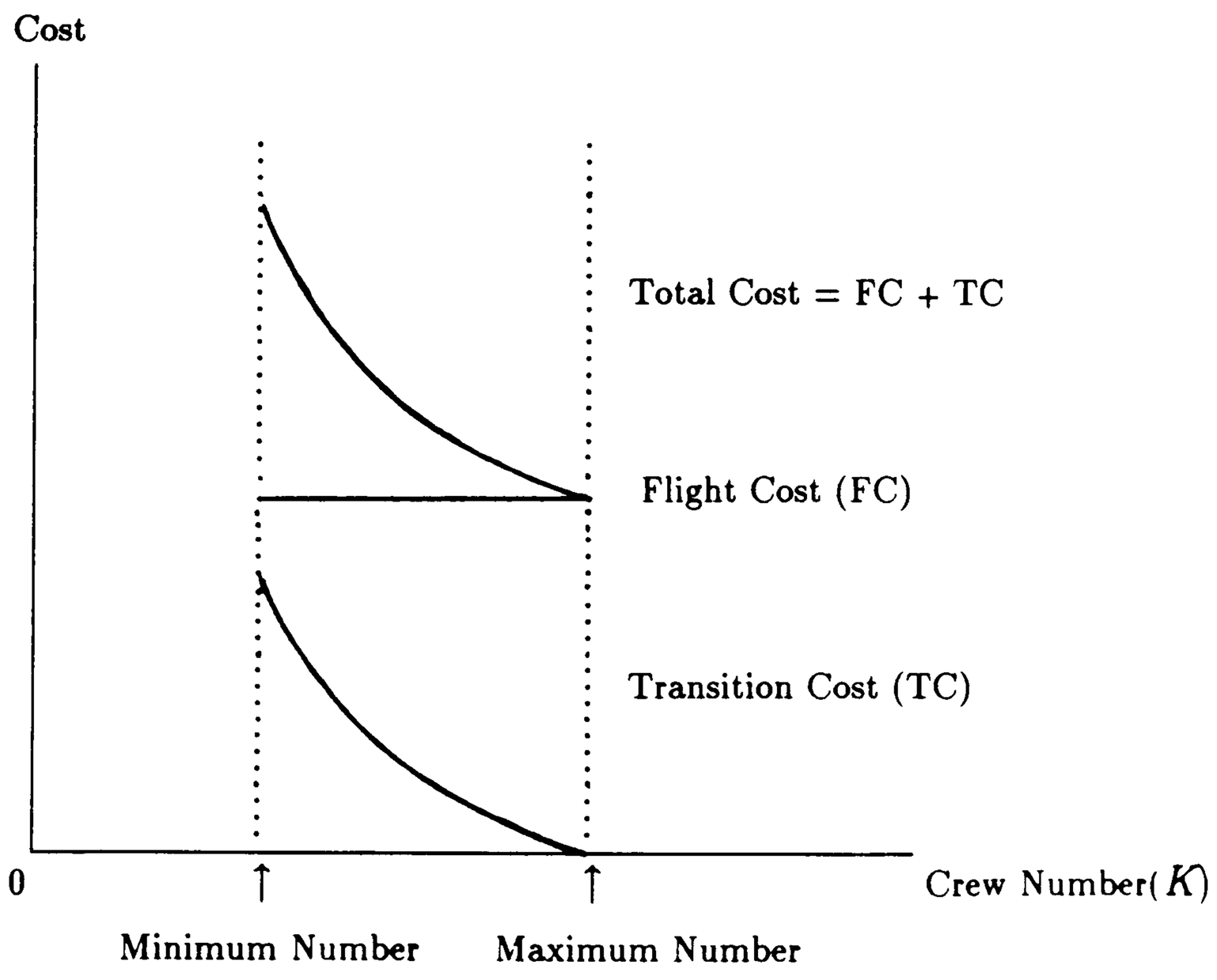
The crew cost is, as its name denotes, the cost for the crew such as the salary, bonus etc. In fact this cost has nothing to do with the flight of the airplanes. In other

words this cost must be paid even if the crew is not working at all. This cost is assumed to be directly proportional to the number of crews.

Lastly the transition cost is the cost for moving the crews from one airport to another in order to enable them to work on the next flight. This cost, which is usually proportional to the distance and transition time between two airports, includes the transportation cost, the cost of lodging and so on.

In the CSP models, so far developed, the crew cost is not taken into consideration. Only the transition cost has been considered to get the optimal solution. The crew costs are assumed to be constant (implicitly), because the number of crews is assumed to be a given constant K. However, as K varies the costs vary as shown in Figure 2-6.

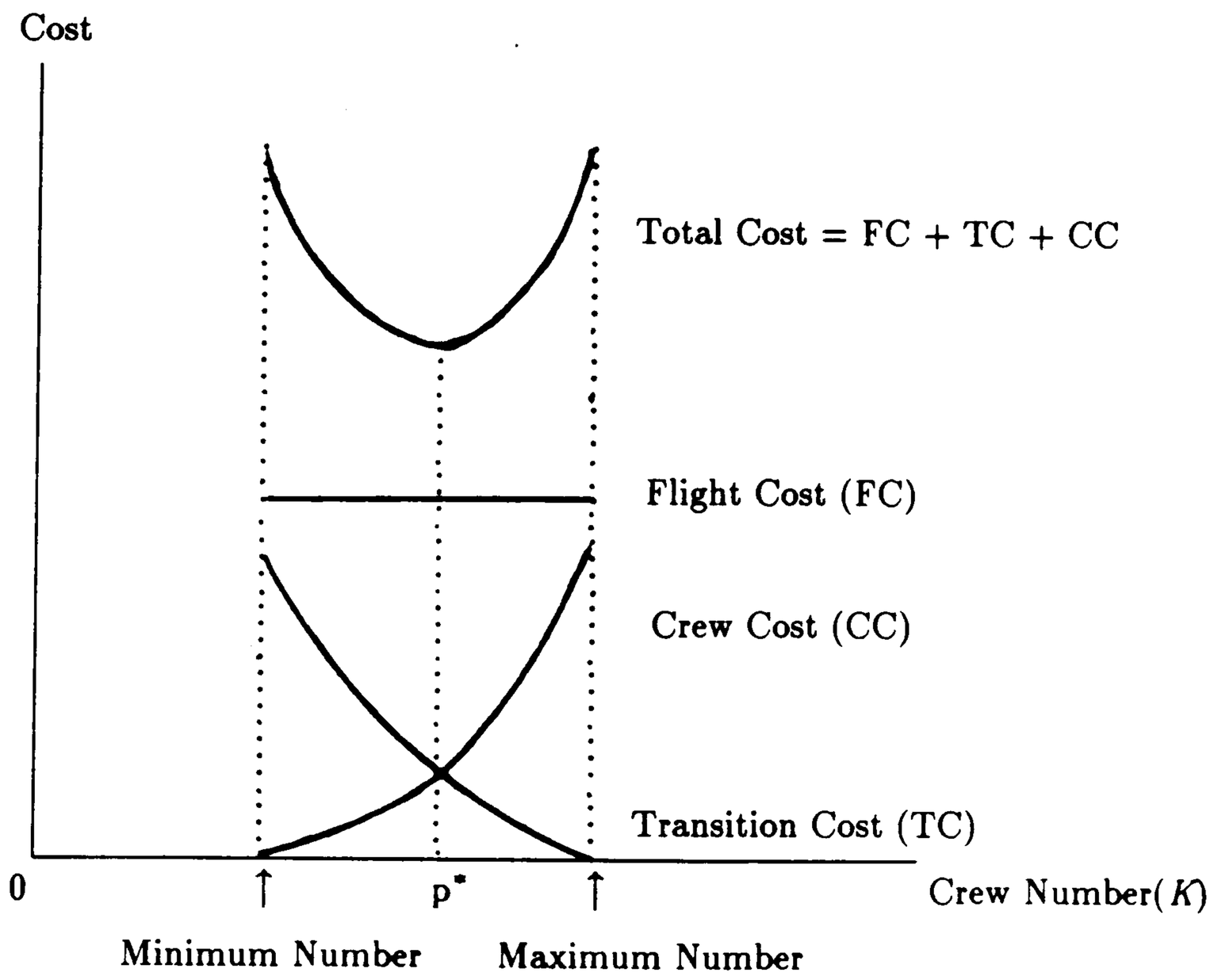
Figure 2-6. CSP Cost Structure without Crew Cost



Obviously, by solving the simpler model many times once for each value of K one can obtain the total cost picture as shown in Figure 2-7 below. However, this can be a very time-consuming process and it would obviously be better to incorporate the crew costs into the CSP model itself as shown in the objective function of Equation (2-2a).

In the CSP, all flights are to be covered exactly once, so the summation of flight costs is always constant regardless of the solution. The first term of Equation (2-2a) represents the transition costs for the crews whereas the second term represents the crew costs. Note that all crews are assumed to have the same cost coefficient C_K.

Figure 2-7. CSP Cost Structure with Crew Cost



2-2-2. Flight-covering Constraints in the CAT Formulation

There are four kinds of "flight-covering" constraints, represented by (2-2b) and (2-2c) constraints in the CAT formulation. The first set of Constraints (2-2b) are for the inflows to each flight and to the super-sink. And the other two (2-2c) are for the outflows from each flight and from the super-source. Note that Constraints (2-2b) and (2-2c) do **not** imply flow conservation in $\overline{G}$ because they are summed over Q_j.

The sum of the inflows from flights to a specific flight (j) must be 1 if flight j is to be covered. Thus:

$$\sum_{q \in Q_j} \sum_{i_p \in V^+_{j_q}} X_{i_p j_q} = 1 \qquad \text{for } j = 1, \ldots, n \text{ ---------- (2-3)}$$

The inflows from all flights to the super-sink T must be the same as the number of crews X_K. Thus, for j=j_1=T and assuming Q_T=$\{1\}$:

$$\sum_{q \in Q_j} \sum_{i_p \in V^+_{j_q}} X_{i_p j_q} = X_K \text{ ---------- (2-4)}$$

The sum of outflows from (the set of copies of) a given flight j must also be 1, if flight j is to be covered. Thus:

$$\sum_{q \in Q_j} \sum_{k_r \in V^-_{j_q}} X_{j_q k_r} = 1 \qquad \text{for } j = 1, \ldots, n \text{ ---------- (2-5)}$$

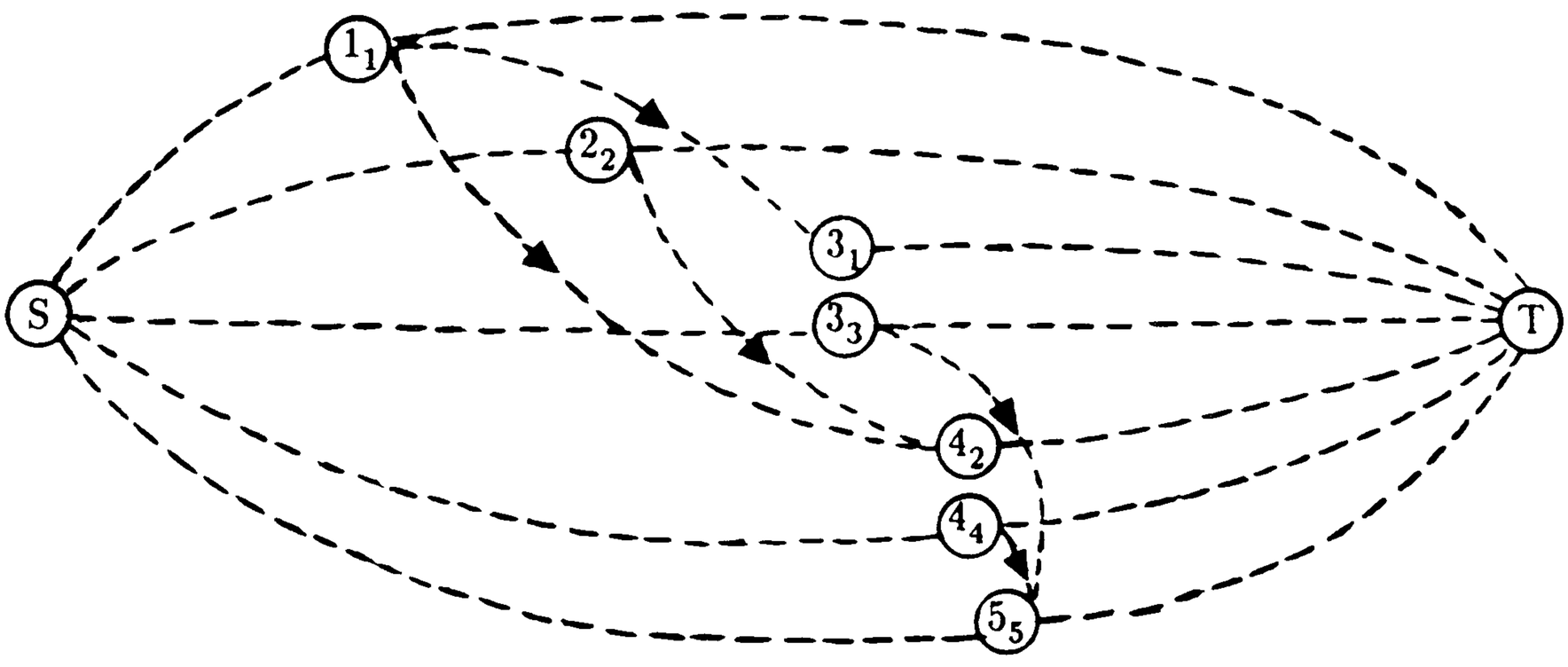
The outflows from the super-source S to all the flights must be the same as the number of crews. Thus, for $j=j_1=$S and assuming $Q_S=\{1\}$:

$$\sum_{q \in Q_j} \sum_{k_r \in V^-_{j_q}} X_{j_q k_r} = X_K \qquad \text{(2-6)}$$

An Example of Flight-covering Constraints

Consider again the coalesced graph $\overline{G}$ of example A shown in Figure 2-8 which has exactly the same structure as Figure 2-3 except that flights are represented by vertices rather than arcs as explained earlier. (We will use this simplified expression in the remaining of the thesis, as long as there arises no confusion.)

Figure 2-8. Network $\overline{G}$ of Example A



The above graph $\overline{G}$ of example A produces the following network flow constraints. Note that we do not need to write the "inflow" constraints for flights 1 and 2 since they have only one inflow arc from S which ensures that the flights are covered simply by writing their outflow constraints. Similarly, we do not need to write the "outflow" constraints for flight 5 since its covering is guaranteed by the inflow constraint by itself.

$$X_{1_1 3_1} + X_{S 3_3} = 1$$
$$X_{1_1 4_2} + X_{2_2 4_2} + X_{S 4_4} = 1$$
$$X_{3_3 5_5} + X_{4_4 5_5} + X_{S 5_5} = 1$$
$$X_{1_1 T} + X_{2_2 T} + X_{3_1 T} + X_{3_3 T} + X_{4_2 T} + X_{4_4 T} + X_{5_5 T} = X_K$$
$$X_{1_1 T} + X_{1_1 3_1} + X_{1_1 4_2} = 1$$
$$X_{2_2 T} + X_{2_2 4_2} = 1$$
$$X_{3_1 T} + X_{3_3 T} + X_{3_3 5_5} = 1$$
$$X_{4_2 T} + X_{4_4 T} + X_{4_4 5_5} = 1$$
$$X_{S 1_1} + X_{S 2_2} + X_{S 3_3} + X_{S 4_4} + X_{S 5_5} = X_K$$

2-2-3. Flow-conservation Constraints in the CAT Formulation

The CAT formulation PN_2 has replaced the normal vertex-based flow-conservation constraints given by $(*)$ in problem PN_1 with the set-based flow-

conservation constraints given by (2-2d) in problem PN_2. It is easy to show that the two sets of constraints are equivalent in the IP sense (in the presence of the other constraints in their respective formulations). It is also simple to show that the number of constraints in formulation PN_2 is the same as the number of constraints in formulation PN_1. However, the two formulations are by no means equivalent in their relaxations.

Firstly, we would like to note that just as for problem PN_1, the solutions to problem PN_2 obtained by relaxing the integrality constraints are almost always integer. A most-important difference, however, arises when the flow-conservation constraints of PN_2 are surrogated in order to reduce their number, and enable LP to be used for providing a bound.

Let us, for example, sum Constraints (2-2d) in PN_2 over the index $k_r \in \mathbb{S}$, $\forall \mathbb{S} \in F_{i_p}$ for a given i_p so as to create just n constraints of this type. We will first illustrate the process by an example based on the clique of flight i_p shown in Figure 2-4.

Consider Constraints (2-2d) for i_p shown in Figure 2-4:

for $\mathbb{S} = \{ i_p\}$ we have:

$$X_{Si_p} = X_{i_p\theta_q} + Xi_p\mu_s + X_{i_p\lambda_t} + X_{i_pT} \text{ ---------- (2-7)}$$

for $\mathbb{S} = \{ i_p, \theta_q\}$ we have:

$$X_{Si_p} + X_{\eta_r\theta_q} = Xi_p\mu_s + X_{i_p\lambda_t} + X_{i_pT} + X_{\theta_q\mu_s} + X_{\theta_qT} \text{ ---------- (2-8)}$$

for $\mathbb{S} = \{ i_p, \theta_q, \mu_s \}$ we have:

$$X_{Si_p} + X_{\eta_r\theta_q} + X_{\eta_r\mu_s} = X_{i_pT} + X_{i_p\lambda_t} + X_{\theta_qT} + X_{\mu_sT} + X_{\mu_s\xi_u} \text{ ---------- (2-9)}$$

and finally for $\mathbb{S} = \{ i_p, \theta_q, \mu_s, \lambda_t \}$ we have:

$$X_{Si_p} + X_{\eta_r\theta_q} + X_{\eta_r\mu_s} = X_{i_pT} + X_{\theta_qT} + X_{\mu_sT} + X_{\mu_s\xi_u} + X_{\lambda_tT} + X_{\lambda_t\xi_u} \text{ --------(2-10)}$$

Surrogate Flow-conservation Constraints

Surrogating Constraints (2-7) to (2-10) above (with all weight constants=1), we construct the final time constraint for the clique of flight i_p as follows:

$$\begin{aligned} 4X_{Si_p} + 3X_{\eta_r\theta_q} + 2X_{\eta_r\mu_s} = \; & 4X_{i_pT} + X_{i_p\theta_q} + 2Xi_p\mu_s + 3Xi_p\lambda_t + \\ & 3X_{\theta_qT} + X\theta_q\mu_s + 2X_{\mu_sT} + 2X_{\mu_s\xi_u} + \\ & X_{\lambda_tT} + X_{\lambda_t\xi_u} \text{ ----------(2-11)} \end{aligned}$$

As can be seen from Figure 2-4 and from the above Expression (2-11), the coefficients of each variables of the form $X_{\alpha j}$ or $X_{j\alpha}$ where $j \in V^-_{i_p}$ and $\alpha \notin V^-_{i_p}$, is given by: $[\ |V^-_{i_p}|+R_{i_p}(j)-1\ ]$, whereas the coefficient of each variable of the form X_{jk} where j, $k \in V^-_{i_p}$, is given by $[\ R_{i_p}(k)-R_{i_p}(j)\ ]$.

Thus, surrogating Constraints (2-2d) by summing over all $\mathbb{S} \in F_{i_p}$ for a given i_p

produces the constraints:

$$\sum_{\mathbb{S} \in F_{i_p}} \sum_{\substack{k_r \in \mathbb{S} \\ j_q \in (V_{k_r}^{+} \setminus \mathbb{S}) \cup \{S\}}} X_{j_q k_r} = \sum_{\mathbb{S} \in F_{i_p}} \sum_{\substack{k_r \in \mathbb{S} \\ j_q \in (V_{k_r}^{-} \setminus \mathbb{S}) \cup \{T\}}} X_{k_r j_q} \quad \text{for } i_p = 1,\ldots, n \text{ ----- (2-12)}$$

where all coefficients are small integer values.

2-3. Non-integer Solutions

In Section 2-2 we gave an alternative formulation of the CSP and in Section 2-2-3 we have given a relaxed formulation based on surrogating Constraints (2-2d) to produce problem $\overline{PN_2}$ which is problem PN_2 with Constraints (2-12) instead of Constraints (2-2d).

We will now discuss the problem $\overline{PN_2}$. In most cases we get the optimal solution (feasible to original problem) with integer values directly from the LP solution to problem $\overline{PN_2}$. Among 250 test problems we obtained 247 integer optimal solutions and 3 non-integer solutions from the LP solution to $\overline{PN_2}$. Even in the 3 cases where solution were not integer a heuristic based on the Transportation Problem (and which is described below) provided a guaranteed optimal answer for 2 of them and provided an answer for the other one problem which could not be confirmed by the heuristic as optimal at that stage but which was finally shown to be optimal anyway by a branch & bound procedure described in Section 2-4. Note also that it is possible for the relaxed problem $\overline{PN_2}$ to have an integer solution which is infeasible to the original CSP. In that case (which has never occurred in our experiments) we could have a controlled

incremental de-surrogation of the surrogated Constraints (2-12) until a feasible solution is obtained. From previous experience with problems exhibiting this type of behaviour (i.e. where the LP solution to a relaxed problem is almost always integer and optimal to the original problem) we suspected that even for the few cases where the solution was fractional, the likelihood was that an alternative integer solution existed with the same value as the LP. We, therefore, decided that it was worthwhile to search for such an (integer) alternative optimum. In particular, we decided that it would be worthwhile to try to produce an integer solution by means of solving a Transportation Problem, formulated on those variables which had fractional values in the LP solution to $\overline{\mathrm{PN}_2}$. We will now describe this heuristic search for an equivalent integer solution.

Let us introduce a new set of variables Ξ_{ij} defined from the variables $X_{i_p j_q}$ as follows.

$$\Xi_{ij} = \sum_{p \in Q_i} \sum_{q \in Q_j} X_{i_p j_q} \qquad \text{----------(2-13)}$$

Then Constraints (2-2b) in problem $\overline{\mathrm{PN}_2}$ imply {in the presence of the integrality Constraints (2-2e)} the constraints:

$$\sum_{i \in B_j} \Xi_{ij} = 1 \qquad j=1,\ldots,n \qquad \text{----------(2-14)}$$

where $\quad B_j = \{\, i \mid \; i_p \in V^+_{j_q} \text{ for any } q \in Q_j,\, p \in Q_i \,\}$

Similarly, Constraints (2-2c) imply constraints:

$$\sum_{i \in D_j} \Xi_{ji} = 1 \qquad j=1,\ldots,n \text{ ----------(2-15)}$$

where $D_j = \{ i \mid i_p \in V^-_{j_q}$ for any $q \in Q_j$, $p \in Q_i \}$

Note that without the imposition of integrality Constraints (2-2b) and (2-2c) do not imply Constraints (2-14) and (2-15).

Constraints (2-14) and (2-15) define a Transportation Problem(TP) polyhedron. We can now construct an TP using those variables $X_{i_p j_q}$ which are fractional in the LP solution to problem $\overline{PN_2}$ to define new variables Ξ_{ij} which are then used in an TP as follows:

Suppose that some variable $X_{i_p j_q}$ takes a non-integer value in the LP solution to problem $\overline{PN_2}$ and that the corresponding variable Ξ_{ij} defined by (2-13) takes the value V_{ij}. It is clear that we wish Ξ_{ij} to take the value 0 or 1 and the increase in the cost of the objective when Ξ_{ij} is increased from V_{ij} to 1 is given by:

$$\Phi_{ij} = (1 - V_{ij}) \cdot C_{ij} \text{ ----------(2-16)}$$

The TP is now defined by the cost coefficients Φ_{ij} in (2-16) and the TP Constraints (2-14) and (2-15). If the solution to the TP is given by Ξ^*_{ij}, it represents only that part of the LP solution to $\overline{PN_2}$ which did not have integer values for the variables $X_{i_p j_q}$, and, therefore, must be combined with those variables $X_{i_p j_q}$ which had integer values in the LP solution to $\overline{PN_2}$ in order to produce (possibly) a feasible solution to $\overline{PN_2}$. This combination of Ξ^*_{ij} with the integer part of the LP solution to $\overline{PN_2}$ involves firstly "translating" the $X_{i_p j_q}$ of that part of the LP solution via a

transformation similar to that given in (2-13) to produce "equivalent" variables Ξ_{ij} involving "flights" as suffices rather than the subflights (or copies) that are implied by the suffices i_p and j_q of the variable $X_{i_p j_q}$ in the expanded graph $\overline{G}$.

Clearly, it is quite possible that when Ξ^*_{ij} is combined with the integer part of the solution to $\overline{PN}_2$ in this way, the result may not be feasible. However in our tests we never encountered this problem, so the TP solution could always be used to provide a heuristic feasible solution to the CSP in the 3 cases (out of 250) when the solution to $\overline{PN}_2$ was not naturally integer.

Moreover, if the improved solution from the Transportation Problem produces a total cost which is the same as the original LP cost, it must be the optimal solution. In case the cost by the Transportation Problem is greater than the cost of the LP, we must use a branch and bound algorithm to get the optimum answer. In our test problems these upper bounds obtained by the TP approach are within 3% from the final optimum cost obtained by a tree search.

Example

Consider the following problem which has 19 flights and 5 airports with a work-duty-time of 600 minutes and a crew cost 600. As previously defined, we arrange the flights in ascending order of their starting times.

Table 2-2. Flights in Example

Flight No	Starting Time	Finishing Time	Starting Place	Finishing Place
1	33	208	A	E
2	114	325	C	B
3	169	230	D	C
4	260	445	B	D
5	337	470	E	A
6	375	489	A	B
7	463	599	D	A
8	555	653	B	D
9	617	830	E	C
10	652	833	B	E
11	740	960	E	A
12	862	1079	D	E
13	912	1056	C	D
14	938	1213	B	C
15	1055	1318	A	B
16	1132	1204	B	A
17	1223	1338	A	D
18	1244	1476	D	B
19	1335	1592	E	C

This CSP covers 5 airports. The transition times between the airports are given as follows. We assume these transition times are not symmetric which is not unusual in the real world.

Table 2-3. Transition Times amongst Airports

Place	A	B	C	D	E
A	0	259	287	66	212
B	263	0	56	171	235
C	112	225	0	137	88
D	84	123	105	0	67
E	134	225	258	123	0

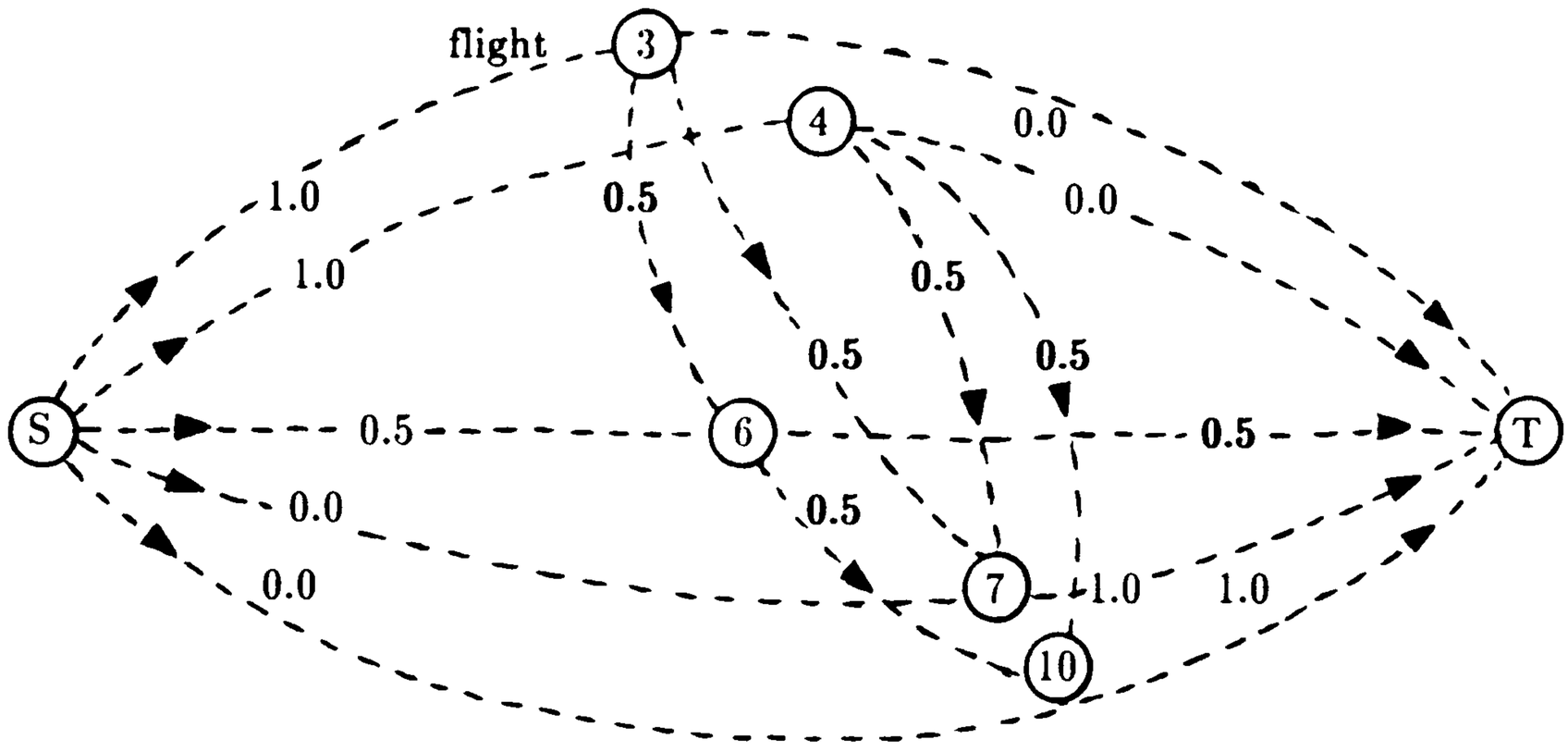
The corresponding transition costs of this problem are shown in the following Table. The costs are also assumed not to be symmetric.

Table 2-4. Transition Cost amongst Airports

Place	A	B	C	D	E
A	0	41	70	58	46
B	248	0	40	30	233
C	81	177	0	105	37
D	46	56	72	0	34
E	80	205	215	79	0

The above problem produces a non-integer solution to problem $\overline{\mathrm{PN}}_2$ with the LP value 6717.9. The non-integer part of the LP solution translated into variables $\Xi_{i,j}$ is shown in Figure 2-9.

Figure 2-9. Non-integer Part of LP Solution to Problem $\overline{\mathrm{PN}}_2$



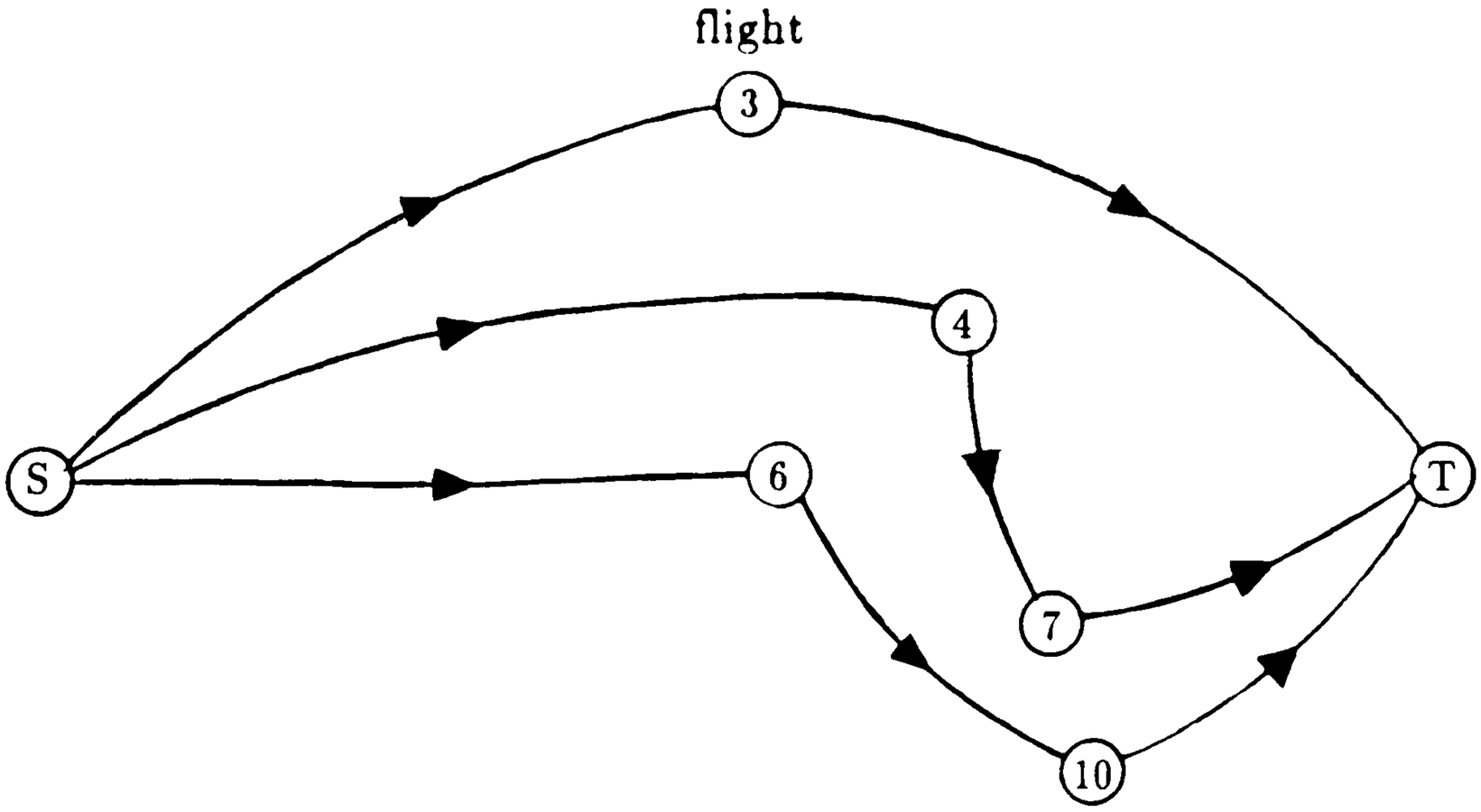
The Transportation Problem cost matrix from the non-integer variables is as follows, where ∞ is inserted for infeasible transitions between flights.

Table 2-5. Transportation Cost Matrix of Example

Flight	3	4	6	7	10	T
3	∞	∞	40.5	52.5	∞	300.0
4	∞	∞	∞	0.0	28.0	300.0
6	∞	∞	∞	∞	0.0	150.0
7	∞	∞	∞	∞	∞	0.0
10	∞	∞	∞	∞	∞	0.0
S	0.0	0.0	150.0	300.0	300.0	0.0

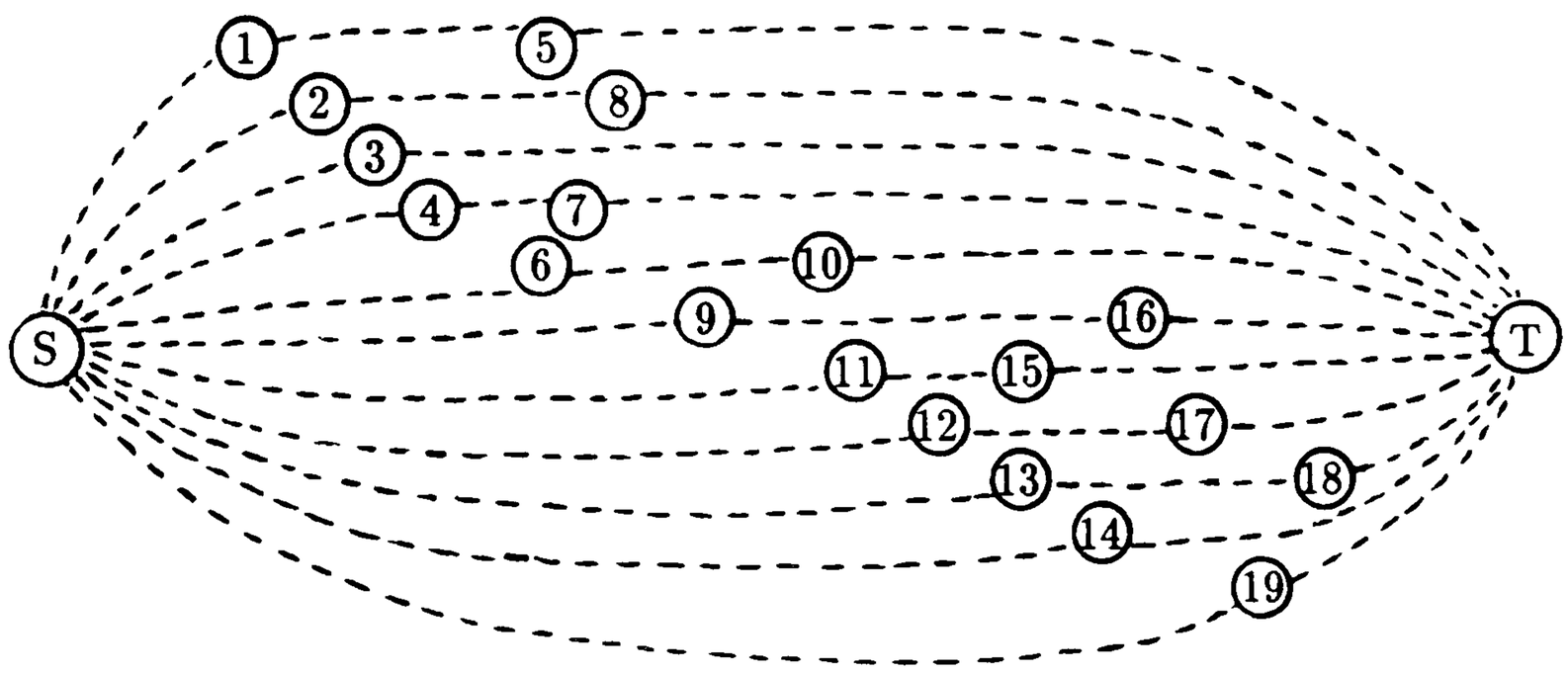
The TP solution is shown in figure 2-10 below.

Figure 2-10. The TP Solution of Example



This solution when combined with the integer part of the LP solution to $\overline{PN}_2$ provides a feasible solution to the CSP given by the path:

Figure 2-11. The Paths of TP Solution of Example



However, for this problem, the upper bound is 6897 which is larger than the initial LP cost and, therefore the above solution may not be optimal. We will discuss the tree search procedure to get the optimum solution in the next section.

2-4. A Tree Search Procedure

We use a tree search algorithm to get the optimal solutions, if the integer

feasible solution obtained by the TP is not proven to be optimal. We must note once again at this stage that out of 250 problems tested, 247 were solved optimally by the LP solution to the surrogated problem $\overline{\mathrm{P\,N}}_2$, 2 were solved optimally (and proven so) by the TP, and for one a feasible solution was found by the TP but not proven optimal by the TP value. Thus, the fact that only 1 problem out of remained, means that the tree search in this section is very rarely used, and for this reason we describe here a generic tree search procedure, without any sophisticated features. This approach has been used for virtually all the combinatorial problems *[Fisher et al 1975, Garfinkel 1979]*. Its concept is so simple as to allow room for the inclusion of many rules which can be devised for any particular instance of problems *[Gouvras 1987]*. The general process has been given several names by different authors, such as combinatorial programming *[Rossman et al 1958]*, branch and bound *[Little et al 1963]*, backtrack programming *[Colomb et al 1965]* and implicit enumeration *[Geoffrion 1967]*. There are mainly two search strategies that can be adopted, namely '*depth-first*' which partitions the last-generated subproblem and '*breadth-first*' which partitions the first-generated subproblem. The former requires a smaller storage space with respect to the breadth first type.

The main ingredients of a tree search algorithm are the branching rules to get subproblems, the lower bounding method on these subproblems, the upper bounding method by a heuristic algorithm and the search strategy to choose next subproblem to examine. This algorithm solves a combinatorial problem by repeatedly partitioning the solution set into smaller subsets to produce subproblems that are manageable. By solving a relaxation of a subproblem, a lower bound is obtained on the optimal solution to that subproblem. The solution is produced by eliminating subproblems which are infeasible solution or have a lower bound greater than the upper bound. General procedures of the tree search algorithm can be found in: *[Garfinkel et al 1972, Balas 1975, Beale 1979, Spielberg 1979]*. As mentioned above, it is not our intention to present here a

special tree search algorithm in any new light. The generic tree search algorithm is given as follows.

Step 1. Initialisation

Begin with an initial list of problem nodes L=$\{S_0\}$,

Upper bound $\overline{Z}$ =UB(obtained by a heuristic)

Step 2. Select subproblem

If L=ϕ, go to step 6

Otherwise choose a problem S from L, L←L\S, and go to step 3

Step 3. Investigation of subproblem

If S is infeasible or its lower bound LB(S) $\geq \overline{Z}$, go to step 2

If a feasible optimal solution for S is found, go to step 4

Else go to step 5

Step 4. Record better solution

If the value for problem S $<\overline{Z}$, update $\overline{Z}$ and record the new best answer

Go to step 2

Step 5. Branching

Split problem S into smaller subproblems S_1, S_2, $\cdots$ which are equivalent to S

(by fixing variables, for example)

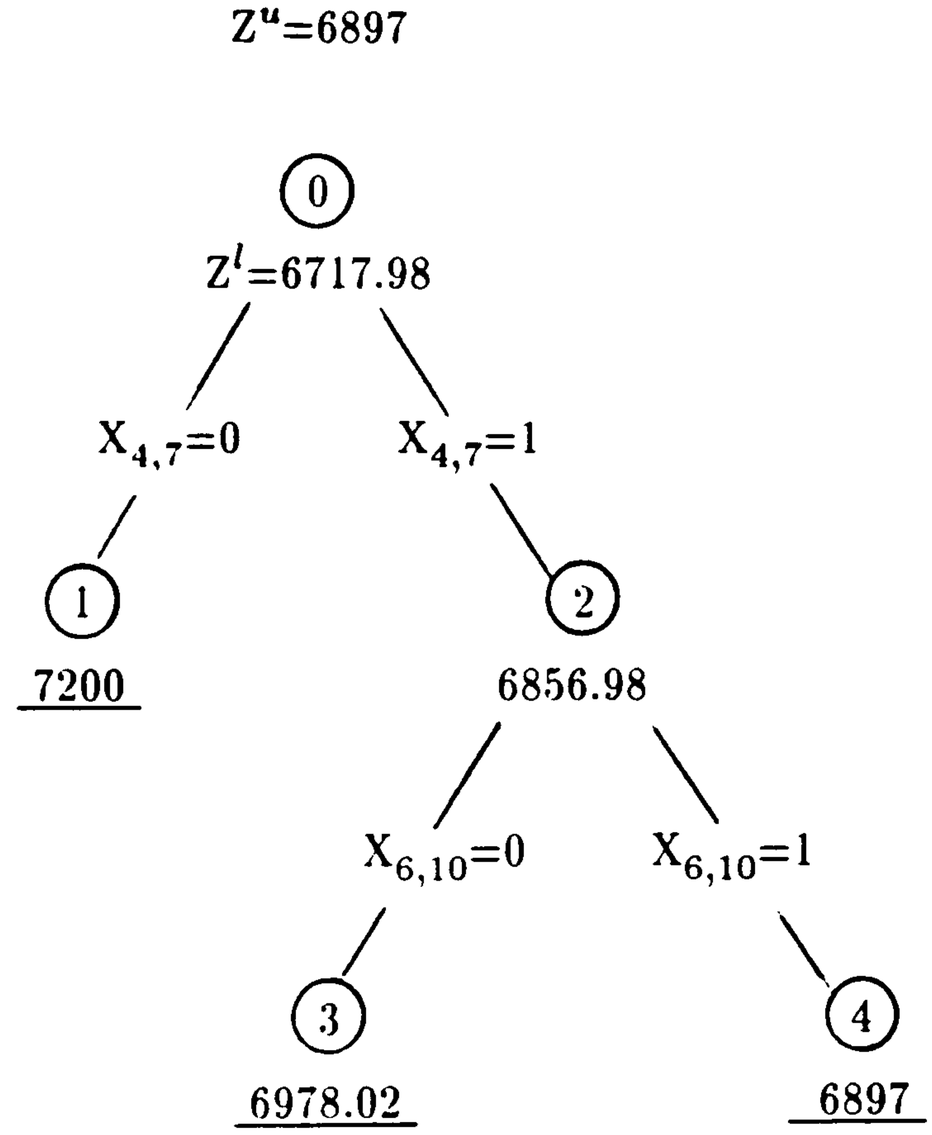
Add S_1, S_2, $\cdots$ into list L and go to step 2

Step 6. Termination

The best answer so far is the optimal answer

In our tree search procedure we used the TP solution for the upper bound UB, and the LP solution value for problem $\overline{\mathrm{PN}}_2$ as the lower bound on the subproblems. We implemented a depth-first tree search (i.e. applied a LIFO procedure to the subproblem list L in the above algorithm). The branching rule used was also very simple. We choose that variable to branch on which had a value closest to an integer (0 or 1).

Figure 2-12. Tree Search Diagram of Problem A



The tree search diagram shown in Figure 2-12 refers to the example problem of

Section 2-3-1 which was the unique problem not solved prior to the tree search. The tree has three terminal, all of them fathomed by the lower bound. So the TP-produced upper bound, 6897 is proven to be the optimal solution.

2-5. Computational Results

The algorithms GET and CAT described in this chapter were coded in FORTRAN 77 (using the Microsoft Fortran 4.1 compiler) and tested on an IBM PC/AT. The performance of the algorithm was investigated on 250 problems which were randomly generated from 6 to 30 flights. The starting times of the flights were randomly generated in the range [0−24 hours]. The number of airports involved in each problem varied (randomly) from an average of 5 for the smaller problems to an average of 10 for larger ones. Flights were generated between airports and the flight durations were sampled uniformly from the range [1.00D−2.00D] where D is a time proportional to the distance between the starting and finishing airports for that flight. This sampling introduces a degree of realism in the CSP generation since in practice different aircraft would cruise at different speeds between the same two destinations. In addition crew costs were chosen so that the total cost of the crews was approximately the same as the total transition cost in the CSP.

The performance of the algorithms was assessed from two points of view between the original algorithm GET and the improved algorithm CAT. Firstly we are interested in how efficiently the CAT algorithm works for CSP's. Secondly we wish to compare the

two algorithm GET and CAT in terms of computing time, for various problem size.

In the Table 2-6, we review the general performance of the CAT algorithm. Among the 25 hardest test problems from the list of 250, we could get the optimal integer solutions for 22 problems directly by the initial LP solution to the surrogated relaxed problem $\overline{PN}_2$. In the next step, all the unsolved problems could be solved by the Transportation Problem and the tree search. The first lower bounds of these problems lie within 3% from their optimal solutions. The maximum number of nodes produced in the tree search algorithm is 5. Note, however, that only one problem is involved here. As a matter of interest, we would like to note here, that for ten problems all with 30 flights, the average number of constraints in the CAT formulation was 75 with a maximum number of 87 and a minimum number of 64. We compare also the computing times of algorithms GET and CAT, on 250 problems, 10 for each size from 6 to 30 flights. The results are summarised in Table 2-7 which shows that the computing time of CAT is about half of that required by GET for almost every case.

Table 2-6. General Performance of CAT Algorithm

Problem	Flight #	LP[1]	TP[2]	TS[3]	Node[4]	Error[5]
1	6	200.0	-	-	-	0.00
2	7	250.0	-	-	-	0.00
3	8	200.0	-	-	-	0.00
4	9	250.0	-	-	-	0.00
5	10	4233.0	-	-	-	0.00
6	11	4287.0	-	-	-	0.00
7	12	5850.0	-	-	-	0.00
8	13	6368.0	-	-	-	0.00
9	14	5238.0	-	-	-	0.00
10	15	6490.0	-	-	-	0.00
11	16	5290.9	5291 opt.	-	-	0.00
12	17	7700.0	-	-	-	0.00
13	18	8916.0	-	-	-	0.00
14	19	6717.9	6897	6897	5	0.03
15	20	6748.0	-	-	-	0.00
16	21	12577.0	-	-	-	0.00
17	22	10559.9	10560 opt.	-	-	0.00
18	23	8473.0	-	-	-	0.00
19	24	10068.0	-	-	-	0.00
20	25	10873.0	-	-	-	0.00
21	26	14719.0	-	-	-	0.00
22	27	14206.0	-	-	-	0.00
23	28	16547.0	-	-	-	0.00
24	29	12221.0	-	-	-	0.00
25	30	16071.0	-	-	-	0.00

1) LP is initial LP value

2) TP is improved value by Transportation Problem

3) TS is final value by tree search

4) Node is node number in tree search

5) Error $= \frac{\text{optimum} - \text{LP}}{\text{TS}}$

Table 2-7. Computing Times[1] by GET and CAT Algorithms

No. of Flight	GET			CAT			Ratio[3] (%)
	Min.	Ave.[2]	Max.	Min.	Ave.[2]	Max.	
6	4.22	6.64	17.97	3.06	3.42	7.28	51.5
7	4.83	8.73	18.65	3.27	3.94	8.35	45.1
8	5.43	9.35	19.75	3.21	4.15	9.26	44.3
9	4.96	9.95	19.14	4.58	4.54	10.74	45.6
10	5.74	10.29	22.35	5.25	5.27	12.37	51.2
11	6.39	12.64	31.42	6.47	7.61	15.41	60.2
12	7.25	14.66	27.49	7.19	9.59	14.83	65.4
13	8.66	19.36	30.65	7.90	11.80	17.22	60.9
14	9.67	22.28	36.82	7.27	13.52	22.50	60.6
15	10.45	28.43	35.61	9.23	15.83	27.57	55.7
16	11.50	32.57	38.72	11.35	19.58	31.63	60.1
17	12.78	34.66	47.86	10.73	20.64	34.61	59.5
18	12.17	47.67	58.70	12.41	23.91	35.77	49.8
19	14.76	50.54	60.07	13.37	25.78	42.89	51.0
20	17.31	54.73	59.45	15.40	29.45	41.87	53.8
21	21.94	61.59	63.47	17.29	33.64	46.54	54.6
22	25.89	73.78	65.41	19.62	35.58	50.05	48.2
23	23.48	79.93	72.18	21.14	36.83	56.41	46.1
24	27.72	86.97	86.64	20.65	39.54	59.79	45.5
25	35.11	92.82	107.45	25.71	43.46	63.26	46.8
26	39.26	108.51	104.52	23.54	47.31	74.60	43.6
27	38.63	115.31	129.63	28.47	52.12	81.88	45.2
28	40.27	126.75	154.78	35.63	54.35	93.68	42.8
29	42.63	133.06	175.04	39.68	58.47	104.52	43.9
30	45.52	145.74	188.63	42.75	63.22	116.19	43.3

1) Seconds on IBM PC/AT

2) Average of 10 problems for each size of CSP

3) Ratio = average of CAT / average of GET (%)

2-6. Conclusion

In this chapter, we have reviewed the CSP and formulated it as a 0-1 integer program. We then reviewed the graph expansion technique (GET) of Cheddad which imposes the work-duty-time constraints by means of expanding the CSP graph and defining a new IP formulation based on this expanded graph. We noted that this approach is inherently problematic because of the large number of constraints in the corresponding IP formulation. Any attempt to reduce the number of these constraints by relaxation (either surrogating them or relaxing them in a Lagrangean fashion) produces a more serious problem in the sense that problem structure is destroyed and the many nice properties of the IP formulation are lost. In particular, the LP relaxation of the full IP formulation of Cheddad is almost always integer, and hence no further effort (tree search, etc) is required to get the optimal solution. This property is destroyed if constraints of the above formulation are surrogated with the result that the reduced problem becomes worthless as a means of solving the CSP. We introduced a new IP formulation of the CSP based on the expanded graph (the *clique aggregation technique*, CAT). This new formulation has the same number of constraints as the one above, but these constraints are much "stronger" (when surrogated) than the surrogated constraints in Cheddad's formulation. In fact the new surrogated constraints are not much weaker than their expanded parents. This is demonstrated by the fact that in only 3 out of 250 test problems did the solutions to the surrogated and non-surrogated problems differ. Thus we could now produce an effective smaller "formulation" which could be solved by an LP to produce almost-always an optimal feasible solution to the CSP. In cases where this was not so, we introduced a heuristic search (based on the solution of a Transportation Problem) which provided a heuristic (integer feasible) solution. This heuristic turned-out to produce the optimal result in all cases. In

particular for the 3 problems not optimally solved by the LP relaxation, this heuristic produced the proven optimal answer for 2 of them (lower bound = upper bound) and produced an unproven optimal answer for the other problem. Finally, we introduced a tree search algorithm using upper bounds from the Transportation Problem, and lower bounds from the LP relaxation, to resolve the last remaining problem (i.e. prove that the heuristic solution produced by the Transportation Problem was optimal). Computational results are given for 250 randomly generated test problems and the computational performance of the two algorithms is compared with CAT requiring about half the computing time of GET and very much smaller memory.

Chapter 3

ITERATIVE TIME-CONSTRAINT ADDITION ALGORITHM

3-0. Summary

The IP formulation of the Crew Scheduling Problem (CSP) based on the incremental addition of time constraints is considered in this chapter. The direct solution of the complete IP formulation of the CSP is difficult due to the large number of mainly redundant constraints which are not binding at the optimal solution. We can reduce the problem size by avoiding the redundant constraints. We try, therefore, to formulate the CSP using the "minimum" required number of constraints. Constraints are added only when they are necessary. The initial formulation starts with the network flow constraints which satisfy the work-duty-time and the transition time condition between pairs of flights. If the solution of this initial formulation satisfies all the other conditions (on the length of the solution path), it must be the optimal solution. Otherwise we add the unsatisfied work-duty-time constraints in the form of a strengthened unit constraint, and solve the problem again. In case the solution of the LP relaxation is not integer we use the fractional LP solution to generate fractional solution paths which are then checked for work-duty-time feasibility. If these fractional paths are infeasible in duration, additional constraints are added. Finally, when no fractional paths which are infeasible can be found, a tree search technique is used to get integer results. Obviously integrality may produce infeasible solution paths again, and further constraints may need to be added. We iterate the above procedures until the optimal solution is obtained. Computational results show that the number of constraints required by such a procedure is considerably reduced.

3-1. Introduction

The general IP approaches try to formulate the model with all the constraints included "*once and for all*" in the beginning, and then algorithmic methods are used to solve the problem. In most cases since the number of constraints explodes astronomically, the problem size becomes unmanageable soon after starting the algorithm.

Many time constraints in the CSP can be removed because they are found to be redundant. The attempt to remove these redundant constraints, however, soon faces the fundamental problem of how to find such constraints. In fact we cannot know a priori which constraints are redundant before we get any solutions. A similar idea of this kind is found in the approaches to the Travelling Salesman Problem where subtour elimination constraints are added as and when needed.

Let us consider the complete formulation first. More detailed discussions about each expression follow in the remainder of this chapter. Assuming the flights are arranged in increasing order of starting time, we present the IP formulation of the CSP as follows. (The subscript 0 is used to refer to both the super-source S and the super-sink T, when no confusion can arise.)

$$\text{minimise} \quad Z = \sum_{(i,j)\in N} C_{ij}X_{ij} + C_K X_K \qquad \text{(3-1)}$$

$$\text{subject to} \quad \left.\begin{aligned} &\sum_{j=1}^{n} X_{0j} = X_K \\ &\sum_{j=0}^{n} X_{ij} = 1, \qquad \text{for } i = 1, \ldots, n \\ &\sum_{i=0}^{n} X_{ij} = 1, \qquad \text{for } j = 1, \ldots, n \\ &\sum_{i=1}^{n} X_{i0} = X_K \end{aligned}\right\} \text{---------- (3-2)}$$

$$\sum_{(i,j) \in P^-} X_{ij} \leq |P^-| - 1 \qquad \forall P^- \in \mathfrak{F}^- \text{ ---------- (3-3)}$$

$$X_{ij} \in \{0,1\} \qquad \text{for } 1 \leq i < j \leq n \text{ ---------- (3-4)}$$

where (i,j) is the transition arc from flight i to flight j

C_{ij} = linking cost between flights i and j [cost of arc(i,j)]

C_K = crew cost per unit crew

X_{ij} = 1, if (i,j) is in optimal solution

= 0, otherwise

X_K = number of crew to work in the flight schedule

$\overline{N}$ = { set of all feasible transitions }

n is the number of total flights

P^- is an infeasible path whose time duration exceeds the work-duty-time. We consider P^- to be a set of all transition arcs in such a path. When no confusion arises, we will also use P^- to be the set of vertices, rather

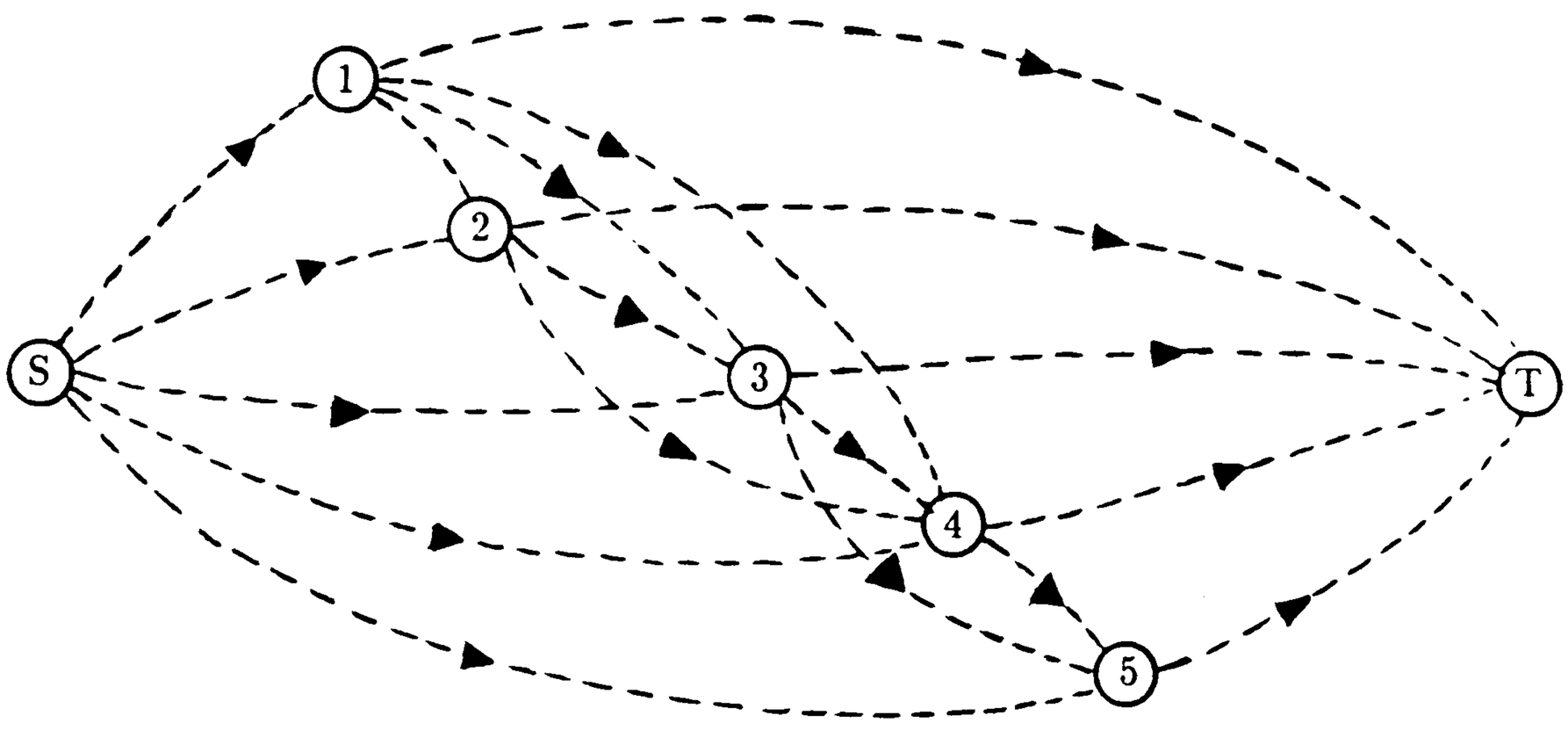
than arcs, on the path.

$|P^-|$ is the cardinality of P^-

$\mathcal{F}^-$ is the set of all paths from the super-source to the super-sink (which are infeasible because of the work-duty-time restrictions)

In the above formulation the Constraints (3-2) are the typical network flow constraints. Suppose these constraints are represented in the graph of Figure 3-1 and vertex-disjoint Constraints (3-3) are ignored. In this graph G, any combination of paths from the super-source S to the super-sink T covering every flight in G is feasible when Constraints (3-3) are ignored. For example the paths S-3-5-T and S-1-2-4-T form a feasible solution.

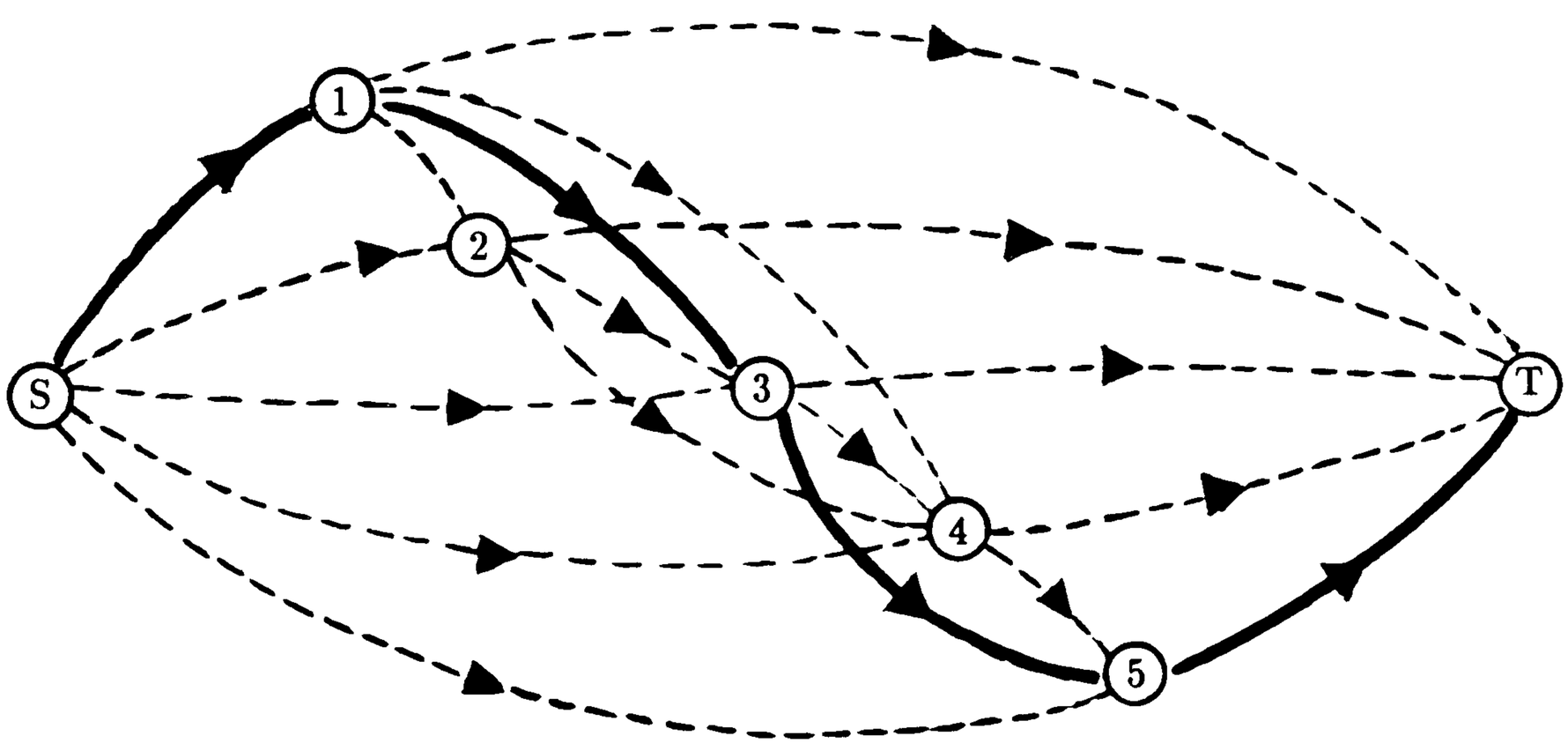
Figure 3-1. Network Presentation of the Initial Graph G of the CSP



However, if we add the work-duty-time constraints, the initial graph would be

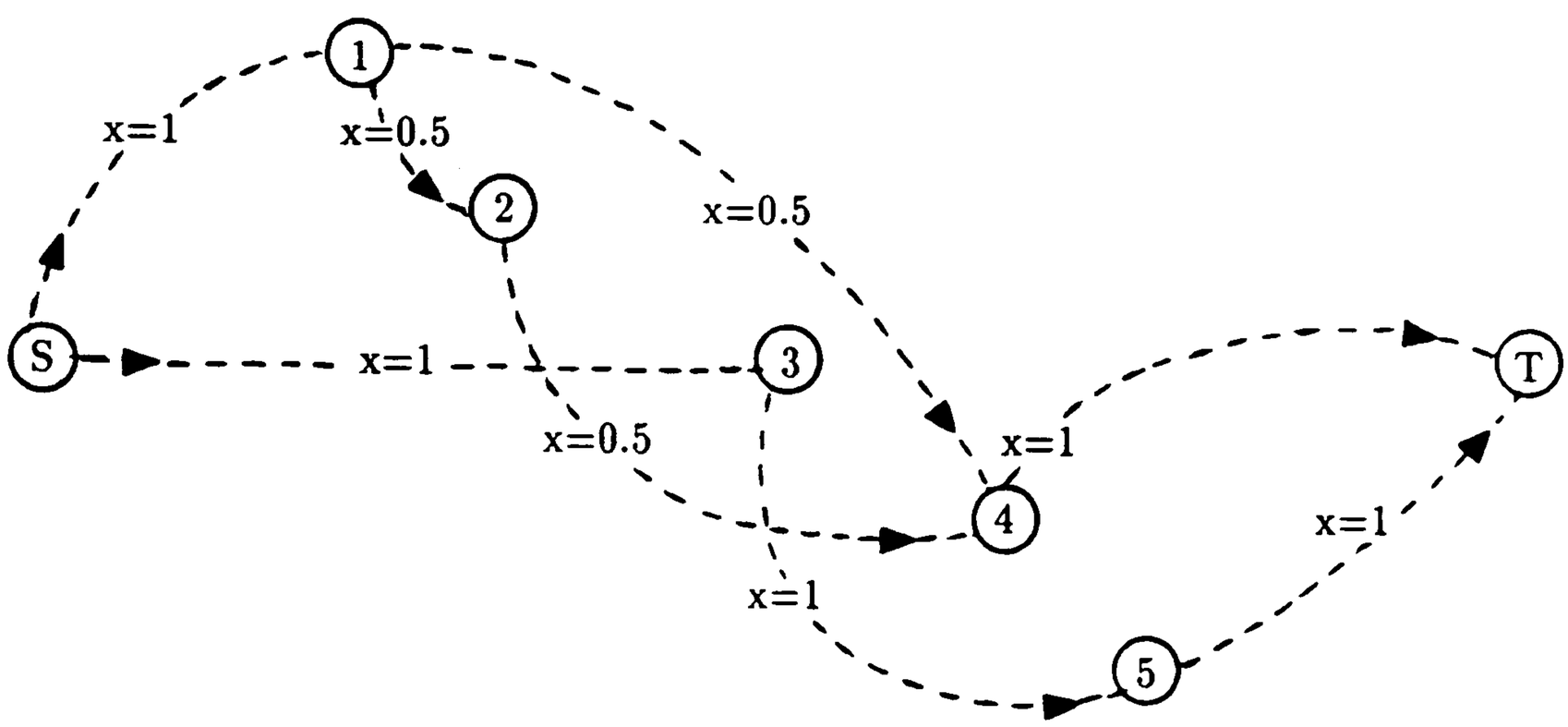
changed as follows. Suppose we have a time constraint that the path S-1-3-5-T should be less than the work-duty-time. Then this constraint is added as the heavy lines in the following graph. The heavy line means that the corresponding path should not be included in any feasible solution. As the number of flights increases the number of this kind of time constraint increases drastically and the formulation size, therefore, explodes to an unmanageable extent.

Figure 3-2. Network Presentation of Time-constraints Added Graph $\overline{G}$ of the CSP



In this chapter we develop an algorithm to get the optimal solution to the CSP with the "minimum" number of constraints by adding the time constraints step by step. In the sense that the time constraints are added to the original graph iteratively, let us call this algorithm the ***iterative time-constraint addition*** algorithm (ITA). First we start with an initial formulation with only network flow constraints (3-2). Then the LP solution of the relaxed problem is obtained. This LP solution may not be feasible and could, for example, be as shown by the flows in Figure 3-3.

Figure 3-3. A Solution to the LP Relaxed CSP



The graph implied by the non-zero flows produced by the LP and which is shown in Figure 3-3 may contain paths (such as P^-=S-1-2-4-T) which are work-duty-time infeasible. Thus, from the LP solution we can exclude such a path by adding the constraint:

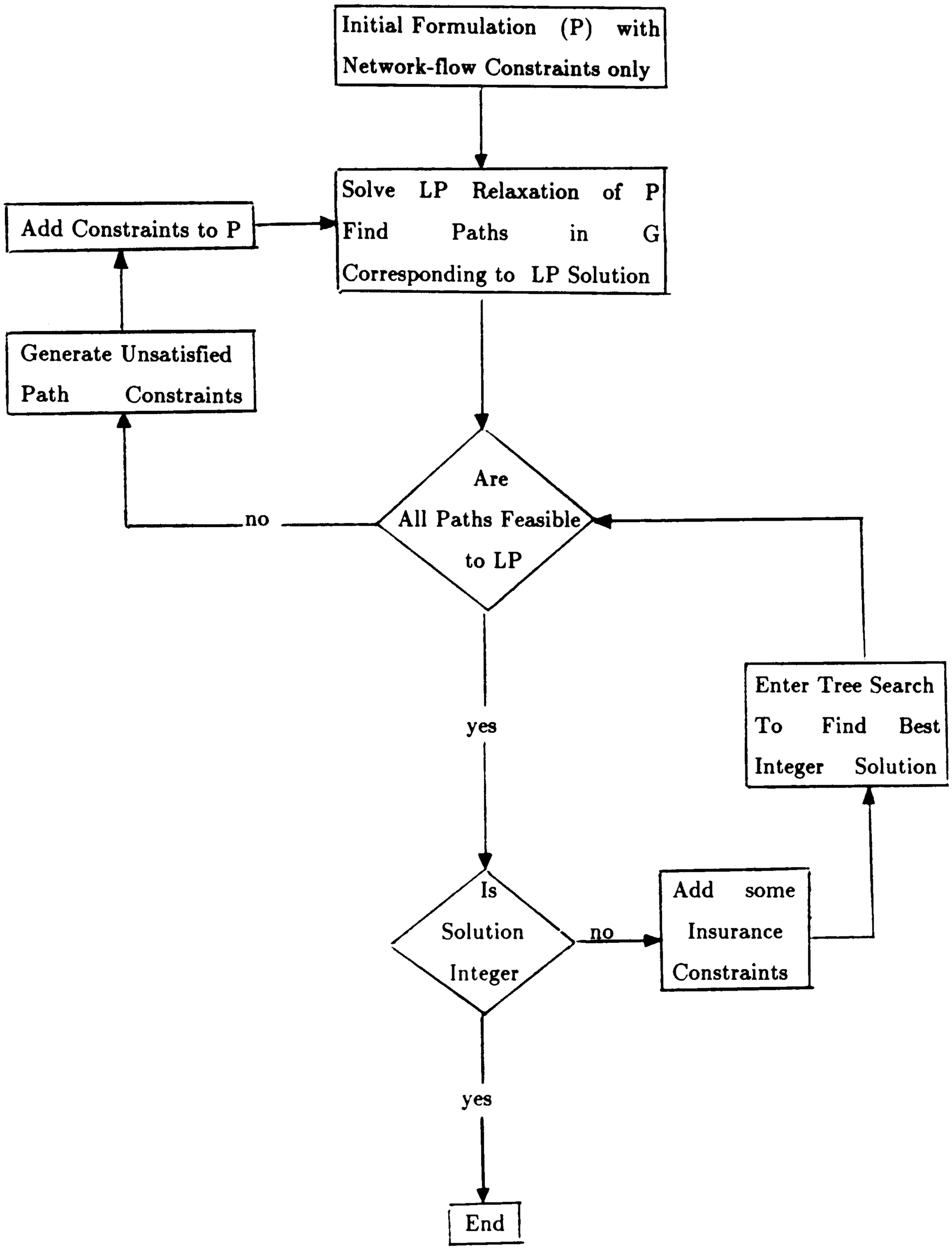
$$X_{12} + X_{24} + X_{14} \leq 1 \quad \text{----------} \quad (3\text{-}5)$$

(Note that the cardinality of the path P^- excludes the arcs from S and to T, therefore $|P^-|$=2. Also note that although the identified path P^- is S-1-2-4-T we also include in (3-5) above the variable X_{14} which is between two vertices of P^- even though arc (1,4) $\notin P^-$. The procedure is explained in detail in Section 3-4-1.)

Constraints of this nature generated by the LP solution are added to the LP (in fact the constraints are "strengthened" before addition) until the LP solution can no longer identify infeasible paths (See Figure 3-4). At that stage we "could" proceed to produce an integer solution by a tree search procedure. However, this tree search procedure is likely to be the most computationally expensive part of the algorithm, and if the integer solution at the end of it happens to have become infeasible again, a great deal of computing efforts would have been essentially wasted.

In order to avoid this situation, we have decided to add prior to entering the tree search a set of "***insurance***" constraints, in order to minimize the likelihood that the integer solution becomes infeasible. Note, however, that we cannot make use of the LP solution to generate the "*insurance*" constraints and such constraints must therefore be chosen at random (choosing the smallest cardinality infeasible paths to generate them) until a predefined number of them have been included. Thus, these "*insurance*" constraints are generated by enumerating paths from S to T in order of increasing cardinality (using a k-shortest path algorithm *[Yen 1971]*) and testing these paths for feasibility to determine whether to use them to generate an added constraint or not.

Figure 3-4. Process of ITA Algorithm



(It is assumed that each flight is individually feasible)

3-2. Variables in ITA Algorithm

Initially we can remove the variables corresponding to transitions between flights which cannot satisfy the time constraints even as a single transition. Thus, the first step of our formulation is to extract the necessary variables from the combination of flights and to reduce the initial problem size by removing the unnecessary time constraints for any feasible solution.

The variable X_{ij} in CSP represents the linking of flight i to flight j. We will extract only the variable X_{ij} in the CSP, if it satisfies conditions (3-6) and (3-7). The other variables can be ignored because they cannot be included in the feasible solution. The first expression says that only two flights whose transition time is less than the time gap between them are to generate a meaningful variable. The second expression says that the time duration from the starting time of one flight to the finishing time of the other must be less than the work-duty-time.

$$FT_i + T_{ij} \leq ST_j \quad \text{---------- (3-6)}$$

$$FT_j - ST_i \leq WT \quad \text{---------- (3-7)}$$

where FT_i is the finishing time of flight i

ST_i is the starting time of flight i

T_{ij} is the transition time from flight i to flight j

WT is the work-duty-time

Consider the following example problem which has 7 flights and 5 airports to be served. We assume that the work-duty-time is 300 minutes and that the crew cost is 50. The flights are arranged in the ascending order of their starting times.

Table 3-1. Flights of Example Problem

Flight No	Starting Time	Finishing Time	Starting Place	Finishing Place
1	30	120	A	C
2	100	150	A	B
3	160	200	B	C
4	280	330	C	D
5	300	450	C	E
6	420	570	D	A
7	450	600	E	D

The problem involves 5 airports. The transition times between the airports are given as follows. We assume these transition times are not symmetric as for the other examples in this thesis.

Table 3-2. Transition Times of Example Problem

Place	A	B	C	D	E
A	0	50	90	120	100
B	80	0	40	60	80
C	100	60	0	50	150
D	150	70	80	0	130
E	120	90	110	150	0

The following table shows the corresponding transition cost of the example problem. The costs are also assumed not to be symmetric.

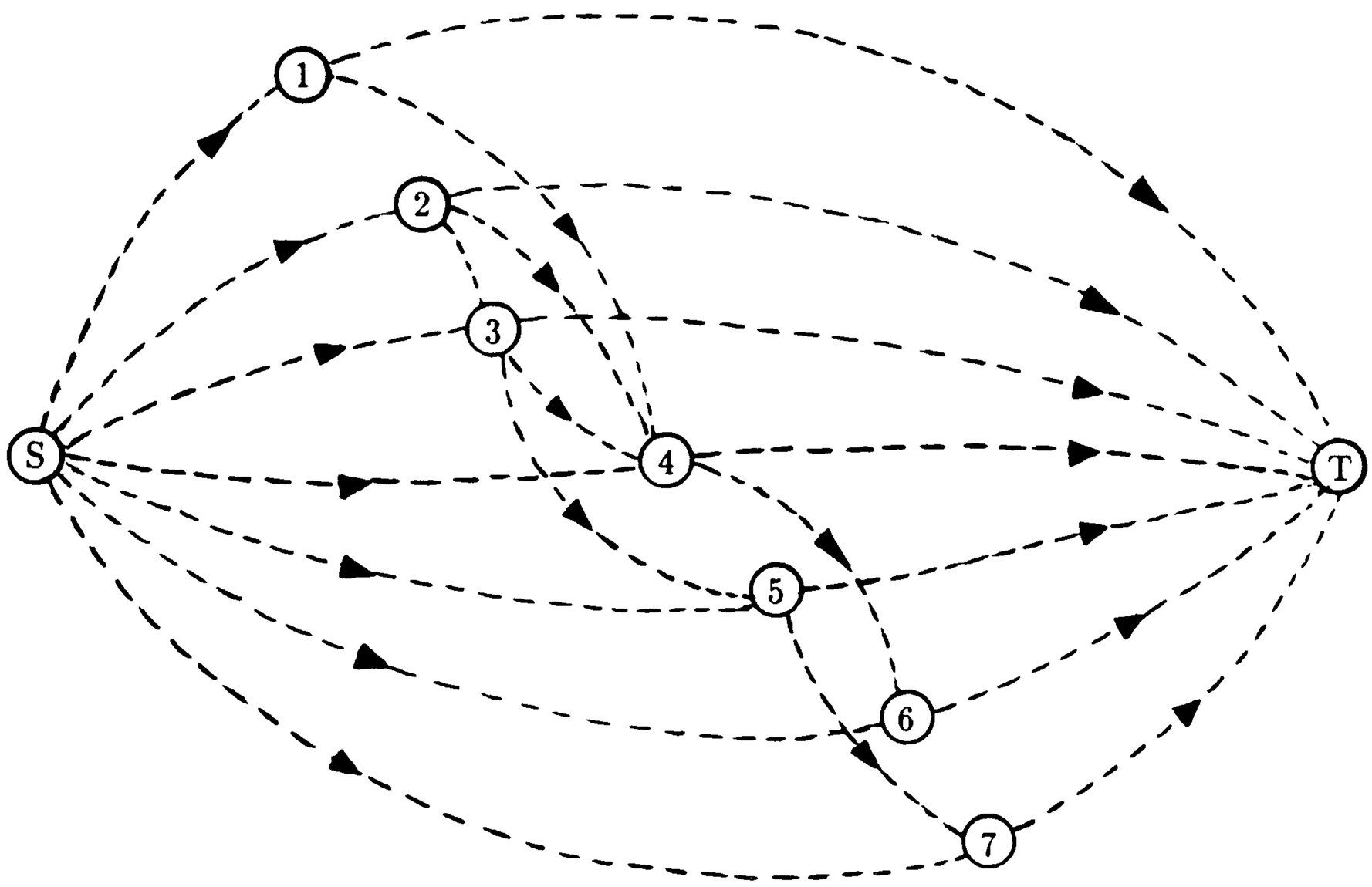
Table 3-3. Transition Cost of Example Problem

Place	A	B	C	D	E
A	0	60	90	110	100
B	80	0	50	60	90
C	90	70	0	70	140
D	130	80	60	0	150
E	130	80	120	140	0

The network presentation of the example problem is shown in Figure 3-5. Flight 1 cannot be connected to flights 2 and 3 because of the transition time. The work-duty-time permits flight 1 to be connected only with flight 4. We do not need to consider the other connections between flight 1 and flights 5, 6 and 7 whose time durations are

greater than the work-duty-time.

Figure 3-5. Network Presentation of Example Problem



We extract the following 21 variables from the example problem according to the above criteria of time constraints. The variable X_{ij} denotes a possible transition from flight i to flight j. Consequently the variable X_{ij} takes the value one when flight i and flight j are linked, whereas it is zero when they are not. The zero of the first subscript (resp. the second subscript) represents the super-source S (resp. the super-sink T).

X_{01}, X_{02}, X_{03}, X_{04}, X_{05}, X_{06}, X_{07},

X_{14},

X_{23}, X_{24},

X_{34}, X_{35},

X_{46},

X_{57},

X_{10}, X_{20}, X_{30}, X_{40}, X_{50}, X_{60}, X_{70}

3-3. Objective Function of CSP in ITA

We have already discussed in Section 2-2-1 of Chapter 2 the significance of the various terms in the objective function of Expression (3-1) and we will not comment on it here further.

3-4. Constraints in ITA Formulation

There are three types of constraints in the ITA formulation:

(a) Constraints (3-2) are the network flow type constraints, very similar in nature to the disaggregated network-flow constraints of the expanded graph; (described by Formulation NP_2 in Section 2-2 of Chapter 2).

(b) Constraints (3-4) are the natural integrality constraints.

(c) Constraints (3-3) are the constraints that are specific to this formulation. They 'suggest' their generation iteratively whilst the algorithm proceeds, since it is out of

the question that they can be written down as real constraints a priori in any computation.

In this section we deal with these time Constraints (c) which are derived from the work-duty-time. This kind of constraint may be the most difficult bottleneck in the CSP since the work-duty-time regulations usually generate such numerous constraints that even small size CSP cannot be handled in an a priori formulation of this type.

We, therefore, propose a method to manage the numerous time constraints step by step. The first step is generating the constraints only when they are required to get the optimum as explained earlier in Section 3-1. The next step is partitioning any generated constraint into a set of smaller, stronger constraints. The last step is the addition of "***insurance***" constraints generated by a k-shortest path algorithm as mentioned in Section 3-1 above in order to reduce the possibility of a generated integer solution being infeasible (See Figure 3-4 for an overall view of constraint addition).

3-4-1. Generation of Constraints from the LP Solution

Since the family $\mathcal{F}^-$ in Constraints (3-3) is huge we describe the iterative generation of these constraints in our ITA algorithm. Let us consider the problem defined by (3-1), (3-2), a subset of (3-3) and the linear relaxation of Constraints (3-4) as problem $\mathcal{P}$. The LP solution to $\mathcal{P}$ is, in general fractional, given by $[\overline{X}_{ij}]$. Construct the subgraph $G(\mathcal{P})$ of the graph G describing the problem $\mathcal{P}$ (See, for example, Figure 3-1) as follows:

Let G=(V,A) and G($\mathcal{P}$)=(V,B)

where V is the set of vertices of G and G($\mathcal{P}$)

A is the set of arcs of G

B is the set of arcs of G($\mathcal{P}$).

Define G($\mathcal{P}$) by the set B as follows:

$$B=\{ (i,j) \mid (i,j) \in A, \overline{X}_{ij} \neq 0 \} \quad \text{----------} \quad (3\text{-}8)$$

We now wish to find S to T paths in G($\mathcal{P}$) and check them for work-duty-time infeasibility. For a small non-dense graph G($\mathcal{P}$), this is an easy task. However, for a larger graph, and especially one where the set of B is large, (e.g. if the LP solution is 'very' fractional), the complete set of paths from S to T in G($\mathcal{P}$) is not easy to obtain.

In our procedure, we overcame this problem by enumerating systematically as follows:

Step 1. Firstly find the 'longest' paths in G($\mathcal{P}$) from S to T. This is normally an NP-complete problem for a general graph, but only a linear complexity problem of the graph is acyclic as is the case here. If the length of this longest path is feasible for the work-duty-time restriction, then all paths in G($\mathcal{P}$) are feasible and the generation of constraints from the LP solution is complete.

Step 2. If the longest paths in G($\mathcal{P}$) is not feasible then use a k-shortest path algorithm to generate in order of length (from the shortest to the longest) k paths from S

to T. Note that if k is chosen to be very large (e.g. set to ∞), all paths in G($\mathcal{P}$) from S to T will be generated in length-order. In our case we used the algorithm of Yen *[1971]* to generate the k-shortest path.

Step 3. For each path generated, say path P^- and which violates the work-duty-time restrictions, we can then check that the constraint

$$\sum_{j \in P^-,\ i \in V_j^+} X_{ij} \leq |P^-| - 1 \qquad (3\text{-}9)$$

where $V_j^+ = \{\ i \mid (i,j) \in B\ \}$,

is violated by the LP, and if so then include this constraint into problem $\mathcal{P}$. A similar procedure can be used to generate a possible constraint from every other path P^- produced at step 2. Note that Constraint (3-9) does not necessarily violate the LP solution, even if P^- is infeasible in terms of the work-duty-time, and therefore, it is quite possible that the addition of Constraint (3-9) will not disturb the existing LP solution in any way. Thus, Constraint (3-9) must be checked against the LP solution $[\overline{X}_{ij}]$ and only added to problem $\mathcal{P}$, if infeasible.

Step 4. When all such constraints are added to $\mathcal{P}$, the LP is solved again to obtain a new solution $[\overline{X}_{ij}]$ and the process repeated from Step 1 until no constraints can be found for addition at Step 3.

For the example in Section 3-2, the initial LP solution is:

$$X_{14}=X_{46}=X_{23}=X_{35}=X_{57}=1$$

implying a path (say) S-2-3-5-7-T in graph G($\mathcal{P}$). This path is identified at Step 3 as one of the paths in G($\mathcal{P}$) that is infeasible (length of 500 compared with the maximum allowed length of 300), and Constraint (3-9) becomes:

$$X_{23} + X_{35} + X_{57} \leq 2 \quad \text{----------(3-10)}$$

(Note that if the LP solution $[\overline{X}_{ij}]$ has values $\overline{X}_{ij}=1\ \forall(i,j) \in P^-$, then $B(P^-) \equiv P^-$ and Constraint (3-9) would always be violated by the LP solution.)

Decomposition of Constraints (3-9)

In the above procedure, Constraints (3-9) have been generated from S to T paths (P^-, say) which were infeasible. However, it may also be possible, that a sub-path of P^- could itself be infeasible in duration. Thus, if P^-=(S, a, b, c, ..., *i*, *j*, *k*, T) then the sub-path (b, c, ..., *i*) could also be infeasible. In this case Constraint (3-9) is clearly not the strongest constraint that could be written from P^-. Indeed each Constraint (3-9) could be decomposed into a set of constraints

$$\sum_{j \in W^-,\ i \in V_j^+} X_{ij} \leq |W^-| - 1 \quad \text{----------(3-11)}$$

one for each sub-path W^- of P^- which is also infeasible. Clearly Constraints (3-11) dominate Constraint (3-9). For the example of previous section, Constraint (3-10), obtained from (3-9) directly, could also be decomposed into:

$$X_{35} + X_{57} \leq 1$$

(S-3-5-7-T is an infeasible sub-path of S-2-3-5-7-T)

and:

$$X_{23} + X_{35} \leq 1$$

(S-2-3-5-T is also an infeasible sub-path of S-2-3-5-7-T)

3-4-2. Generation of Insurance Constraints

It was mentioned earlier, that we would like to avoid the situation, whereby a feasible LP solution is finally produced, having added a number of constraints of type (3-9); only to find that when an integer solution is produced (after some tree search procedure is applied to the LP solution - See Section 3-5) we discover that the integer solution has become once-more infeasible. This is a very serious situation since the tree search is a computationally very time consuming process.

With this in mind, it was decided that after no more constraints of type (3-9) could be added, a new constraint-generation procedure would be applied, independently of the LP solution, in order to maximize the chances that the integer solution found by the initial subsequent tree search is an optimal "*feasible solution to the CSP*" and no more further tree searches are embarked on. In view of the purpose of these additional constraints we will refer to them as "***insurance***" constraints.

Insurance -constraints Generation Procedure

It is easy to see that the most useful constraints of type (3-3), that one could use as "***insurance***" constraints, would be derived from those S to T paths in the graph G (See Figure 3-1) which although of small cardinality, they are infeasible with respect to the work-duty-time. Some of these path have already been generated from the LP solution, either as Constraints (3-9) or one of its decompositions. Other constraints, however, of the above type might well have not been generated by the investigation of the LP solution and would be candidates for the insurances constraint group. Thus, the following procedure was applied to generate the insurance constraints.

Step 1. Apply the k-shortest path algorithm directly (mentioned earlier) to graph G, with a values of k high enough (say k=1000) so as to generate the first 1000 paths from S to T in order of their cardinality.

Step 2. For each path P^- generated in Step 1 above, and which is infeasible with respect to the work-duty-time generate the corresponding constraints.

Step 3. Check that the above constraints have not been previously generated during the LP solution construction and if not add them to problem $\mathcal{P}$.

Step 4. Repeat Step 2 and 3 for each path P^- generated in Step 1, until a predefined maximum number of insurance constraints have been generated, or no more constraints can be produced.

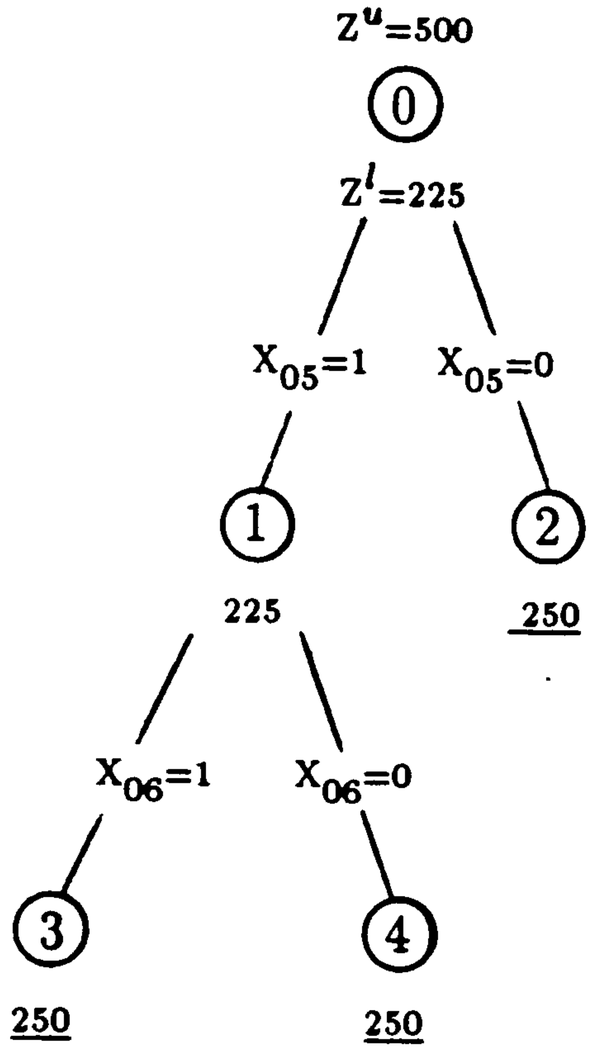
The final problem $\mathcal{P}$ is now ready for the application of a tree search algorithm to find an integer optimal solution.

3-5. A Tree Search Algorithm for Obtaining an Integer Solution

The LP solution to the relaxed problem of (3-1) $\sim$ (3-3) may not be integer. In this case we make use of a tree search algorithm to impose integrality. This tree search algorithm is the generic procedure described in Section 2-4 of Chapter 2 and will not be described in detail again. It consists of branching (to 0 or 1) on those variables with fractional values in the LP solution. The value of the LP itself is used as a bound. Note that all the insurance constraints are added prior to the tree search and hence the number of constraints during the tree search remains constant.

In the example problem of Section 3.2, the whole structure of a tree search is shown in Figure 3-6. We assume an initial upper bound of 500, obtained by a trivial heuristic. Note that although the initial LP solution to this example before Constraints (3-9) were appended was integer, as mentioned in Section 3-4-1 (the LP is a network-flow problem at this stage), the final LP solution was in fact fractional.

Figure 3-6. Tree Search Diagram of Example Problem



As discussed above, we branch from node 0 to nodes 1 and 2 by setting the variable X_{05} to 0 or 1. The LP solution to the subproblem of node 2 is integer and of value 250, so this node is fathomed. Solving again the LP for the problem of node 1, we get a non-integer solution of value 250. We choose X_{06} as the next variable to branch on, producing nodes 3 and 4. We get an integer solution to the LP solving problem of node 4 and its value is 250 so this node is also fathomed. At node 3, we also get an integer LP solution of value 250. The duration of this solution is within the work-duty-time of 300. X_K is 5 which is in fact the same as the minimum possible number of crew. Therefore we get the optimal solution and the tree search is terminated.

3-6. Computational Results

Our algorithm ITA was tested on the 250 test problems described in Section 2-5 of Chapter 2. The computational results are summarised in Tables 3-4, 3-5 and 3-6.

Table 3-4 shows the number of constraints in the model described by (3-1) ~ (3-4), and the number of constraints of type (3-3) generated, via Expression (3-11). The initial lower bounds are on average within 4.6% from the optimal solution. Note that only a few of the problems in Table 3-4 have been solved optimally at the root node, although (with the exception of problem 23) the number of nodes in the tree search required to prove optimality, was quite small. Incidentally although the generated feasibility constraints given by Expressions (3-9) and (3-10) are based on the LP solutions, the addition of the "insurance" constraints prior to solving the integer problem by tree search, has ensured that in all cases, only one tree search needed to be executed. The advantage of a formulation which is not based on graph expansion is that the mathematical model is smaller in size, but it turns-out that it is quite significantly harder to solve than the larger model based on graph-expansion described earlier in Chapter 2. (Compare, for example, the results of Table 3-4 with Table 2-6)

Table 3-5 compares problem size for the ITA algorithm with the GET algorithm for 250 test problems, 10 for each size. As shown in the table we can reduce the problem size by 80% on average as far as number of variables is concerned and by 40% on average as far as number of constraints is concerned, compared with algorithm GET. Moreover these reduction factors increase as the problem size increases.

Algorithm ITA, however, takes longer computational time than GET algorithm. Table 3-6 shows that algorithm ITA takes 75% longer than algorithm GET, on average. Moreover this ratio increases with problem size as shown Table 3-6.

Table 3-4. Computational Results of ITA Algorithm

Problem	Flight#	Variables#	Constraints[1]#	Nodes#	Bound[2]	Optimum	Ratio[3]
1	6	22	15	0	200.0	200	1.00
2	7	27	16	5	250.0	250	1.00
3	8	36	18	0	200.0	200	1.00
4	9	38	20	0	250.0	250	1.00
5	10	29	26	11	3782.0	4233	0.89
6	11	37	25	0	4287.0	4287	1.00
7	12	33	32	7	5340.0	5850	0.91
8	13	33	31	3	6098.0	6368	0.96
9	14	43	39	7	4930.0	5238	0.94
10	15	44	34	0	6490.0	6490	1.00
11	16	56	46	5	5221.0	5291	0.99
12	17	54	45	11	6959.0	7700	0.90
13	18	52	48	17	8228.0	8916	0.92
14	19	67	59	89	5976.0	6897	0.87
15	20	99	55	23	6748.0	6748	1.00
16	21	56	48	13	11827.0	12577	0.94
17	22	62	55	21	10243.0	10560	0.97
18	23	70	62	59	8000.0	8473	0.94
19	24	71	59	7	9791.0	10068	0.97
20	25	69	62	23	10040.0	10873	0.96
21	26	69	61	13	14340.0	14719	0.97
22	27	72	64	9	14147.0	14206	0.99
23	28	86	76	539	14150.0	16547	0.86
24	29	75	65	5	12148.0	12221	0.99
25	30	82	70	39	14806.0	16071	0.92

1) The number of constraints shown exclude the network-flow type Constraints (3-2)

2) Bound at the root node

3) Ratio = bound/optimum

Table 3-5. Comparison of ITA with GET for Problem Size[1]

No. of Flights	GET(A)		ITA(B)		Ratio(B/A)	
	Var.	Con.[2]	Var.	Con.[2]	Var.	Con.
6	46	19	18	15	0.39	0.79
7	61	21	22	18	0.36	0.85
8	83	24	25	19	0.30	0.78
9	92	31	27	21	0.29	0.67
10	118	35	32	25	0.27	0.71
11	145	37	35	24	0.24	0.69
12	174	41	31	28	0.18	0.68
13	198	45	32	30	0.16	0.66
14	215	48	38	35	0.18	0.73
15	231	57	48	35	0.21	0.62
16	254	64	43	38	0.17	0.59
17	284	68	46	40	0.16	0.58
18	271	72	50	45	0.18	0.61
19	282	79	52	46	0.19	0.58
20	314	86	66	52	0.21	0.60
21	340	88	53	48	0.16	0.55
22	377	92	60	52	0.15	0.57
23	398	98	63	54	0.16	0.55
24	416	105	59	55	0.14	0.52
25	448	109	62	55	0.14	0.50
26	476	114	67	60	0.14	0.52
27	514	121	73	64	0.14	0.53
28	528	127	72	65	0.13	0.51
29	563	133	79	63	0.14	0.47
30	584	138	87	75	0.14	0.54

1) Average of 10 test problems for each flight size

2) The number of constraints shown exclude the network-flow type (3-2)

Table 3-6. Comparison of Computing Times[1] for ITA & GET Algorithm[2]

Flight #	GET(A)	ITA(B)	Ratio(B/A)
6	6.64	7.95	1.19
7	8.73	9.96	1.14
8	9.35	14.67	1.57
9	9.95	17.52	1.76
10	10.29	18.49	1.79
11	12.64	21.60	1.70
12	14.66	27.17	1.85
13	19.46	35.43	1.82
14	22.28	42.83	1.92
15	28.43	55.71	1.96
16	32.57	59.33	1.84
17	34.66	64.49	1.86
18	47.67	87.66	1.83
19	50.54	92.79	1.84
20	54.73	97.62	1.78
21	61.59	119.13	1.93
22	73.78	141.82	1.92
23	79.93	147.29	1.84
24	86.97	161.50	1.86
25	92.82	167.49	1.81
26	108.51	177.81	1.64
27	115.31	201.33	1.75
28	126.75	212.39	1.68
29	133.06	220.68	1.66
30	145.74	247.86	1.70

1) Seconds on IBM PC/AT

2) Average of 10 test problems for each flight size

3-7. Conclusion

In this chapter we present an IP formulation of the CSP based on iterative time-constraint addition (ITA). In ITA algorithm the time constraints are added to the problem only when they are necessary to reduce the problem size. At each iteration if the solution is not feasible in terms of work-duty-time, the unsatisfied time constraints are added. To avoid unnecessary iterations, the constraints are partitioned into smaller units of a stronger form. We have defined a measure of problem "feasibility" which is applicable to fractional solutions, and we use this measure to generate constraints that inhibit the emergence of non-feasible integer solutions, thus reducing the number of times we have to use a tree search to just once. The computational results show that the size of this formulation both in terms of numbers of variables and numbers of constraints, is reduced. With this formulation we can reduce (on average) the necessary number of variables by 80% and the number of constraints by 40% for CSP's of up to 30 flights. Consequently with this algorithm larger size problems could be tackled more easily, although ITA takes 75% longer to get the optimal solutions in terms of computing time. Despite its weakness in terms of computing time, we believe that our algorithm ITA with further elaboration could provide the basic foundation to tackle larger scale problems.

Chapter 4

A SET-PARTITIONING BASED HEURISTIC

4-0. Summary

The currently most often used approach for dealing with the CSP in practice, involves the use of a Set Covering Problem (SCP) as the mathematical model. The columns of the SCP are generated a priori as a small subset of the set of S to T paths in the graph G (defined in Chapter 3) which satisfy the work-duty-time restrictions. By the very fact that not every feasible S to T path of G can be generated (because of their huge numbers), the Set Covering Problem approach to the CSP is inherently approximate. Moreover, even with only a few thousand columns in the SCP, an optimal solution to the SCP is not easy to get and problems of more than about 4000 columns become impossible to solve exactly. Hence a second approximation is often made whereby the SCP is solved heuristically, either by an incomplete exact tree search, or by a specialised heuristic.

In this chapter we apply the SCP model to solve the CSP but with some very significant differences from what was done previously. At no time do we make an arbitrary decision to exclude certain S to T paths of G permanently from the SCP and hence the initial generation of columns is not so significant since other columns may be generated during the procedure. Also, although the initial solution to the SCP is produced by a greedy procedure, and although this initial solution is improved by local "exchanges" (as in any λ-optimal procedure *[Lin and Kerninghan 1973]*), the actual exchange mechanism used is novel, and may involve the optimal solution to a small subproblem of the CSP.

The net merit of the above heuristic algorithm, is that the solutions obtained initially are often optimal to the CSP; in fact in over 71% of the cases, and within 4.7% of the optimum value in the remaining 29% of the cases. After the local optimization procedure the heuristic produces optimal results in all cases tried. This excellent performance for a heuristic is achieved with a reasonable computing time, so the heuristic can also be used to provide the initial upper bound for the tree searches in Chapter 2 and 3.

4-1. Introduction

The CSP can be formulated as a Set Covering Problem (SCP) or Set Partitioning Problem (SPP) as initially suggested in 1961 by Spitzer and discussed at length in *[Rousseau 1984]*. This approach has many advantages connected with the fact that complex constraints (not expressed in our algorithm of the CSP) can also be incorporated quite easily by checking that each single-crew schedule (represented by a column in the CSP) is feasible *[Rubin 1973, Balas and Padeberg 1979, Mitra 1985]*. Both SCP and SPP have been used as models for the CSP, although there is a little difference between them and one can be converted to the other, though not symmetrically *[Lemke et al 1971, Korman 1975, Paixão 1983]*. Because of the ability of the SCP and SPP models to incorporate extra (and even subjective) constraints as mentioned above, it has had a large number of uses in other problems to do with routing and scheduling in addition to the CSP *[Balas and Padeberg 1979, Lucena 1986]*.

4-1-1. The Classical SCP Formulation of the CSP

A 'rotation' in the CSP is a sequence of flights (possibly with some rest periods) which satisfy the work-duty-time and other constraints, In our terminology, since we only consider work-duty-time constraints, a rotation is an S to T path in the activities graph G satisfying these constraints. Large numbers of rotations (say $\hat{N}$) can be generated a priori for the CSP by applying some heuristic rules as to what rotations are likely to be candidates for possible inclusion in a CSP solution. A matrix $A \equiv [a_{ij}]$ can

now be set up, whose columns are the generated rotations and whose rows are the flights.

Thus, an entry a_{ij} of the matrix A is equal to 1 if flight i is included in rotation j, and equal to 0 otherwise. Let C_j be the cost associated with rotation j, (being the sum of all transitions between flights in this rotation and the crew cost C_k). The SCP formulation of the CSP can now be written as *[Spitzer 1961, Marsten et al 1981]*:

$$\text{minimise} \quad \sum_{j=1}^{\hat{N}} C_j X_j \text{ ---------- (4-1)}$$

$$\text{subject to} \quad \sum_{j=1}^{\hat{N}} a_{ij} X_j \geq 1 \qquad i = 1,\ldots,n \text{ ---------- (4-2)}$$

$$X_j \in \{ 0 , 1 \} \qquad j = 1,\ldots,\hat{N} \text{ ---------- (4-3)}$$

} SCP

(Note that an SPP formulation results if the $\geq$ in (4-2) above is replaced with $=$)

There are two major problems with the above SCP formulation of the CSP: The problem generation, and the problem solution.

4-1-2. The Problem Generation

It is quite clear, that it is computationally out of the question to generate all possible (feasible) rotations as columns of the SCP. The number of columns $\hat{N}$ must,

therefore, be set to some sufficiently large value (say a few thousand) so that the SCP is a reasonable representation of the CSP, *[Marsten et al 1981, Ball et al 1985]*, and still be solvable by whatever method (exact or approximate) that will eventually be used. Also note here, that the generation (of columns) process itself can be very time consuming, especially in cases when the number of columns being generated is very large, because only an approximate solution to the resulting large SCP is to be produced. For example, in the case of the Flying Tiger (the world's largest aircargo carrier) obtained good results for a problem with 300 flights (in one month's plan) by generating 40,000 columns and solving the resulting SCP heuristically *[Marsten et al 1981]*.

4-1-3. The Problem Solution

SCP's of 70 rows and 400 columns were solved optimally quite early-on using implicit enumeration by *[Etcheberry 1977]*. Cutting plane methods were used in *[Balas et al 1979, 1980]* to solve SCP's of up to 200 rows and 2000 columns. Similar size SCP's were solved using shortest-path-based Lagrangean Relaxation *[Shepardson et al 1980]*, a variety of graph-theoretic relaxations *[Hey 1980, Mitra et al 1985]* and using SETPAR a set-partitioning code by Marsten *[Marsten 1974]*. Somewhat larger problems, with 400 rows and 4,000 columns have also been solved using a combination of problem reductions, subgradient optimization and linear programming in *[Beasley 1987, Beasley et al 1990]*, and using state-space relaxation in *[Christofides et al 1987, Paixão 1983]*. An even larger SCP with 1,000 rows and 8,800 columns was recently reported to have been solved optimally in *[Lavoie et al 1988]*.

The above comments on the size of SCP's that can be solved optimally, give a false impression of the size of CSP's that can be solved in this way however. It has been

known for a long time now that some SCP's are much harder to solve than others. In particular, randomly generated SCP's are much simpler to solve than "real" ones, and it is, unfortunately, still quite simple to produce small SCP's (involving 50 rows and 200 columns) that cannot be solved optimally by **any** of the existing SCP algorithms. These problems tend to be ones in which the cost of columns are not very different from each other (as would be the case if the crew costs are much larger than the transition costs), and problems where the columns have costs that are proportional to the number of 1's in the columns.

4-2. Heuristic Methods

Since the SCP model of the CSP is itself approximate, especially when the SCP is very large, and since the exact solution of SCP's is difficult, a natural question to ask is as to whether good heuristic solutions to large SCP's can be obtained so that approximate solution to large CSP's can be derived. A very large number of heuristics for SCP's or SPP's have been proposed in the literature. These include: the cutting plane heuristic of *[Balas et al 1980]*, the marginal improvement heuristics of *[Toregas 1972, Rubin 1973a, Rubin 1975]*, the swapping rule heuristic of *[Orloff 1973]*, as well as the heuristics in *[Salkin 1975, Chvatal 1979, Minoux 1984, Ryan 1988]*.

4-2-1. Greedy Heuristics

One of the best known and fastest procedures for obtaining an approximate solution to the SCP is the greedy heuristic, which consists of s sequence of steps each of

which includes in the solution a variable (column) j which covers the maximum number of additional rows. The worst-case performance of this heuristic for the unicost (all C_j equal) SCP was shown by Johnson *[1974]* and Lovász *[1975]* to be given by

$$Z_H \leq (\sum_{j=1}^{d} 1/j) \cdot V(LP)$$

where Z_H is the value obtained by the heuristic, d is the largest number of 1's in any column of the SCP, and V(LP) is the value of the LP solution to the SCP.

Since $V(LP) \leq V(SCP)$, where V(SCP) is the optimal value of the SCP, we have:

$$\frac{Z_H}{V(SCP)} \leq \sum_{j=1}^{d} 1/j \quad \text{---------- (4-4)}$$

This relation has been shown by Chvatal *[1979]* to be "tight" (i.e. = is achievable) and that it is also true for arbitrary costs. Moreover in *[Ho 1979]* it is shown that the worst case bound of (4-4) is also valid and tighter for **any** greedy heuristic that assembles a solution by choosing at each stage for inclusion into the solution that $f(C_j,\Theta_j)$, where Θ_j is the number of rows covered at that step by column j.

Although the result of (4-4) should be very damning for any greedy heuristic (in the worst case); on **average** well-constructed greedy heuristics are within 10% of the optimal solution. Thus, such heuristics are useful in obtaining reasonable solutions in a very short time, especially in cases where additional operations are to be performed to improve this heuristic solution further.

4-2-2. Reduced Cost Heuristics

Another class of heuristics that has been used for large SCP's involves the use of "reduced costs" either provided by a dual feasible solution for the LP obtained from the SCP *[Balas and Ho 1980]* or provided by Lagrangean Relaxation *[Geoffrion 1974]*. In either method, the procedure involves choosing columns to enter into the solution in increasing order of their reduced cost until every row is covered. Once that is achieved, an attempt is made to remove columns from the solution-set in order of decreasing reduced cost, until a prime cover is achieved. The worst case analysis of this type of heuristic is given in *[Hochbaum 1980]* but, again although the worst case ratio $Z_H/V(SCP)$ is shown to be ≈ 1.2, in practice these heuristics are within 8% of optimality on average.

Thus, in Section 4-3 we will also apply a similar (albeit even simpler) heuristic to generate a feasible initial heuristic solution prior to applying a novel improvement procedure. We will, however, put considerable effort into this improvement phase which is performed not on the SPP but on the original CSP (and in thus unrestricted by the approximation implied by the SPP), with the result that our heuristic is vastly superior to any of the above heuristics, when applied to the CSP.

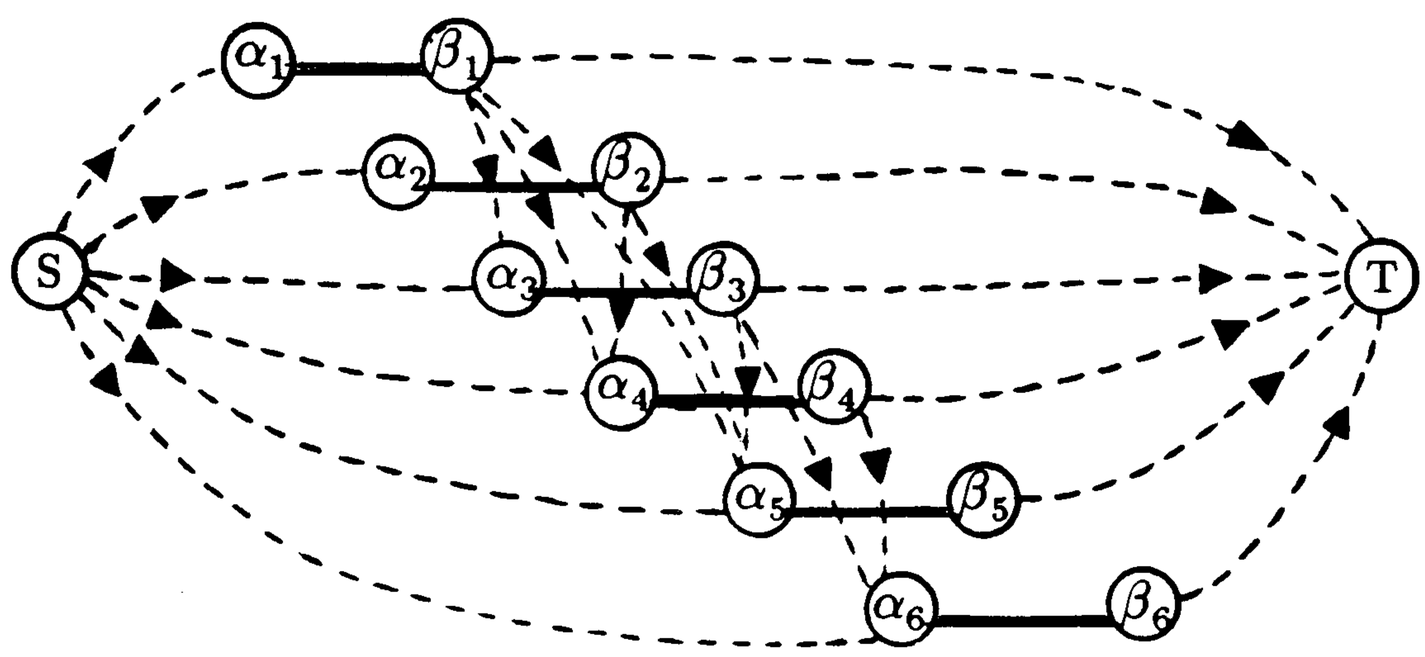
4-3. An Initial Heuristic Solution to the CSP

The first step in constructing an initial feasible heuristic solution to the CSP is to create a SPP by generating columns which correspond to crew 'rotations'.

4-3-1. Generation of Columns for the SPP

Consider once more the graph G representing the CSP, (See Figure 4-1), where the flights are represented by the arcs (α_i,β_i) and the transition between flights i and j represented by the arcs (β_i,α_j).

Figure 4-1. The CSP Graph G



To all arcs (S,α_i) and (β_i,T), $i=1,\ldots,n$ assign 0 "costs"; arcs (α_i,β_i) are assigned "costs" equal to the duration of the flight i (i.e. cost= FT_i-ST_i); and to arcs (β_i,α_j) are assigned "costs" equal to the transition time from the end of flight i to the beginning of flight j (i.e. cost=ST_j-FT_i).

A column for the SPP is now an S to T feasible path in G (i.e. a path whose total "cost" is not greater than the maximum allowed work-duty-time WT). A natural way to generate all these columns is, therefore, to generate the k-shortest paths in G in ascending order of cost, until the $(k+1)$-th shortest path has "cost" greater than WT.

Even though it is very likely that such a value of k would be far too high to be practicable for reasonable-size CSP's, it is quite possible to generate thousands of paths in this way, and these would be expected to constitute a good set of columns for the SPP. Thus, we can (once more) make use of our implementation of the k-shortest path algorithm of Yen *[1971]* to generate the initial set of columns of for the SPP.

4-3-2. The Construction of the Initial Solution

Let ϕ_j be the cost of column j, computed as the sum of the transition costs between flights included in column j, plus the crew cost (which would operate that rotation). Let us assume a virtual "worst-case" solution composed only of columns containing one flight each. Let us denote by θ_j the "gross contribution" of column j in reducing the virtual CSP cost. We compute θ_j as the difference between what the cost would have been had all the flights in column j, been individually 'covered' by crews and the cost of column j. Thus, if d_j is the number of 1's in column j, we set

$$\theta_j = C_k \cdot d_j - \phi_j \quad \text{----------} \quad (4\text{-}5)$$

(where C_k is the crew cost)

If a column j is chosen to enter the solution, certain other columns, in the set $\mathbb{J}_j$ say, cannot be chosen subsequently, (since they cover some row that has already been covered by column j). Thus, there is an "opportunity cost" in choosing column j which will define as:

$$\gamma_j = \max_{j^* \in \mathbb{J}_{j^*}} [\theta_{j^*}] \quad \text{----------} \quad (4\text{-}6)$$

Thus, we will define the "net contribution" of column j towards improving the solution as:

$$\delta_j = \theta_j - \gamma_j \quad \text{----------} \quad (4\text{-}7)$$

The values of δ_j will now be used to select columns of the SCP to construct a solution according to the following algorithm:

Step 0: Set $\mathbb{J}=\{1,2,...,\widehat{N}\}$, the set of all columns. Solution $\mathbb{S}=\phi$

Step 1: Compute δ_j for each column $j \in \mathbb{J}$

Step 2: Choose that column $\widehat{j}$ which maximizes $\max_{j \in \mathbb{J}} [\delta_j]$
and include in the solution: $\mathbb{S} \leftarrow \mathbb{S} \cup \{\widehat{j}\}$

Step 3: Remove from $\mathbb{J}$ any column j which has a 1 in any of the same rows as column $\widehat{j}$. (i.e. j covers a row already covered by $\widehat{j}$). If $\mathbb{J}=\phi$, stop; $\mathbb{S}$ is a feasible solution. Else go to Step 1.

The solution composed from the columns in the set $\mathbb{S}$ at the termination of the above algorithm is taken to be the initial heuristic solution to the CSP.

The above simple heuristic has, on the 250 test problems discussed earlier, produced answers that were on average about 4.7% from the optimal answers. The results of the above heuristic were not meant to be the final heuristic answers however. They are to be used as initial input to a second "improvement" heuristic (based on the original CSP) that produced dramatically better results, having obtained optimal answers to the CSP in all cases tried.

4-4. A Local Improvement Procedure

Let $\mathbb{S} = \{ P_1, P_2, ..., P_k \}$ be the heuristic solution produced by the procedure of Section 4-3, where $P_1, P_2, ..., P_k$ are the corresponding paths in the flight-transition graph G. Consider a subset $\mathbb{S}' \subset \mathbb{S}$ of this solution containing paths $P'_1, P'_2, ..., P'_l$, and let $N(\mathbb{S}')$ be the set of vertices of G which lie on the paths in $\mathbb{S}'$. Let $<\mathbb{S}'>$ be the subgraph of G induced *[Christofides 1985]* by the set of vertices $N(\mathbb{S}')$.

For example, let G be the flight-transition graph shown in Figure 4-2 and let us suppose that the heuristic procedure of Section 4-3 has produced the solution $\{P_1, P_2, P_3, P_4\}$ where the paths P_i are as shown in Figure 4-3. (Quite clearly $P_1, ..., P_4$ must have been paths that corresponded to some columns in the generated SPP for the CSP in Figure 4-2.)

Figure 4-2. Initial Graph G

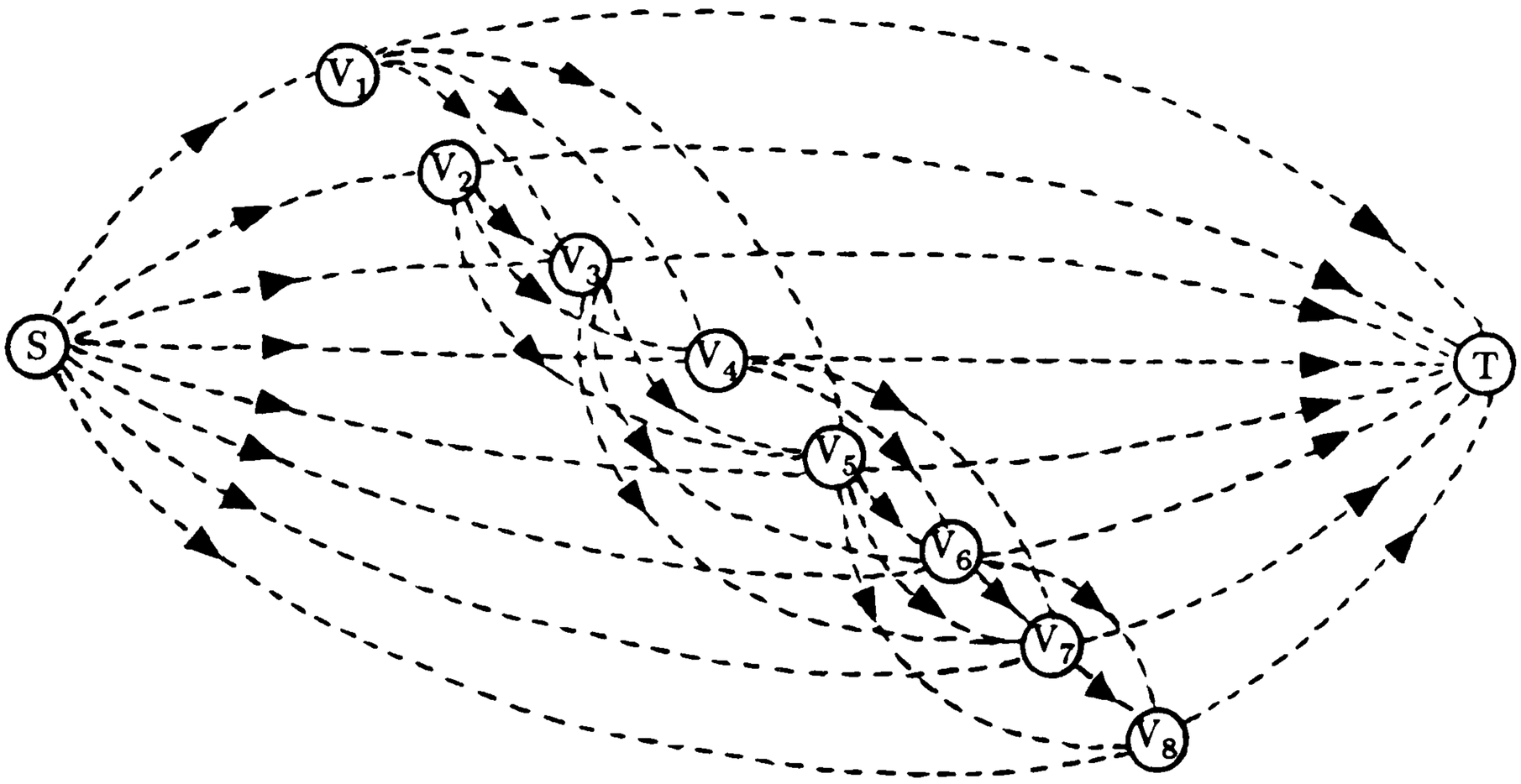
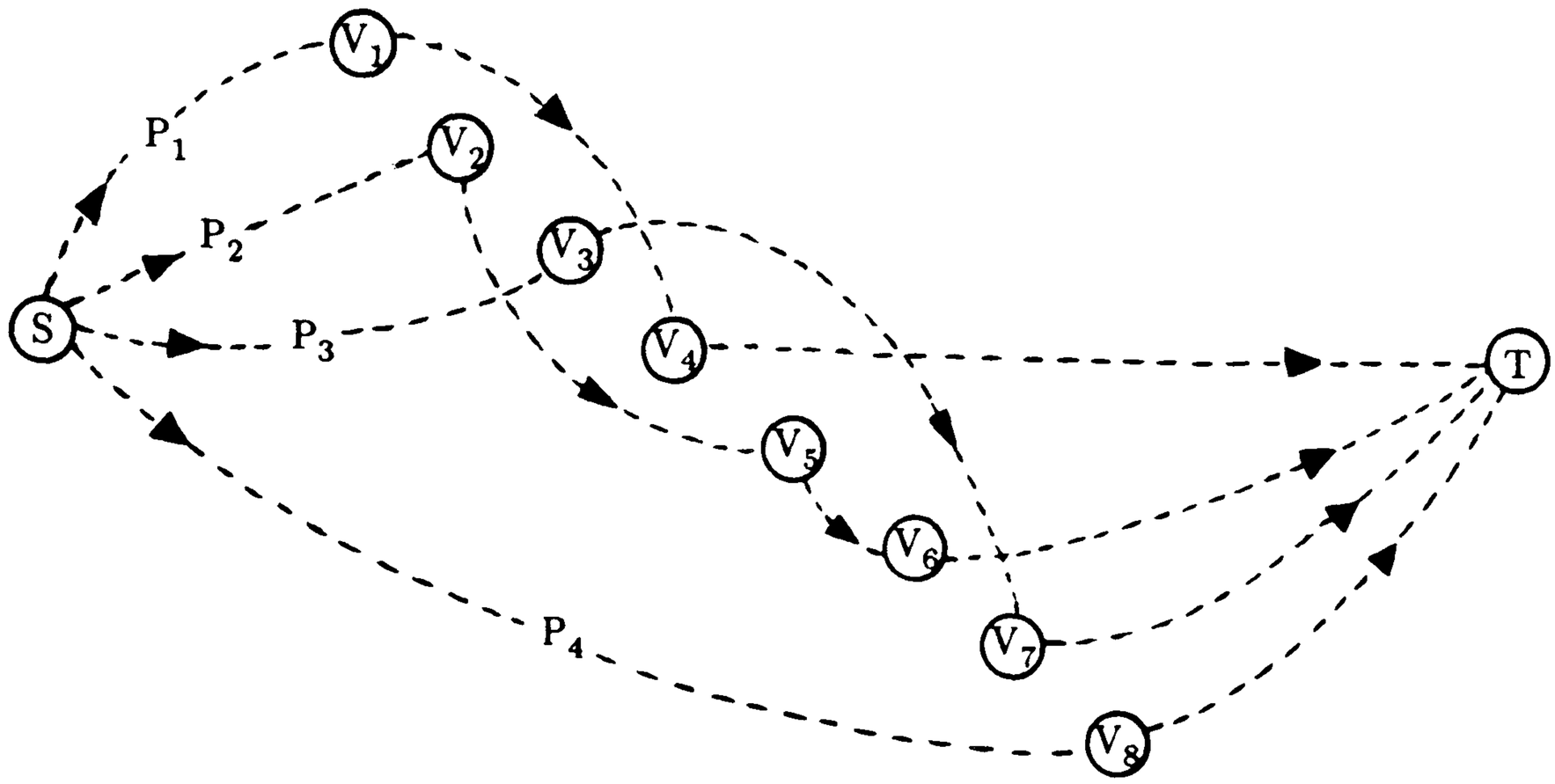


Figure 4-3. Initial Solution



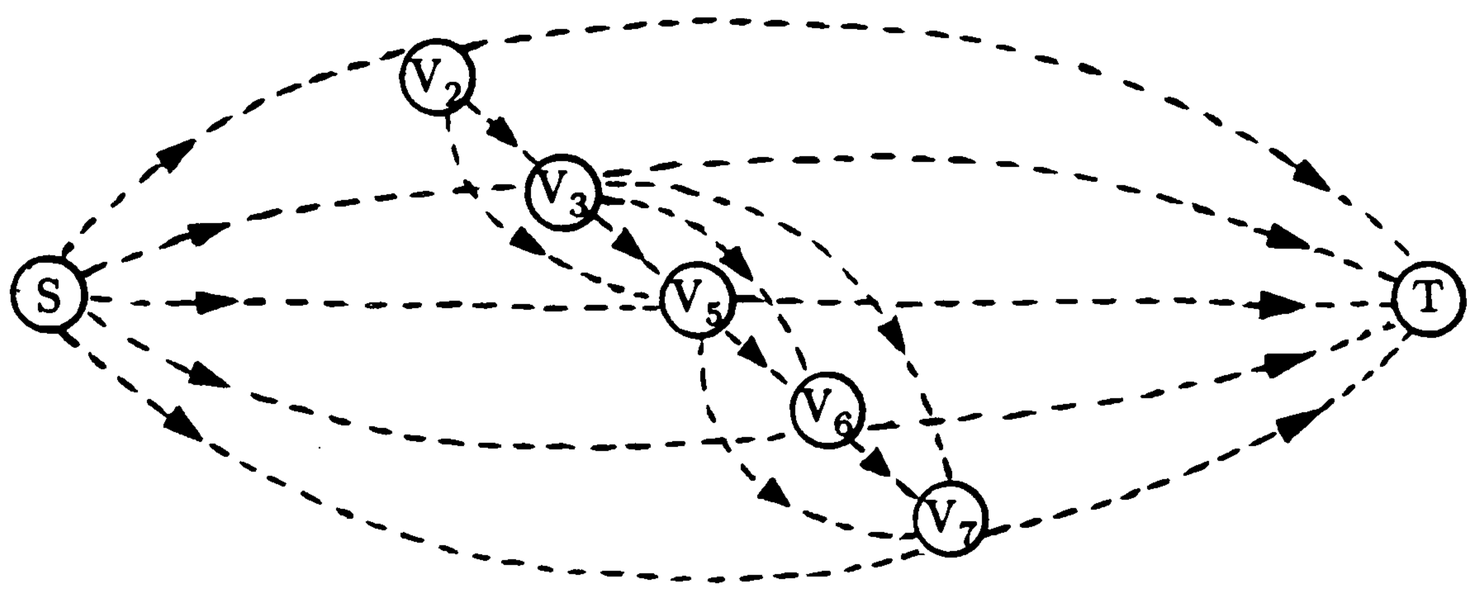
Now let us consider a subset $\mathbb{S}' \subset \{P_1, P_2, P_3, P_4\}$ and say we choose $\mathbb{S}'=\{P_2, P_3\}$. This subset now defines the set of vertices of G given by $N(\mathbb{S}')=\{V_2, V_3, V_5, V_6, V_7\}$. The induced subgraph $<P_2,P_3>$ is then the subgraph shown in Figure 4-4 and is defined by $<\mathbb{S}'> \equiv \{V_2, V_3, V_5, V_6, V_7\}$. Clearly paths P_2 and P_3 cover all the vertices of $<\mathbb{S}'>$ and are feasible in the sense that they satisfy the work-duty-time restrictions.

The question that immediately arises is then:

"**Can a feasible lower cost cover of the graph $<\mathbb{S}'>$ be found ?**" ---------- (4-8)

Note that the above question is **not** equivalent to asking for a lower cost set of columns in the SPP to cover the same set of rows as $\{P_2, P_3\}$. This question is restricted by the columns generated in the SPP, but the answer to question (4-8) is not.

Figure 4-4. Subgraph <P_2, P_3> Induced by Paths P_2 and P_3



4-4-1. The Search for a Local Improvement

In this section we discuss a procedure to attempt answer question (4-8) above.

Let us define a network flow problem on graph <$\mathbb{S}'$> from S to T, where the flow through each vertex V_j($\neq$ S or T) of <$\mathbb{S}'$> has an upper and lower bound of 1. Let the costs of the arcs (V_i, V_j) be the transition costs between flights i and j ($i,j \neq$ S,T) the cost of the arcs (S,i) be equal to the crew cost C_k and the cost of the arcs (i,,T) be 0. The minimum cost flow from S to T in <$\mathbb{S}'$> is then a collection of paths from S to T which cover every vertex V_j of <$\mathbb{S}'$> exactly once and minimizes the cost. (This corresponds precisely to the network-flow solution of the unconstrained CSP mentioned in Chapter 2).

(i) If the set of paths $\{\pi_1, \pi_2, ...\}$ corresponding to the above minimum cost network flow subproblem is the same as the set of paths in $\mathbb{S}'$, the answer to question (4-8) is

"no", since the set of paths in $\mathbb{S}'$ are optimal in the graph $<\mathbb{S}'>$.

(ii) If the set of paths $\{\pi_1, \pi_2, ...\}$ above are not the same as the set of paths in $\mathbb{S}'$ and they are feasible, (i.e. each π_r satisfies the work-duty-time constraint), the answer to question (4-8) is "yes" and $\mathbb{S}'$ can be replaced by $\{\pi_1, \pi_2, ...\}$ to provide a lower-cost solution to the CSP as:

$$\mathbb{S} \leftarrow \mathbb{S} \backslash \mathbb{S}' \cup \{\pi_1, \pi_2, ...\}$$

(iii) If the set of paths $\{\pi_1, \pi_2, ...\}$ above is infeasible, then question (4-8) cannot be answered directly. Indeed, it is no more than a restatement of the CSP for the subgraph $<\mathbb{S}'>$ instead of the original graph $G \equiv <\mathbb{S}>$.

Thus, any of the methods used to solve the original CSP in Chapters 2 and 3 can be used here for the much smaller problem. Moreover, computational methods that are not of the question for the original CSP become not only feasible, but trivially simple for the much-smaller problem defined by $<\mathbb{S}'>$. In our algorithm, we have applied the following procedure:

Step 0: Generate all feasible S to T paths in $<\mathbb{S}'>$ using the k-shortest path algorithm with k set to some small value. (We picked k=50 and used our implementation of Yen's k-shortest path algorithm. If not all feasible S to T paths have been generated in the first 50 shortest paths, we use only the set of feasible paths produced)

Step 1: Set up the set-partitioning problem whose rows correspond to the vertices in $N(\mathbb{S}')$ and whose columns correspond to the paths above.

Step 2: Solve the LP relaxation of the above SPP. If the solution is integer with columns (say) corresponding to paths $\{\pi_1, \pi_2, ...\}$ we have answered question (4-8) and have situation (i) or (ii) described earlier. Stop. If the LP solution is fractional we declare question (4-8) 'unanswered'. Stop.

Since our original intention is to try to replace a **small** subset $\mathbb{S}'$ of paths of the original heuristic solution by a better set $\{\pi_1, \pi_2, ...\}$, we always choose $\mathbb{S}'$ to be small, so the graph $<\mathbb{S}'>$ rarely contains more than a few (say 10) vertices. The generated number of columns in the SPP is also less than 50 (see above) and, therefore, the SPP has a maximum size of (10×50). The net result is that the algorithm in (ii) above terminates quickly and the LP solution is very often integer.

Example

For the example of $<\mathbb{S}'>$ given in Figure 4-4, let us say that the unconstrained solution is infeasible so that cases (i) and (ii) above do not apply. Then case (iii) involves the generation of the following paths: (Let us say the ones marked with a $*$ are infeasible)

S - V_j - T for j=2, 3, 5, 6, 7

S - V_2 - V_3 - T

S - V_2 - V_5 - T

S - V_3 - V_5 - T

S - V_3 - V_6 - T

S - V_3 - V_7 - T

S - V_5 - V_6 - T

S - V_5 - V_7 - T

S - V_6 - V_7 - T

S - V_2 - V_3 - V_5 - T

S - V_2 - V_3 - V_6 - T

* S - V_2 - V_3 - V_7 - T

S - V_2 - V_5 - V_6 - T

* S - V_2 - V_5 - V_7 - T

S - V_3 - V_5 - V_6 - T

S - V_3 - V_6 - V_7 - T

S - V_5 - V_6 - V_7 - T

S - V_2 - V_3 - V_5 - V_6 - T

* S - V_2 - V_3 - V_5 - V_7 - T

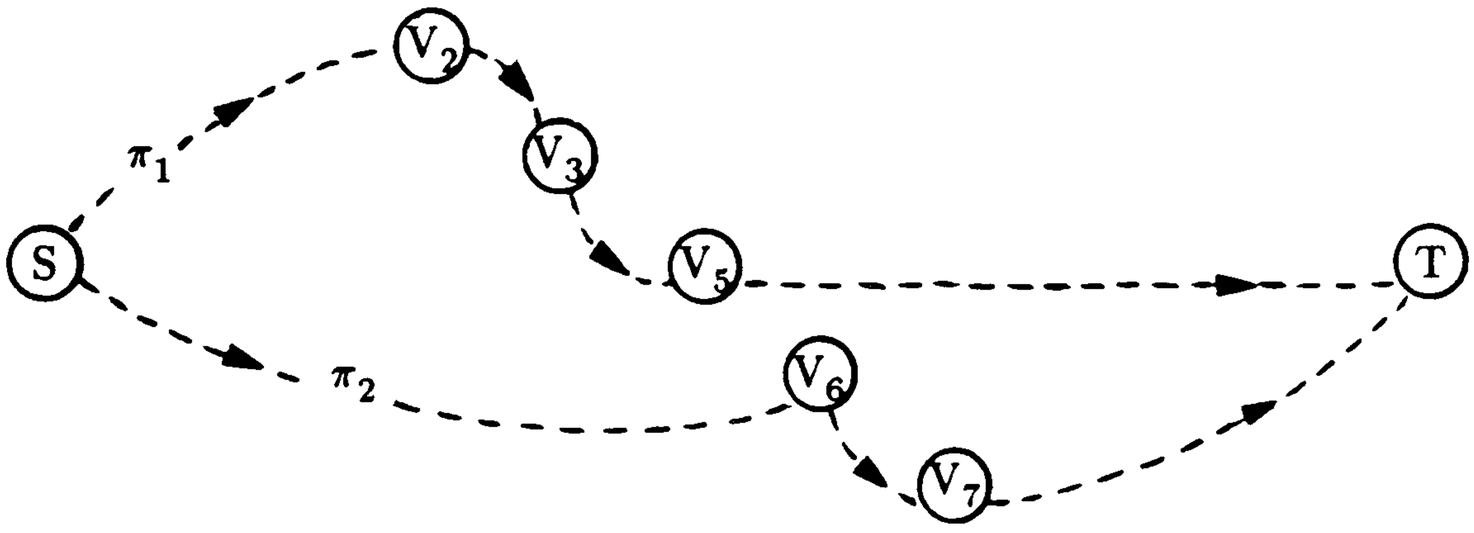
* S - V_2 - V_3 - V_6 - V_7 - T

* S - V_2 - V_5 - V_6 - V_7 - T

* S - V_2 - V_3 - V_5 - V_6 - V_7 - T

This leaves 15 paths (excluding the 7 single flight paths) from which we construct a 5-row-15-column SPP (the single-flight paths are considered implicitly). The solution could, for example involve columns π_1 = (S - V_2 - V_3 - V_5 - T) and π_2 = (S - V_6 - V_7 - T) as shown in Figure 4-5.

Figure 4-5. A New Optimal Cover $<\pi_1, \pi_2>$ of Graph $<P_2, P_3>$



4-4-2. The Choice of $\mathbb{S}'$

In Section 4-4-1 we started with an initial subset $\mathbb{S}'$ of the set $\mathbb{S}$ of paths representing the solution, defined the subgraph $<\mathbb{S}'>$, and attempted to produce a lower-cost cover of the vertices of $<\mathbb{S}'>$ using network-flow and LP relaxation procedures. One of the advantage of this procedure is (as mentioned above) that use can be made of paths in $<\mathbb{S}'>$ not represented by the generated columns of the initial SPP.

The question now remaining is:

"How do we choose the sets $\mathbb{S}'$?"

We wish for the set $\mathbb{S}'$ to possess certain properties that would make it more likely that a better alternative exists covering graph $<\mathbb{S}'>$. These properties include:

(i) $\mathbb{S}'$ is a collection of "short" paths rather than longer ones

This allows for the possibility that the graph $<\mathbb{S}'>$ can be covered by a set $\mathbb{S}^*$ of paths, with $|\mathbb{S}^*| < |\mathbb{S}'|$ thus producing a saving in crew costs as well. Thus, sets with short paths are preferable to consider for possible improvements.

(ii) $\mathbb{S}'$ should represent "one problem"

In particular $\mathbb{S}'$ should not contain a path P^* for which:

$$\min_{j \in P^*} [FT_j] \quad - \quad \max_{j \in (\mathbb{S}' \setminus P^*)} [ST_j] \; > \; WT \qquad \text{----------} \quad (4\text{-}9)$$

where $\mathbb{S}'$ and P^* are used to represent the sets of flights in the respective path(s). Note that if Constraints (4-9) is satisfied, the graph $<\mathbb{S}'>$ can be decomposed into two problems, since it is impossible for any flight in $\mathbb{S}'-P^*$ to be on the same path with any flight in P^*.

(iii) $\mathbb{S}'$ should represent a "complex problem"

Obviously, the chances of discovering a lower cost alternative cover in Section 4-4-1 are greater if the graph $<\mathbb{S}'>$ has large numbers of alternative covers. This would tend to be the case if the graph $<\mathbb{S}'>$ is dense, and hence we wish to consider paths for inclusion into $\mathbb{S}'$ which have as much overlap with each other as possible.

(iv) $\mathbb{S}'$ should be a "poor" set of paths

Quite clearly the chances of improving a "poor" (high-cost) set of paths are greater, and the size of the improvement is likely to be greater too. Thus, we wish $\mathbb{S}'$ to contain "poor" paths. We have chosen the measure:

$$M_P = \frac{\text{Total transition cost of path P}}{\text{Total duration of path P}}$$

to indicate the quality of a path. The lower the value of M_P, the better the path. Note that the "total duration" of path P is its ending time minus the starting time (i.e. includes both flight times and transitions).

In our procedure we listed each path in the solution S in ascending order of path length {criterion (i); and, (separately), in ascending order of the measure M_P (criterion (iv))}. A final list L is then produced in order of their positions in the previous two lists (in ascending order of the product of their positions in the two lists). This list L is then used to pick subsets $\mathbb{S}'$, according to the following algorithm.

Step 0: Initialization

Starting from the list L, initialize for the generation of all subsets of L in

lexicographical order. Set Δ and MAXCOUNTER (See Note a and b).

Step 1: Generate subset

Generate next subset $\mathbb{S}'$ *[Nijenhuis and Wilf 1978]*. If "enough" subsets have been examined, declare the local optimization procedure finished, and **Stop**.

Step 2: Test subset

Test the subset $\mathbb{S}'$ against Constraint (4-9) of criterion (ii) and test that the path overlap is above a threshold Δ chosen to impose criterion (iii). Test that the number of flights in the paths of the subset is $\leq$ 10.

Step 3: Iteration counter

If any of the above conditions fail,update COUNTER $\leftarrow$ COUNTER+1.

If COUNTER > MAXCOUNTER, skip all supersets of the existing subset, set COUNTER $\leftarrow$ 0 and go to step 1.

Else go to step 1.

Else go to step 4.

Step 4: Local optimization

Apply local optimization procedure to $<\mathbb{S}'>$ as explained in Section 4-4-1. If an improvement has been found, replace the solution S with the better solution, recompute list L and go to step 0. Else go to step 1.

Notes: (a) The "overlap" in a set $\mathbb{S}'$ is measured as the ratio:

$$\frac{\text{Total duration of all paths in } \mathbb{S}'}{\left(\underset{j \in \mathbb{S}'}{\text{Max}} [\text{FT}_j] - \underset{j \in \mathbb{S}'}{\text{Min}} [\text{ST}_j] \right)}$$

The larger the above ratio is, the larger the overlap.

In the computational tests, we required this ratio to be $> \Delta \equiv 2$ (see step 2).

(b) A counter is kept (COUNTER) denoting the number of consecutive attempts by the subset generator to produce a "good" subset to try. If more than MAXCOUNTER attempts fail, all supersets of the existing set $\mathbb{S}'$ (in the lexicographical order) are abandoned and a new subset is generated.

In the computational tests we set MAXCOUNTER=10

(c) The lexicographic subset-generation procedure described in *[Nijenhuis and Wilf 1978]* was used at step 1.

(d) At step 1 a maximum limit on the number of attempts to generate sets $\mathbb{S}'$ and test them for improvement is imposed. In our computational tests we imposed a maximum time limit (of 20 minutes), instead.

4-5. Computational Results

We tested the heuristic described in this chapter on the standard set of 250 problems described in Chapter 2. Computational experiments were again performed on an IBM PC/AT using the Microsoft FORTRAN 4.1 compiler. In addition, we tested the algorithm on larger CSP's (with up to 210 flights) also generated randomly, but with a bias towards the structure (flights per rotation, etc.) found in airline crew

scheduling problems. Obviously, for the larger problems, we cannot compare the heuristic solution against an optimal one since optimal answers cannot -in general- be found. However, for these larger problems, we can comment on the computational time taken by the heuristic.

For the 250 smaller test problems, the computational results are shown in Table 4-1. The initial heuristic solved an average of 71% of the 250 problems optimally. Of the remaining problems the heuristic obtained an answer to within 4.7% of the optimum. The local optimization procedure examined possible improvement to sets $\mathbb{S}'$ varying in number from about 5 for the smaller problems to about 50 for the larger ones. In these attempts, the number of successful improvements varied from 1 or 2 for the smaller problems to about 10 for the larger problems.

Total computing time for the complete algorithm varied with problem size but reached approximately 10 minutes for the larger problems.

Table 4-1. Computational Results of Heuristic

No. of Flights[1]	Initial Heuristic		Local Optimization			Time[7]
	Ratio-I[2]	Gap(%)[3]	Ratio-II[2]	Set No.[4,6]	Improve[5,6]	
6	90	1.5	100	3	1	21.9
7	90	1.1	100	6	1	28.0
8	80	1.6	100	4	1	29.1
9	90	0.9	100	3	1	31.8
10	80	2.1	100	5	1	44.0
11	80	3.7	100	7	2	43.7
12	90	1.8	100	6	1	52.0
13	90	0.4	100	6	1	57.3
14	60	7.2	100	8	3	63.5
15	70	5.2	100	8	2	70.6
16	80	3.6	100	5	2	76.8
17	80	4.2	100	9	2	84.3
18	90	1.6	100	16	1	91.6
19	50	11.7	100	26	4	96.9
20	90	2.4	100	7	2	94.8
21	80	3.5	100	20	2	98.5
22	80	2.2	100	13	2	105.4
23	60	8.7	100	34	6	112.5
24	70	4.9	100	25	3	109.3
25	70	3.7	100	21	3	115.9
26	90	1.3	100	17	2	118.8
27	40	15.3	100	48	7	122.1
28	70	7.3	100	32	6	125.2
29	60	12.8	100	44	8	128.9
30	50	10.6	100	31	12	132.7

1) 10 problems of each flight
2) Percentage of problems solved optimally
3) Average error in problems of that size not solved optimally
4) Number of sets $\mathbb{S}'$ tried
5) Number of improvements
6) Average over the problems unsolved by initial heuristic
7) IBM PC/AT sec.

Table 4-2 shows the same computational details but for much larger problems. Ten test problems from 30 flights to 210 flights were tested. The cut-off limit for these problems was set at 20 minutes of IBM PC/AT time. From this table it is quite apparent that the heuristic (especially the local optimization procedure) provides an excellent way of solving large CSP's approximately.

Table 4-2. Computational Results of Heuristic for Large CSP's

No. of Flights	Gap(%)[1]	Local Optimization		Time[4]
		Set No.[2]	Improve[3]	
30	12.3	37	9	153
50	14.6	48	11	324
70	14.1	31	16	411
90	17.2	49	24	679
110	13.0	57	12	1048
130	16.4	55	19	1200*
150	19.9	49	22	1200*
170	23.1	44	28	1200*
190	21.7	66	21	1200*
210	22.2	60	18	1200*

1) Difference of initial heuristic from final answer

2) Number of sets $\mathbb{S}'$ tried

3) Number of improvements

4) IBM PC/AT sec.

*) Time limit

4-6. Conclusion

This chapter has described a heuristic procedure for the CSP which is composed of two parts. The first part involves a heuristic solution to a standard SPP formulation of the CSP. The second part is a novel and highly effective (see computational results) procedure which can detect improvements to the solution of the CSP - beyond those possible from even the optimal solution to the SPP - based on network flow and LP relaxation of small CSP problems (defined by subgraphs) induced by parts of the solution. The complete heuristic provides a effective means of solving practical-size CSP's.

Chapter 5

A TREE SEARCH ALGORITHM FOR THE CSP

5-0. Summary

In this chapter we introduce the computation of a lower bound for the CSP based on the new combined concept of k-best solution and Lagrangean Relaxation. This combination provides a powerful method of computing tight bounds based on the Assignment Problem (AP). This bound is incorporated into a tree search procedure, similar in concept to those used successfully for the Travelling Salesman Problem (TSP). The final algorithm is tested on large CSP's (up to 100 flights) and shown to be able to solve such problems in under 1 hour computing time on the IBM PC/AT. The resulting algorithm is considered to be practically useful.

5-1. Introduction

In the previous 3 chapters we discussed:

(i) An exact algorithm based on graph expansion (originally proposed by Cheddad) with a modified set of constraints derived from cliques; which lead to a strong surrogate relaxation. This surrogate problem, when solved as an LP, often leads to integer feasible (and hence optimal) solutions, and only requires a very small tree search when the LP solution is fractional.

(ii) An exact algorithm based on the unexpanded graph (with fewer variables than in (i) above) and with constraints added iteratively to an LP relaxation of the problem. The LP solution is used to guarantee these constraints which are then "strengthened" and which, together with a set of "insurance" constraints (generated by making use of a k-shortest path algorithm) are used to solve the problem optimally in a tree search procedure.

(iii) A heuristic algorithm based local optimization with a network flow algorithm being used to identify improvements to a starting solution. This heuristic has never failed to obtain the optimal solution on every problem tried (when the value of the optimum is known).

In the present chapter we investigate a tree search procedure for the exact solution of the CSP. We start from the formulation of the CSP given in Section 3-1 (but excluding crew costs C_k (i.e. replacing X_k by K) and which is repeated below for convenience:

$$\text{minimise} \quad Z = \sum_{(i,j)\in N} C_{ij}X_{ij} \qquad \text{(5-1)}$$

$$\left.\begin{aligned}
&\text{subject to} \quad \sum_{j=1}^{n} X_{0j} = K \\
&\sum_{j=0}^{n} X_{ij} = 1, \qquad \text{for } i = 1, \ldots, n \\
&\sum_{i=0}^{n} X_{ij} = 1, \qquad \text{for } j = 1, \ldots, n \\
&\sum_{i=1}^{n} X_{i0} = K
\end{aligned}\right\} \qquad \text{(5-2)}$$

$$\sum_{(i,j)\in P^-} X_{ij} \leq |P^-| - 1 \qquad \forall P^- \in \mathfrak{F}^- \quad \text{----------} (5\text{-}3)$$

$$X_{ij} \in \{0,1\} \qquad \text{for } 1 \leq i < j \leq n \quad \text{----------} (5\text{-}4)$$

The formulation given by (5-1) ~ (5-4) above, is very similar to the formulation of the Travelling Salesman Problem (TSP) in two respects:

(i) Constraints (5-2) which are of Transportation Problem variety (indeed they are assignment constraints if the super-source and super-sink index 0 is repeated K-times, say 0_1, 0_2, ..., 0_K) are also in the TSP formulations.

(ii) Constraints (5-3) are identical to the subtour elimination constraints of the TSP, except for the fact that the subsets P^- in (5-3) represent ""subtour-subsets" in the TSP. Indeed (5-3) is a smaller set of constraints since P^- is restricted to be an element of $\mathfrak{F}^-$, whereas in the TSP the subtour-subsets are unrestricted subsets of {1, 2, ..., n}.

Note that the Assignment Problem (AP) solution represented by (5-1), (5-2) and (5-4) cannot have subtours in this case, since the underlying graph G is acyclic.

Because of this similarity, we will investigate in this chapter an approach to the CSP which is motivated by algorithms for the TSP.

This approach is a dual-based tree search algorithm making use of the solution to an AP as a lower bound and also using the solution to direct the tree search. This approach is called "dual" because at any intermediate node of the tree search, no feasible solution exists, and the motivation for branching is to reduce the infeasibility.

No other principle of duality is implied here.

In the above approach, we make use of very simple principle that has been shown to be extremely successful for the vehicle routing and other combinatorial problems by Christofides *[1991]*, Christofides and Mingozzi *[1992]*, Christofides and Weston *[1992]* and Christofides and Hadjiconstantinou *[1992]*, namely the use of k-best optimal solution to relaxed problems, in an attempt to generate an optimal solution that is feasible to constraints that have been relaxed.

5-2. Lagrangean Relaxation and k-best Optimal Solution

Before discussing the specific algorithms we will make some general comments. Consider the general integer program:

$$\left.\begin{array}{lll} \text{minimise} & Z = \underline{cx} \text{ ---------- (5-5)} & \\ \text{subject to} & A\underline{x} \geq \underline{b} \text{ ---------- (5-6)} & \\ & D\underline{x} \geq \underline{e} \text{ ---------- (5-7)} & \\ & \underline{x} \in \{0,1\} \text{ ---------- (5-8)} & \end{array}\right\} \text{(P)}$$

where $\underline{c}$, $\underline{b}$, $\underline{e}$ *are* $[1 \times n]$, $[m_1 \times 1]$, $[m_2 \times 1]$ vectors respectively and A, D are $[m_1 \times n]$, $[m_2 \times n]$ matrices.

Let $\underline{x}^*$ be the optimal solution to problem (5-5) ~ (5-8) and consider the Lagrangean -relaxed problem:

$$\left.\begin{array}{ll} \text{minimise} & Z_R = \underline{cx} + \underline{\lambda}(\underline{b}-A\underline{x}) \text{ ---------- (5-9)} \\ \text{subject to} & D\underline{x} \geq \underline{e} \text{ ---------- (5-10)} \\ & \underline{x} \in \{0,1\} \text{ ---------- (5-11)} \end{array}\right\} (P_R)$$

where $\underline{\lambda}$ is a $[1 \times m_1]$ non-negative row vector.

Since $\underline{x}^*$ is feasible for problem (P) it is also feasible for (P_R) and its cost is $Z_R(\underline{x}^*) = \underline{cx}^* + \underline{\lambda}(\underline{b}-A\underline{x}^*) \leq \underline{cx}^*$, since $\underline{b}-A\underline{x}^* \leq 0$. Hence, we get the well-known result that the minimum value of Z_R for problem P_R is a lower bound on the minimum value of Z for problem P.

Now consider a sequence of k solutions to problem P_R: $\underline{x}_R^1$, $\underline{x}_R^2$, ..., $\underline{x}_R^k$, being the k-best (feasible) solutions.

Case (i) Let $\underline{x}^* \neq \underline{x}_R^i$ for any i=1, ..., k (for example if $\underline{x}_R^1$, $\underline{x}_R^2$, ..., $\underline{x}_R^k$ are all infeasible to P). Then, by definition;

$$Z_R(\underline{x}^*) = \underline{cx}^* + \underline{\lambda}(\underline{b}-A\underline{x}^*) \geq Z_R(\underline{x}_R^k)$$

$$Z_R(\underline{x}_R^k) \leq \underline{cx}^* \text{ ----------(5-12)}$$

i.e. the value of the k-best solution to the relaxed problem P_R is also lower bound to the original problem P_R

Case (ii) Consider the sequence $\underline{x}_R^1$, $\underline{x}_R^2$, ... $\underline{x}_R^k$ and the sequence ($\underline{x}_R^{a1}$, $\underline{x}_R^{a2}$, ... $\underline{x}_R^{ap}$) of solutions feasible to problem P, (i.e. satisfying constraints $A\underline{x} \geq \underline{b}$).

Let α^* be such that

$$\underline{cx}_R^{\alpha^*} = \min_{j=1,\ldots,p} [\underline{cx}_R^{\alpha_j}] \quad \text{----------(5-13)}$$

Then, either: $\underline{x}^* = \underline{x}_R^{\alpha^*}$ (i.e. solution $\underline{x}_R^{\alpha^*}$ is optimal)

or: none of the solutions in the sequence are optimal, in which case:

$$Z_R(\underline{x}^*) = \underline{cx}^* + \underline{\lambda}(\underline{b}-A\underline{x}^*) \geq Z_R(\underline{x}_R^j) = \underline{cx}_R^j + \underline{\lambda}(\underline{b}-A\underline{x}_R^j)$$

$$\underline{cx}_R^j \leq \underline{cx}^* - \underline{\lambda}(\underline{b}-A\underline{x}_R^j) \quad \text{for all } j=1, \ldots, k \quad \text{----------(5-14)}$$

5-2-1. A Test for Optimality

Consider solution $\underline{x}_R^{\alpha^*}$ identified be Equation (5-13) above. If this solution satisfies Constraints (5-6) with equality, (i.e. $A\underline{x}_R^{\alpha^*} = \underline{b}$), or if $\underline{\lambda}(\underline{b}-A\underline{x}_R^{\alpha^*})=0$, anyway, (i.e. $\lambda_r=0$ for any constraint $A^r\underline{x} \geq \underline{b}^r$ *not* satisfied with equality), then Constraint (5-14) becomes:

$$\underline{cx}_R^{\alpha^*} \leq \underline{cx}^*$$

and since $\underline{x}^*$ is optimal solution, $\underline{x}_R^{\alpha^*}$ must also be the optimal solution.

5-2-2. Derivation of a Lower Bound

We know that $\underline{x}^*$ cannot be any of the solution $\underline{x}_R^i$ for i=1, ..., α^*-1. Hence we can write:

$$Z_R(\underline{x}^*) = \underline{c}x^* + \underline{\lambda}(\underline{b}-A\underline{x}^*) \geq Z_R(\underline{x}_R^{\alpha^*})$$

$$Z_R(\underline{x}_R^{\alpha^*}) \leq \underline{c}x^*$$

i.e. $Z_R(\underline{x}_R^{\alpha^*})$ is **a lower bound to the value of problem P**

The above lower bound, and the possibility of generating a feasible solution (when k is large) and proving it to be optimal, are very significant improvements to the capability of Lagrangean Relaxation to solve combinatorial optimization problems. Clearly, subgradient ascent can still be used to pick the best value of $\underline{\lambda}$ in order to maximize the value of the lower bound. Indeed, prior to generating the k-best solutions to the relaxed problem P_R we assume that the subgradient ascent has been performed and that the λ's in question are the result of this ascent.

5-3. Computation of Lower Bound

Consider the solution of the Assignment Problem (AP) defined by (5-1), (5-2) and (5-4) using, for example, the "Hungarian Algorithm". If the paths corresponding to the optimal solution $\widehat{X} \equiv [\widehat{X}_{ij}]$ of this AP are all feasible, then $\widehat{X}$ is the optimal solution to the CSP. If not, let P_1^-, ..., P_l^- be some paths not satisfying the work-duty-time

constraints. Constraints (5-3) can now be used to produce the Lagrangean problem:

$$R \begin{cases} \text{minimise} \quad Z(\lambda) = \sum_{(i,j)\in N} C_{ij}X_{ij} + \sum_{r=1}^{l} \lambda_r \left(\sum_{(i,j)\in P_r^-} X_{ij} + 1 - |P_r^-| \right) \quad \text{----------(5-15)} \\ \text{subject to} \quad \text{Constraints (5-2) and (5-4)} \end{cases}$$

The Lagrangean multipliers are initially picked arbitrarily, but with values that are proportional to the amount by which the corresponding paths P_r^- violates the work-duty-time restriction.

Problem R can now be resolved as an AP to produce a new solution $[\hat{X}_{ij}]$. If the new solution generates some paths P_{l+1}^-, ..., P_q^- which are infeasible as far as the work-duty-time is concerned, these paths are again incorporated into the relaxed problem R above by relaxing in a Lagrangean way their corresponding Constraints (5-3). (note that some of these "new" paths may have been generated previously, in which case their corresponding λ is simply increased).

The above process of solving a cost-modified AP, iteratively whilst adding constraints in a Lagrangean way and modifying the multipliers λ, is an ascent procedure for improving the bound attained from the AP. One way of performing this ascent is to apply the usual subgradient optimization procedure described in Section 2-1-1, Chapter 2, but utilizing (for $A\underline{x} \geq \underline{b}$, and for computing the step size t_k) only these constraints that have already been generated.

Using this procedure (and starting with all $\underline{\lambda}^0$=0) the ascent for a 30-flight CSP over 15 iterations is shown in Figure 5-1. At the end of this ascent, a lower bound of 266 was obtained which is 5.33% from optimality.

As described in Section 5-2, this bound can be improved by generating the k-best solution to the AP (with the values of λ at the 15-*th* iteration) for a suitably large value of k. Figure 5-2 shows the further ascent of this bound from the initial value of 266 as the value of k is increased up to k=18. At k=18, a second feasible solution was found which did **not** satisfy the test for optimality of Section 5-2-1. However, the lower bound produced from this solution was 280.3, and the cost of the solution was 281, hence (with integer data) its optimality is proven.

In the computations to obtain this bound, and in all the other computational results given in this Chapter, the k-best solutions to the AP have been produced using an algorithm due to Lawler *[1972]* which is a generalization of an earlier algorithm due to Murty *[1968]*. Better algorithms for finding the k-best solutions to an AP have been proposed by Derigs *[1983]* based on the principle of "exchanges", but these were not used in this thesis. We should note, however, that for all the above procedures, the worst-case result is for computing time to increase linearly with k.

Figure 5-1. Results of Lagrangean Bound After 15 Subgradient Iterations for a 30-flight CSP

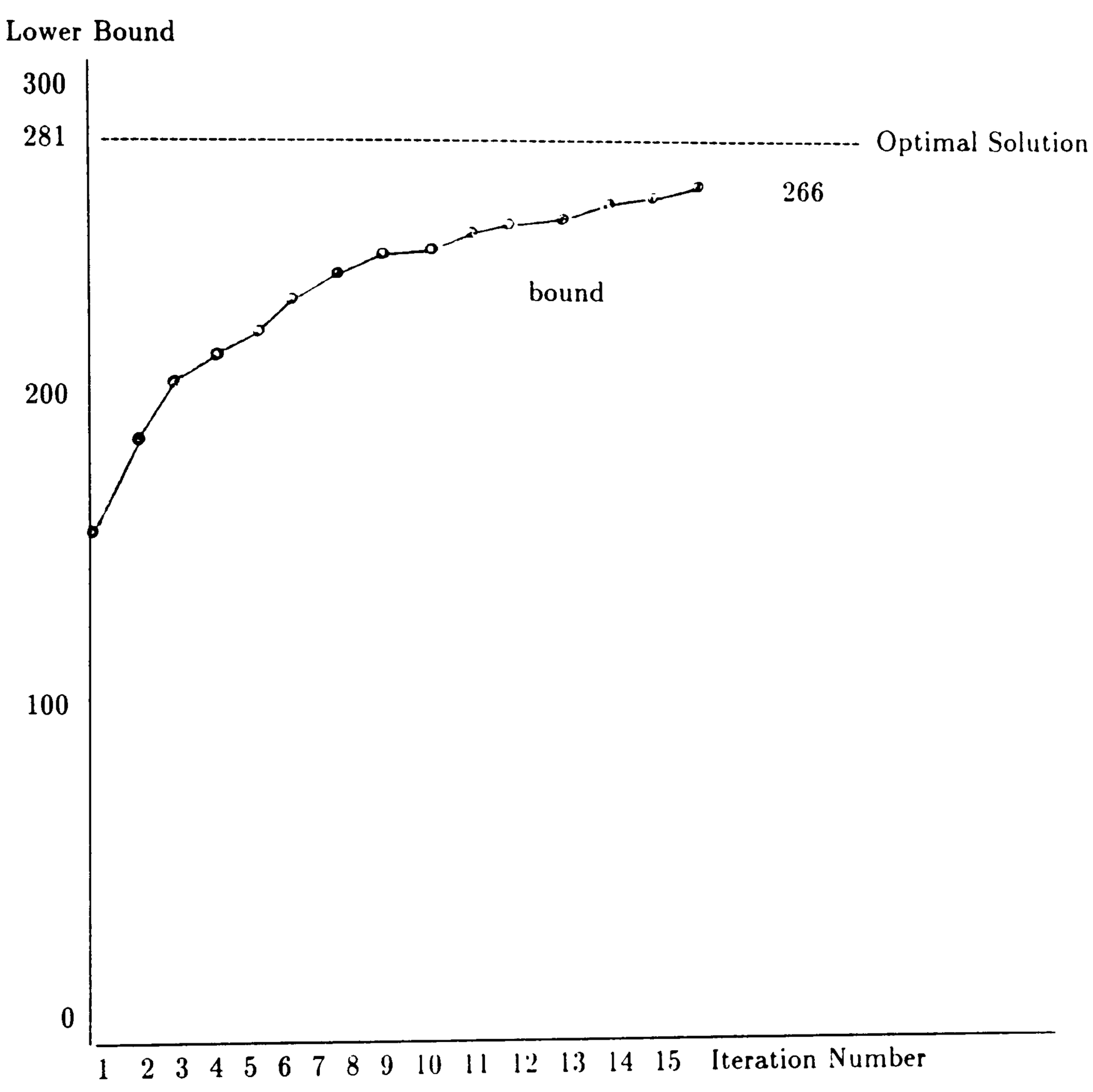
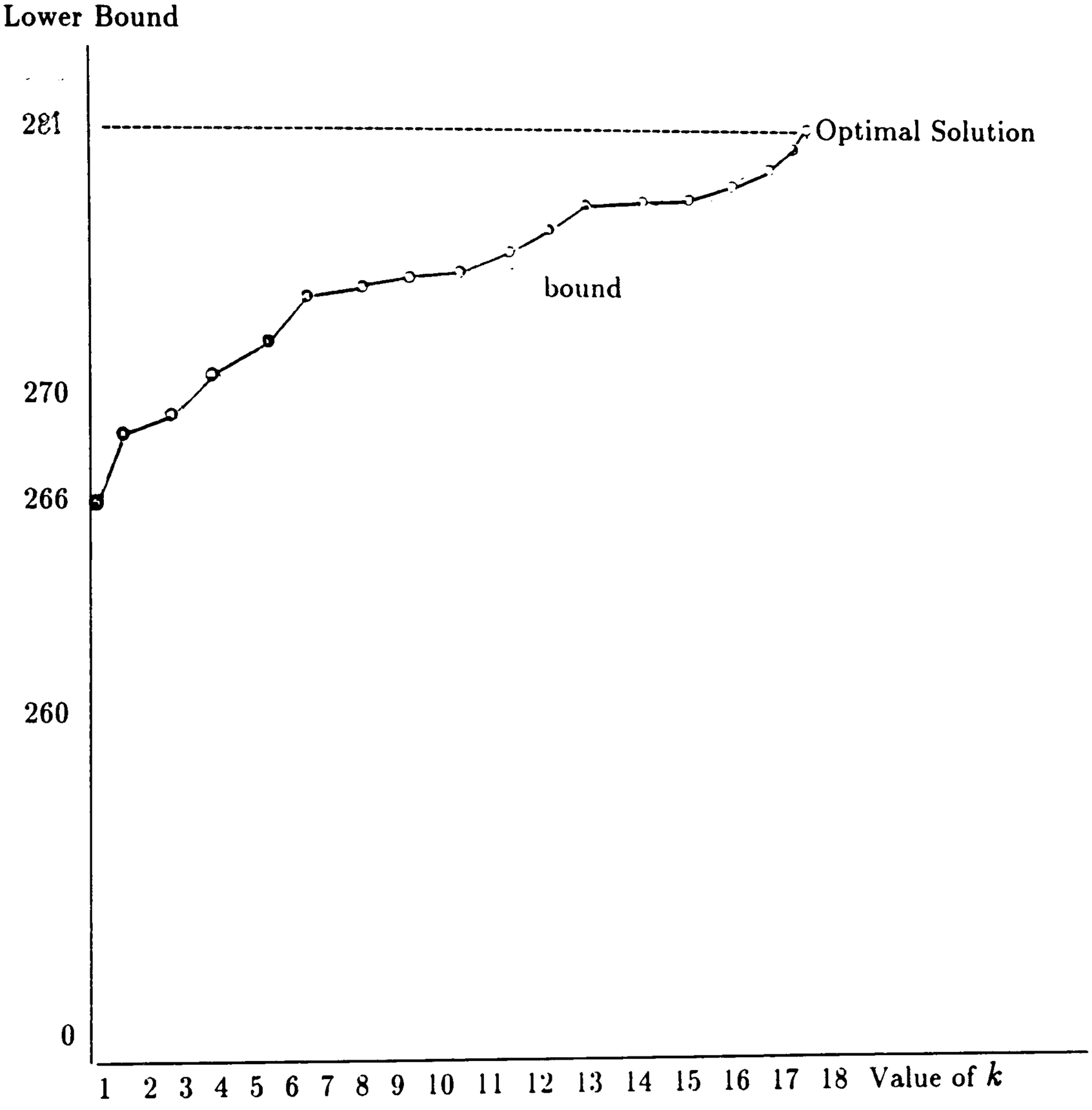


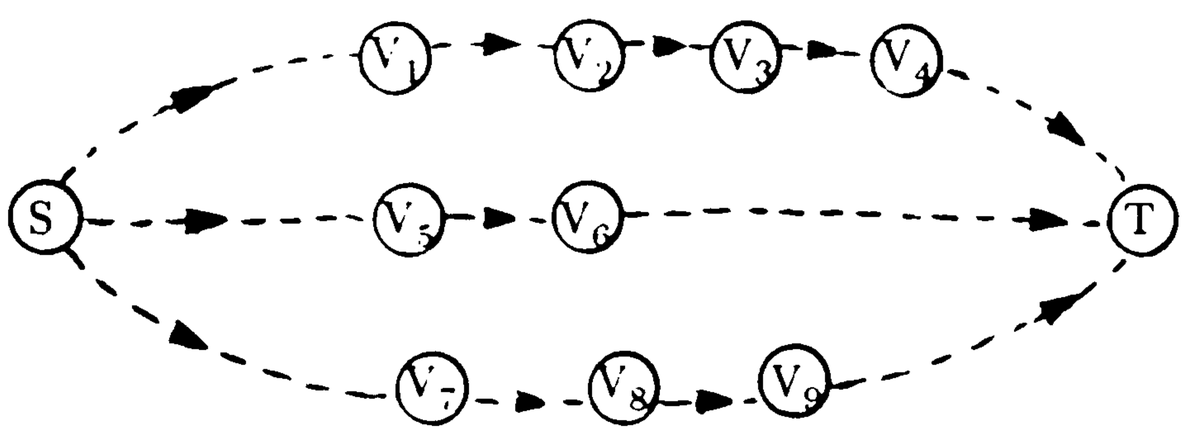
Figure 5-1. Increase in Lagrangean Bound with Value of k
After Finding k-best Solutions to AP



5-4. A Dual Tree Search Algorithm

Let $[X^0_{ij}]$ be the solution to the AP at the end of the Lagrangean ascent. This solution is represented by Figure 5-3.

Figure 5-3. The Solution to the AP at the Root Node



Let us say that path $P_1^- = (S, V_1, V_2, V_3, V_4, T)$ is identified to violate the work-duty-time constraint. Constraint (5-3) then becomes:

$$X_{12} + X_{23} + X_{34} \leq 2 \qquad \text{(5-16)}$$

and let us say that this constraint does not decompose into a set of more elementary constraints as mentioned in Section 3.4.1, Chapter 3, (otherwise we could deal with a decomposed constraint.)

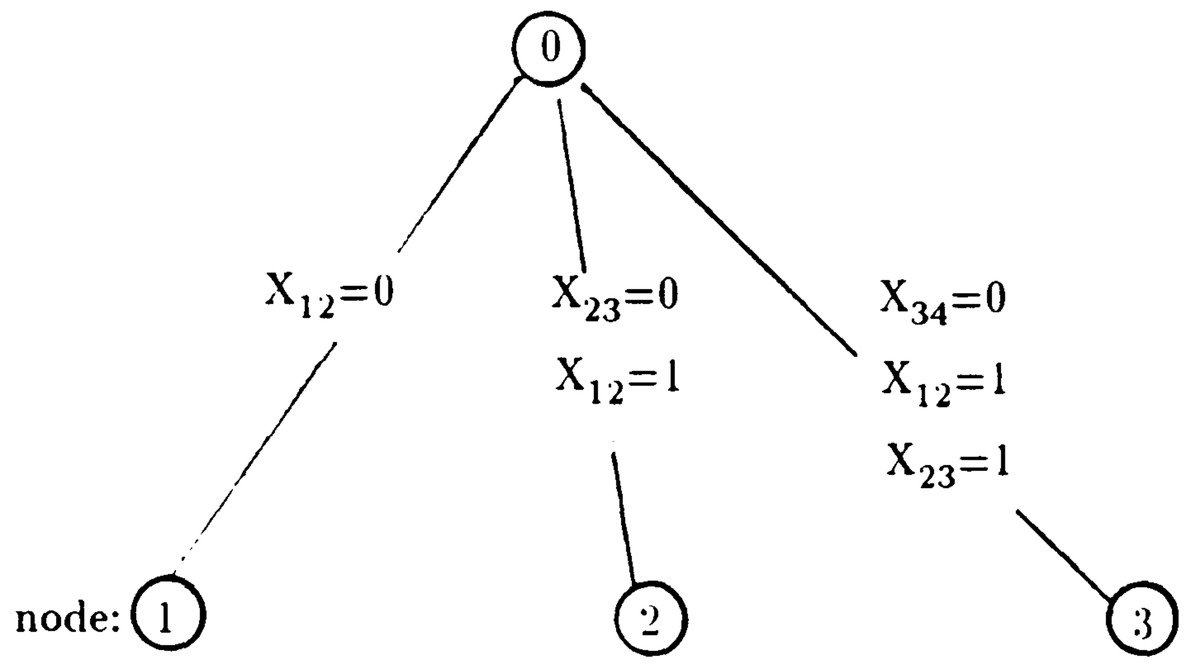
Constraint (5-16) can be imposed by branching on variables X_{12}, X_{23} and X_{34} to produce 3 subproblems (See the search tree in Figure 5-4) represented by tree nodes:

Node 1: $X_{12}=0$

Node 2: $X_{23}=0$, $X_{12}=1$

Node 3: $X_{34}=0$, $X_{12}=1$, $X_{23}=1$

Figure 5-4. The First 3 Nodes of the Tree Search



Let $[X_{ij}^1]_1$, $[X_{ij}^1]_2$, ..., $[X_{ij}^1]_k$ be the k-best solutions to the AP represented by node 1. If any of these are feasible, we may have an improved solution, or we may be even able to prove its optimality with respect to AP represented by node 1. If its lower bound is computed (See Section 5-3) to be higher than the best answer so far, the node is abandoned; otherwise branching continues from there in the same way as was done from node 0. For example, if both the optimality test and the bound test have failed and the solution $[X_{ij}^1]_1$ is infeasible because of path $P^-=(S, V_5, V_2, V_3, V_4, T)$ as shown in Figure 5-5, the constraint

$$X_{52} + X_{23} + X_{34} \leq 2$$

can be imposed by branching on nodes:

Node 4: $X_{52}=0$

Node 2: $X_{23}=0$, $X_{52}=1$

Node 3: $X_{34}=0$, $X_{52}=1$, $X_{23}=1$

to produce the tree shown in Figure 5-6.

Figure 5-5. The Solution to the AP at Node 1

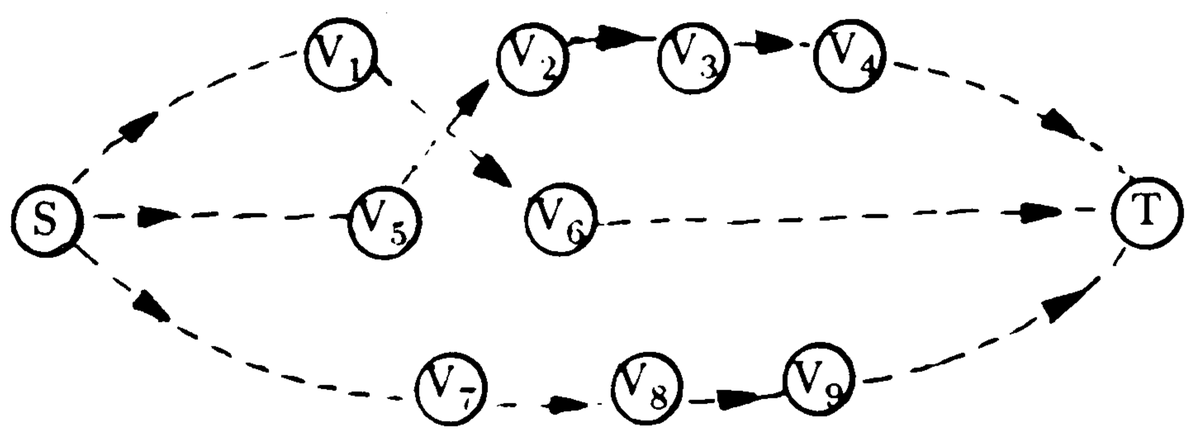
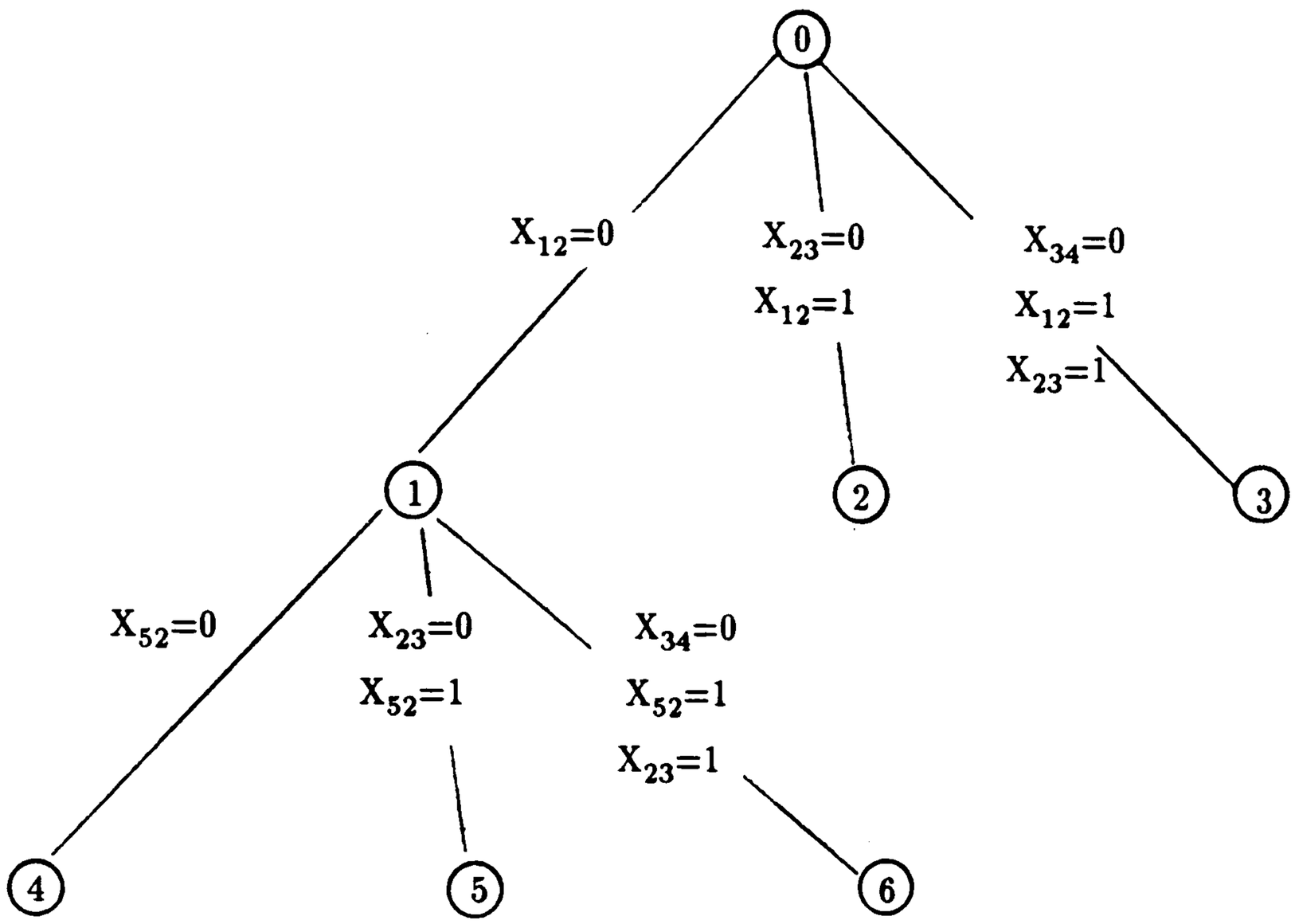


Figure 5-6. The Search Tree after Branching from node 1



The above branching continue until the entire tree is searched.

We should mention here, that the enumeration of the *k*-best solution to the AP at each node of the tree is performed for the values of λ computed at the root node. This enumeration of *k*-best solutions reduces the size of the tree enormously, both because the lower bound computed is much tighter, but also because it is often the case that a feasible solution of value equal to the lower bound (after rounding) is obtained. (See computational results - Section 5-6)

In the computational tests reported in the next section we implemented a depth-first tree search where the node to branch-from next was chosen to be that of the last-generated nodes, with the lowest lower bound. If, more than one path P^- which is infeasible is identified at any one stage, we chose to branch on that infeasible path with

the lowest cardinality, since this is both likely to be the strongest constraint, and also the one which requires the smallest number of new nodes to be generated.

5-5. Computational Results

We tested the algorithm of this chapter on the 250 test problems used throughout this thesis, and on 10 additional random problems of larger size, up to 100 flights. Table 5-1 shows the results of the bound computation at the root node with k chosen as 200. Table 5-2 shows the results of the tree search for those problems that required a tree search. For all non-root nodes, the value of k was chosen as 20. From Table 5-1 it is seen that it is very worthwhile to enumerate the k-best solutions to the AP, as far as improvements to the quality of the lower bound are concerned, even though computing times at the root node tend to be almost 50% of the total. Table 5-2 shows that the heuristic solution produced by the heuristic in Chapter 4 is excellent having obtained the optimal solution in every case. The total computing times do **not** include the time to obtain this heuristic solution, but include both the time at the root node and at other nodes of the tree search. Note that the increase in both computing time and number of tree nodes are not excessive.

Table 5-3 shows the computational results of 10 additional CSP's from 10 to 100 flights. These problems are also generated randomly (in the same way as the other 250 test problems) but with a structure found most often in airline crew scheduling problem. Again, the number of nodes generated is not excessive nor are the computation times, although they are now close to 1 hour computing time for the 100-flight problem. The heuristic solution for all the above problems was also equal to the optimum value for all cases.

Table 5-1. The Lower Bound at the Root Node for 250 CSP's

Number[1] of Flights	Initial[2] AP Bound	Bound[2,3] after Ascent	Bound[2] after K-best Solutions	Total[4] Computing Time
6	10.3	1.2	0.2	4
7	11.4	0.8	0.2	5
8	9.1	1.8	0.3	7
9	8.7	1.4	0.3	7
10	8.4	0.5	0.1	12
11	8.4	2.6	0.4	16
12	9.9	1.8	0.4	22
13	10.8	2.2	0.5	31
14	14.3	2.4	0.3	33
15	11.0	1.9	0.2	42
16	9.2	0.9	0.1	56
17	9.7	1.8	0.1	50
18	9.3	2.1	0.4	64
19	12.8	3.8	0.5	63
20	14.1	4.1	0.5	69
21	13.2	3.3	0.6	69
22	13.0	3.1	0.4	76
23	12.1	3.7	0.5	72
24	10.4	3.9	0.5	84
25	8.8	2.4	0.8	80
26	18.3	6.2	1.8	88
27	15.4	4.8	0.7	94
28	11.6	4.0	0.8	116
29	14.3	4.5	1.0	109
30	16.7	5.1	1.6	121

1) Average of 10 problems for each size
2) Average % from optimum
3) 50 ascent iteration
4) IBM PC/AT sec. (average)

Table 5-2. Computational Results of Tree Search on 250 CSP's

Number[1] of Flights	Initial[2] Upper Bound	Tree Search		
		Nodes in Tree	Computing Time-I[3]	Computing Time-II[4]
6	0	3	2	6
7	0	3	3	8
8	0	4	3	10
9	0	3	3	10
10	0	4	4	16
11	0	7	4	20
12	0	5	6	28
13	0	12	8	39
14	0	8	14	47
15	0	11	23	65
16	0	14	28	84
17	0	18	36	86
18	0	27	47	111
19	0	16	56	119
20	0	131	167	236
21	0	26	75	144
22	0	27	89	165
23	0	38	94	166
24	0	52	108	192
25	0	66	117	197
26	0	72	122	210
27	0	84	135	229
28	0	107	167	283
29	0	110	182	291
30	0	94	194	315

1) Average of 10 problems for each size

2) UB from heuristic of Chapter 4 (average % above optimum)

3) IBM PC/AT sec.

4) Total Computing time

Figure 5-3. Computational Results for Large CSP's

Number of Flights	Bound at Root Node[1]	Tree Search	
		Nodes	Total Time[2]
10	0.04	13	18
20	0.08	135	255
30	0.16	162	417
40	0.39	222	687
50	0.63	225	867
60	0.42	345	1137
70	0.82	459	1281
80	0.48	588	1362
90	0.97	645	2742
100	1.20	1401	3177

1) % above optimum (after 100 ascent iterations, k=300)

2) IBM PC/AT sec.

5-6. Conclusion

The combination of Lagrangean Relaxation and the generation of k-best solution described in this chapter provided a means of producing a tight lower bound to the CSP based on solutions to the AP. This bound when embedded in a branch and bound, produced an effective algorithm which can solve CSP's up to 100 flights on an IBM PC/AT. It is expected that perhaps CSP's up to 200 flights can be solved on a

workstation or other fast machine, since the rate of growth of the computing time of the algorithm is not high (even though in the worst case it could be exponential !) Thus, medium-size practical problems can be solved optimally by this procedure.

Chapter 6

CONCLUSIONS

In this thesis we consider three exact and one approximate procedure for the solution of Crew Scheduling Problems, such as those found in airline crew scheduling.

The first algorithm (called as the Clique Aggregation Technique - CAT) is based on a formulation dependent on the expansion of a graph representing all flights and their possible transitions. We present a method of expressing network-flow-type constraints which when used to surrogate them, produces a strong LP relaxation.

The second algorithm (called as the Iterative Time-constraint Addition algorithm - ITA) is based on the unexpanded flight-transition graph and uses the LP-relaxation solution to identify which constraints to add, as and when needed. These two algorithms are tested computationally and are shown to be able to solve small CSP's optimally.

The third exact algorithm, based on a unique combination of Lagrangean Relaxation and the generation of k-best optimal solutions to associated Assignment Problems, produces excellent lower bounds which, when embedded in a tree search can solve CSP's with more than 100 flights in less than one-hour computing time on an IBM PC/AT.

We describe a heuristic based on local optimization and using network-flow to search for possible improvements. This heuristic has produced optimal results on all the problems tried and whose optimal solution is known.

From the above it is apparent that practical-size problems can be solved on a fast PC (Intel 486CPU) or workstation exactly for up to 200 or so flights, and approximately for more than 200 flights. Unfortunately, practical problems are likely to include many additional constraints (not only the length of work-duty-time) including most-importantly rest-periods. Modifications of the exact algorithm of Chapter 5 are unlikely to lead to an effective solution procedure that includes these additional constraints. Thus, more research would be necessary to incorporate rest-period requirements successfully into a mathematical model that can exploit them. On the other hand, the heuristic procedure may not suffer as greatly from the addition of exogenous constraints and may be simpler to modify to incorporate rest periods. Further computational testing would be required to confirm (or otherwise) the effectiveness of the heuristic in solving CSP's with practical constraints.

REFERENCES

Antosik J.L. [1978], "Automatic Monthly Crew Assignment : A New Approach", ***AGIFORS 18 Symposium***, pp369-402.

Arabeyre J.P., Fearnley J., Steiger F.C. and Teather W. [1969], "The Airline Crew Scheduling Problem : A Survey", ***Transportation Science***, Vol. 3, pp140-163.

Assad A. [1980], "Models for Rail Transportation", ***Transportation Research***, Vol. 14A, pp205-220.

Baker E. [1979], ***Efficient Heuristic Solutions for the Airline Crew Scheduling Problem***, DBA Thesis, University of Maryland, College Park, Maryland, USA.

Baker E., Bodin L.D., Finnegan W.F. and Ponder R.J. [1979a], "Efficient Heuristic Solutions to an Airline Crew Scheduling Problem", ***AIEE Transactions***, Vol. 11, pp79-85.

Baker E., Bodin L. and Fisher M. [1980], "The Development and Implementation of a Heuristic Set Covering Based systems for Air Crew Scheduling", ***Working Paper No. 80-015, University of Maryland***, USA.

Baker E. and Fisher M. [1981], "Computational Results for Very Large Air Crew Scheduling Problems ", ***OMEGA***, Vol. 9, No. 6, pp613-618.

Baker K.R. [1974], ***Introduction to Sequencing and Scheduling***, John Wiley & Sons.

Balakrishnan N and Wong R.T. [1990], "A Network Model for the Rotating Workforce Scheduling Problem", ***Networks***, Vol. 20, pp25-42.

Balas E. [1965], "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", ***Operations Research***, Vol. 13, pp517-546.

Balas E. [1974], "Intersection Cuts from Disjunctive Constraints". ***Man. Res. Rep., No. 330, Carnegie-Mellon University.***

Balas E. [1975], "Disjunctive Programming: Cutting-Planes from Logical Conditions", in ***Nonlinear Programming***, Vol. 2, Mangasarian O.L. et al (eds.), Academic Press, New York.

Balas E. and Christofides N. [1981], "A Restricted Lagrangian Approach for the Travelling Salesman Problem", ***Mathematical Programming***, Vol. 21.

Balas E. and Ho A. [1980], "Set Covering Algorithms Using Cutting-Planes, Heuristics and Subgradient Optimization: A Computational Study", ***Mathematical Programming***, Study 12, Combinatorial Optimization, pp37-60.

Balas E. and Jeroslow R. [1975], "Strengthening Cuts for Mixed Integer Programs", ***MSSR No.359, GSIA, Carnegie-Mellon University***, USA.

Balas E. and Padeberg M. [1979], "Set Partitioning - a Survey", in ***Combinatorial Optimization***, Christofides N. et al (eds).

Ball M. [1980a], "A Comparison of Relaxation and Heuristics for Certain Crew and Vehicle Scheduling Problems", presented at the ***National ORSA/TIMS Meeting***, Washington DC.

Ball M., Bodin L. and Dial R. [1980], "Experimentation with a Computerized System for Scheduling Mass Transit Vehicles and Crews", presented at the ***International Workshop on Urban Passenger, Vehicle and Crew Scheduling***, the University of Leeds, England.

Ball M., Bodin L. and Dial R. [1983], "A Matching Based Heuristics for Scheduling Mass Transit Crews and Vehicles ", ***Transportation Science***, Vol. 17, pp4-31.

Ball M. and Roberts A. [1985], "A Graph Partitioning Approach to Airline Crew Scheduling", ***Transportation Science***, Vol. 19, No. 2, pp107-126.

Bazaraa M.S. and Jarvis J.J. [1977], ***Linear Programming and Network Flows***, John Wiley & Sons, New York.

Bazaraa M.S. and Goode J.J. [1979a], "A Survey of Various Tactics for Generating Lagrangean Multipliers in the Context of Lagrangian Duality", ***European Journal of Operational Research***, Vol. 3, pp322-338.

Bazaraa M. S. and Shetty C.M. [1979b], ***Nonlinear Programming: Theory and Algorithms***, John Wiley & Sons, New York.

Beale E.M.L. [1965], "Survey of Integer Programming", ***Operations Research Quarterly***, Vol. 16, pp219-228.

Beale E.M.L. [1979], "Branch and Bound Methods for Mathematical Programming Systems", ***Annals of Discrete Mathematics***, Vol. 5, Discrete Optimization II, pp201-219.

Beasley J. [1983], "Route First - Cluster Second Methods for Vehicle Routing", ***Omega***, Vol. 11, pp403-408.

Beasley J. [1987], "An algorithm for set covering problems", ***European Journal of Operational Research***, Vol. 31, pp85-93.

Beasley J. [1988], "A lagrangean heuristic for non-unicost set covering problem", ***Technical Report, The Management School, Imperial College***, London, England.

Beasley J. [1990], "Enhancing an algorithm for set covering problems", ***Technical Report, The Management School, Imperial College***, London, England.

Bellman R. [1957a], ***Dynamic Programming***, Princeton University Press, Princeton, N.J.

Bellman R. and Dreyfus S.E. [1957b], ***Applied Dynamic Programming***, Princeton University Press, Princeton, N.J.

Bellmore M., Bennington G. and Lubore S. [1971]. "A Multivehicle Tanker Scheduling

Problem", ***Transportation Science***, Vol. 5, No. 1, pp36-74.

Benichou M. et al [1971], "Experiments in Mixed Integer Linear Programming", ***Mathematical Programming***, Vol. 1, pp76-94.

Bennington G.E. and Rebibo K.K. [1975], "Overview of the RUCUS Vehicle and Scheduling Program (BLOCKS)", in ***Workshop on Automated Techniques for Scheduling of Vehicle Operations for Urban Public Transportation Systems*** held in Chicago, Bergmann D. and Bodin L. (eds)

Berge C. [1972], "Balanced Matrices", ***Mathematical Programming***, Vol. 2, pp19-31.

Bodin L. [1983], " Routing and Scheduling of Vehicles and Crews: the state of the art ", Samuel J. Raff ed., ***Computer and Operations Research***, Vol. 10, No. 2.

Bodin L. and Berman L. [1979], "Routing and Scheduling of School Buses by Computer", ***Transportation Science***, Vol. 13, pp113-129.

Bodin L. and Golden B. [1981], "Classification in Vehicle Routing and Scheduling", ***Networks***, Vol. 11, pp97-108.

Bodin L., Golden B., Assad A. and Ball M. [1982], "The State-of-the-Art in the Routing and Scheduling of Vehicles and Crews", ***Computers and Operations Research***, Vol. 10, pp63-212.

Bodin L. and Kursh S. [1978], "A Computer-Assisted Systems for the Routing and Scheduling of Street Sweeper", ***Operations Research***, Vol. 26, pp525-537.

Bodin L. Rosenfield D. and Kydes A. [1978a], "UCOST, A Micro Approach to a Transit Planning Problem", ***Journal of Urban Analysis***, Vol. 5, No. 1, pp47-69.

Bornemann D.R. [1970], " A crew planning scheduling system ", ***AGIFORS 10 Symposium.***

Brough B.M. [1962], "The Proper Study of an Air Crew", ***AGIFORS 2 Symposium.***

Camion P. [1965], "Characterization of Totally Unimodular Matrices", ***Proc. Am. Math. Soc.***, Vol. 16, pp1068-1073.

Carraresi P. and Gallo G. [1984], "Network models for vehicle and crew scheduling", ***European Journal of Operational Research***, Vol. 16, pp139-151.

Cheddad H. [1987], ***Algorithms for Crew Scheduling Problems***, PhD Thesis, Department of Management Science, Imperial College, London.

Cheshire I.M., Malleson A.M. and Naccach P.F. [1982], "A dual heuristic for the vehicle scheduling", ***Journal of Operations Research Society***, Vol. 33, pp51-61.

Christofides N. [1975], ***Graph Theory, an Algorithmic Approach***, Academic Press.

Christofides N. [1979], "The Travelling Salesman Problem", in ***Combinatorial Optimization*** edited by Christofides N. Mangoes A. Toth P. and Sandi C., John Wiley & Sons.

Christofides N. [1985], "Vehicle Routing", in ***The Travelling Salesman Problem*** edited by Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. and Shimoys D.B., Wiley Chichester.

Christofides N. [1991], "The Use of k-best Solutions in Combinatorial Optimization", IC OR Section Report, Imperial College

Christofides N. and Hadjiconstantinou E. [1992], "An Exact Algorithm for Vehicle Routing Problem Based on k-shorteset Paths", IC OR Section Report, Imperial College

Christofides N. and Mingozzi A. [1992], "An Exact Algorithm for Vehicle Routing Problem", IC OR Section Report, Imperial College

Christofides N. Mingozzi A. and Toth P. [1981], "State Space Relaxation Procedures for the Computation of Bounds to Routing Problems", ***Networks***, Vol. 11. pp145-164.

Christofides N. and Paixão J. [1982], "***State Space Relaxation Algorithms for the Set Covering Problem***", Imperial College Report No. IC-OR-81/12.

Christofides N. and Valls V. [1986], "Finding All Optimal Solutions to the Network Flow Problem" ***Mathematical Programming Study***, Vol. 26, pp209-212

Christofides N. and Weston [1992], "The Generation of k-shortest Paths in Graph", IC OR Section Report, Imperial College

Chvatal V. [1979], "A Greedy Heuristic for the Set Covering Problem", *Mathematics of* ***Operations Research***, Vol. 4, pp233-235.

Connolly D.T. [1989], ***Solution Techniques for the Quadratic Assignment Problem and the Development of a General Purpose Simulated Annealing Algorithm***, Ph.D. Thesis, London School of Economics and Political Science, London.

Cook S.A. [1971], "The Complexity of Theorem Proving Procedures", ***Proc. 3rd Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery***, New York, pp151-158.

Cook S.A. [1973], "A Hierarchy for Nondeterministic Time Complexity", ***J. Compt. System Ski.***, Vol. 7, pp343-353.

Cook W. [1990], "Cutting-plane Proofs in Polynomial Space", ***Mathematical Programming***, Vol. 47, pp11-18.

Crainic T.G. and Rousseau J.M. [1987], "The column generation principle and the airline crew scheduling problem ", ***INFOR***, Vol. 25, No. 2.

Crowder H., Johnson E.L. and Padeberg M. [1983], "Solving Large-scale Zero-one Linear Programming Problems", ***Operations Research***, Vol. 31, No. 5, pp803-804.

Dakin R.J. [1965], "A Tree Search Algorithm for Mixed-Integer Programming Problems", ***Computer Journal***, Vol. 8, pp250-255.

Dantzig G. [1963], ***Linear Programming and Extensions***, Princeton University Press. Princeton, N.J.

Dantzig G. and Fulkerson D. [1954], "Minimizing the Number of Tankers to Meet a Fixed Schedule", ***Naval Research Logistics Quarterly***, Vol. 1, pp217-222.

Dantzig G. and Wolfe P. [1961], "The Decomposition Algorithm for Linear Programming", ***Econometrica***, Vol. 9, No. 4.

Deo N. [1974], ***Graph Theory with Application to Engineering and Computer Science***, Prentice Hall.

Derigs U. [1983], ***Exchange Properties and K-best Strategies in Combinatorial Optimization***, Institute fur Okonometrie und Operations Research, Rheinische Friedrich-Wilhelms-Universitat, Bonn.

Desrochers M. and Soumis F. [1988], "A re-optimization algorithm for the shortest path problem with time windows ", ***European Journal of Operations Research***, Vol. 35, pp242-254.

Desrochers M. and Soumis F. [1989], "A Column Generation Approach to the Urban Transit Crew Scheduling Problem", ***Transportation Science***, Vol. 23, No. 1, pp1-13.

Desrosiers J., Desrochers M. and Soumis F. [1984], "Routing with Time-window by Column Generation", ***Networks***, Vol. 14, pp545-565.

Dreyfus S.E. and Law A.M. [1977], ***The Art and Theory of Dynamic Programming***, Academic Press, New York.

Etcheberry J. [1977], "The Set-Covering Problem: A New Implicit Enumeration Algorithm", ***Operations Research***, Vol. 25, No. 5, pp760-772.

Etschmaier M. M. [1984], "Aircraft scheduling: the state of the art". ***AGIFORS 24 Symposium***, Strassbourg, France.

Etschmaier M. M. [1985], "Airline Scheduling : An Overview", ***Transportation Science***, Vol. 19, No. 2, pp127-138.

Fearnley J. [1966], "Crew scheduling development in Air Canada", ***AGIFORS 6 Proceedings***.

Finnegan W. [1977], "A Network Model for Bidline Generation", in ***FEC Technical Report***, Federal Express Corporation, Memphis, TN.

Fisher M.L. [1973], "Optimal solutions of scheduling problems using Lagrange multipliers: part 1", ***Operations Research***, Vol. 27, pp1114-1127.

Fisher M.L. [1981], "The Lagrangian Relaxation Method for Solving Integer Programming Problems", ***Management Science***, Vol. 27, No. 1, pp1-18.

Fisher M.L. [1985], "An Application Oriented Guide to Lagrangian Relaxation", ***Interfaces***, Vol. 15, No. 2, pp10-21.

Fisher M.L. and Rabin M.O. [1974], "Super-exponential Complexity of Presburger Arithmetic", in ***Complexity of Computation*** edited by Karp R.M., American Mathematical Society, Providence, RI, 27-41.

Fisher M.L. and Rasenwein M. [1985], "An Interactive Optimization System for Bulk Cargo Ship Scheduling", ***Working Paper*** 85-08-07, University of Pennsylvania.

Florian M., Guerin G. and Bushel G. [1976], "The Engine Scheduling Problem in a Railway Network", ***INFOR***, Vol. 14, pp121-138.

Ford L. and Fulkerson D. [1962], ***Flows in Network***, Princeton University Press, Princeton, New Jersey.

Forrest J.J.H., Hirst J.P.H. and Tomlin J.A. [1974], "Practical Solution of Large Mixed Integer Programming Problems with UMPIRE", ***Management Science***, Vol. 20, pp736-773.

French S., Hartley R., Thomas L. and White D.J. [1986], ***Operational Research Technique***, Arnold

Gallo G. and Pallottino S. [1986], "Shortest Path Methods, a Unifying Approach", ***Mathematical Programming Study***, Vol. 26, pp38-64.

Garey M.R. and Johnson D.S. [1979], ***Computers and Intractability: a Guide to the Theory of NP-Completeness***, Freeman & Co, San Francisco.

Garfinkel R.S. [1979], "Branch and Bound Methods for Integer Programming", in ***Combinatorial Optimization*** edited by Christofides N. Mingozzi A. Toth P. and Sandi C., John Wiley & Sons, pp1-20.

Garfinkel R.S. and Nemhauser G.L. [1972], ***Integer Programming***, John Wiley & Sons.

Gavish B. [1978], "On Obtaining the Best Multipliers for a Lagrangean Relaxation for Integer Programming", ***Computers and Operations Research***, Vol. 3, pp322-338.

Gavish B. Srikanth K.N. [1983], "Efficient Branch and Bound Code for Solving Large Scale Travelling Salesman Problems to Optimality", ***Working Paper*** **Qm 8329, Graduate School of Management, University of Rochester.**

Geoffrion A.M. [1967], "Integer Programming by Implicit Enumeration and Balas' Method" ***SIAM Review***, Vol. 9, pp178-190.

Geoffrion A.M. [1974], "Lagrangean Relaxation for Integer Programming" ***Mathematical Programming Study***, Vol. 2, pp82-114.

Geoffrion A.M. and Graves G. [1974], "Multicommodity Distribution System Design by Benders Decomposition", ***Management Science***, Vol 20., pp822-844.

Geoffrion A.M. and Mac Bride R. [1978], "Lagrangian Relaxation Applied to Capacitated Facility Location Problems", ***AIIE Transactions***, Vol. 10, pp40-47.

Geoffrion A.M. and Marsten R.E. [1972], "Integer Programming Algorithms: A

Framework and State-of-the-Art Survey", ***Management Science***, Vol. 18. No. 9. pp465-491.

Gerbracht R.J. [1978], "A New Algorithm for Very Large Crew Pairing Problems", ***AGIFORS 18 Proceedings***, Vancouver, British Columbia, Canada.

Gerschkoff I. [1989], "Optimize flight crew schedules", ***Interface***, Vol. 19. No. 9. pp29-43.

Gertsbach I. and Gurevich Y. [1977], "Constructing an optimal fleet for a transportation schedule", ***Transportation Science***, Vol. 11, No. 1.

Giannacopoulos A. [1982], ***Optimization Algorithms for Mass Transportation Networks***, Ph.D. Thesis, Imperial College, London.

Giannessi F. and Nicoletti B. [1979], "The Crew Scheduling Problem: A Travelling Salesman Approach", in ***Combinatorial Optimization*** edited by Christofides N. Mingozzi A. Toth P. and Sandi C., John Wiley & Sons, pp389-419.

Gillet B. and Miller L. [1974], "A Heuristic Algorithms for the Vehicle Dispatch Problem", ***Operations Research***, Vol. 22, pp340-349.

Glanert W. [1984], "A Timetable Approach to the Assignment of Pilots to Rotations", ***AGIFORS 24 Proc.***, pp369-392.

Glover F. [1968], "Surrogate Constraints", ***Operations Research***, Vol. 16, No. 4, pp741-749.

Glover F. [1973], "Convexity Cuts and Cut Search", ***Operations Research***, Vol. 21. pp123-134.

Goldon B.L and Assad A.A. (ed.) [1987], ***Vehicle Routing : Method and Studies***, North-Holland.

Gomory R.E. [1958], ***"An Algorithm for the Mixed Integer Problem"***. RM-2597. Rand

Corporation.

Gomory R.E. [1963], "An Algorithm for Integer Solutions to Linear Programs", in *Recent Advances in* ***Mathematical Programming*** edited by Graves and Wolfe, pp269-302.

Gouvras S. [1987], ***Optimization Methods for Public Transport Networks***, PhD thesis, Imperial College.

Griesshaber R. [1980], "Model for crew scheduling for a short haul fleet", ***AGIFORS 20 Symposium.***

Graham R., Lawler E., Lenstra J. and Rinnooy Kan A. [1970], "Optimization and Approximization in Deterministic Sequential and Scheduling: A Survey", ***Annals of Discrete Mathematics***, Vol. 5, pp287-326.

Guignard M. and Kim S [1987], "Lagrangean decomposition: a model yielding stronger lagrangean bounds", ***Mathematical Programming***, Vol. 39, North Holland.

Guignard M. and Kim S. [1987], "Lagrangean decomposition for Integer Programming: theory and applications", ***Technical Report, Department of Statistics, Wharton School, University of Pennsylvania.***

Hamacher H.W. [1983], "K Best Solutions to Combinatorial Optimization Problems", ***University of Florida***, Research Report No.83-5.

Harary F. [1971], ***Graph Theory***, Addison-Wesley.

Hartmanis J. and Stearns R.E. [1965], "On the Computational Complexity of Algorithm", ***Trans. Amer. Math. Soc.***, Vol. 117, pp285-306.

Held M. and Karp R.M. [1970], "The Travelling Salesman Problem and Minimum Spanning Trees", ***Operations Research***, Vol. 18, pp1138-1162.

Held M. and Karp R.M. [1971], "The Travelling Salesman Problem and Minimum

Spanning Trees: Part II", ***Mathematical Programming***, Vol. 1, pp6-25.

Held M., Wolfe P. and Crowder H.P. [1974], "Validation of Subgradient Optimization", ***Mathematical Programming***, Vol. 6, pp62-88.

Heslop I. G. [1965], "A Computer Method for the Allocation of Duties to Air Crews", ***AGIFORS 5 Symposium.***

Heurgon E. [1975], "Preparing Duty Rosters for Bus Routes by Computer", Reprints. ***Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Transportation Services***, Chicago, Illinois, USA.

Hey A. [1980], ***Algorithms for Set Covering Problem***, Ph. D. Thesis, Department of Management Science, Imperial College, London.

Hillier F.S. and Lieberman G.J. [1986], ***Introduction to Operations Research***, Holder-Day Inca, Oakland, California, U.S.A.

Hinson J. and Mulherkar S. [1975], "Improvements to the Clarke and Wright Algorithm as Applied to an Airline Scheduling Problem", ***Technical Report, Federal Express Corporation.***

Ho A.C. [1979], "Worst Case Analysis of a Class of Set Covering Heuristics", ***Management Science Research Report***, No. 473, Carnegie-Mellon University.

Hochbaum D. [1980], "Approximation Algorithm for the Weighted Set Covering and Node Cover Problems", ***GSIA***, Carnegie-Mellon University.

Hoffman A.J. and Kruskal J.B. [1956], "Integral Boundary Points of Convex Polyhedra", in ***Linear Inequalities and Related Systems*** **(Annals of Mathematics Studies No 38)** edited by Kuhn H.W. and Tucker A.W., Princeton, N.J.

Hong J. D. and Chandra M. J. [1990], "A heuristic algorithm to minimize the number of crews allocated in an airlift operation", ***Computers and Operational Research***, Vol. 17, No. 4, pp389-396.

Horowitz E. and Sahni S. [1978], ***Fundamentals of Computer Algorithms***, Pitman.

Jeroslow R.E. [1977], "Cutting-Plane Theory: Disjunctive Method", ***Annals of Discrete Mathematics***, Vol. 1, pp293-339.

Jeroslow R.E. [1978], "The Theory of Cutting-Planes" in ***Combinatorial Optimization*** edited by Christofides N. Mingozzi A. Toth P. and Sandi C., John Wiley & Sons.

Jeroslow R.E. [1979], "An Introduction to the Theory of Cutting-Planes". ***Annals of Discrete Mathematics***, Vol. 5, Discrete Optimization II, pp71-95.

Johnson D.S. [1974], "Approximation Algorithm for Combinatorial Problems", ***Journal Computer Sci.***, Vol. 9, pp256-278.

Jones R.D. [1979], "Development of an Automated Airline Crew Bid Generation System", ***Interface***, Vol. 19, pp44-51.

Karp R.M. [1972], "Reducibility among Combinatorial Problems", in ***Complexity of Computer Computations*** edited by Millet R.E. and Thatcher J.W., Plenum Press, New York, pp85-103.

Karp R.M. [1975], "On the Complexity of Combinatorial Problems", ***Networks***, Vol. 5, pp45-68.

Klee V. [1980], "Combinatorial Optimization : What is the State of Art ?", ***Mathematics of Operations Research***, Vol. 5, pp1-26.

Klee V. and Minty G.J. [1972], "How Good is the Simplex Algorithm ?", in ***Inequalities III*** edited by Shisha O., Academic Press, pp159-175.

Klingman D. [1977], "Finding Equivalent Network Formulations for Constrained Network Formulations", ***Management Science***, Vol. 23, No. 7.

Klingman D. and Russel R. [1975], "Solving Constrained Transportation Problems".

Operations Research, Vol. 23, No 1.

Klorak P. and Nelson J. [1978], "A Family of Truck Load Clustering Heuristics for Vehicle Routing Problems", ***Technical Report 78-2, Department of Computer Science, Vanderbilt University***, Nashville, TN.

Klorak P. and Williams M. [1978], "Computerized School Bus Routing", ***Technical Report 78-1, Department of Computer Science, Vanderbilt University***, Nashville, TN.

Kolen A., Rinnooy Kan A. and Trienekens H. [1987], "Vehicle Routing with Time Windows", ***Operations Research***, Vol. 35, No. 2.

Kolner T.K. [1966], "Some highlights of a scheduling matrix generation system", ***AGIFORS 6 Symposium***, pp19-22.

Korman S.M. [1975], ***Graph Colouring and Related Problem in O.R.***, Ph.D. Thesis, Department of Management Science, Imperial College, London.

Koskosidis Y.S.[1988], ***An Optimization Based Models and Algorithms for Routing and Scheduling with Time Window Constraints***, Ph.D. Thesis, Princeton University.

Koskosidis Y.S., Powell W.B. and Solomon M.M [1989], "An Optimization Based Heuristic for Vehicle Routing and Scheduling with Time Window Constraints", ***TMS 88-09-1, The Institute for Transportation Systems, The City College of CUNY***, New York.

Krishnamoorthy M. [1990], ***Contributions to the Solution of the Symmetric Travelling Salesman Problem***, PhD Thesis, the Management School, Imperial College, London.

Land A. and Doig A.G. [1960], "An Automatic Method for Solving Discrete Programming Problems", ***Econometrica***, Vol. 28, pp497-520.

Land A. and Powell S. [1960], "Computer Codes for Problems of Integer Programming", presented at the ***Conference on Discrete Optimization***, Vancouver, B.C.

Laporte G. and Nobert Y. [1987], "Exact algorithm for the vehicle routing problem", ***Annals of Discrete Mathematics***, Vol. 31., pp147-184.

Lasdon L.S. [1970], ***Optimization Theory for Large Systems***, The Macmillan Company.

Lavoie S., Minoux M. and Odier E. [1988], "A new approach for pairing problems by column generation with an application to air transportation". ***European Journal of Operations Research***, Vol. 35, pp45-58.

Lawler E.L. [1972], "A Procedure for Computing the K-best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem", ***Management Science,*** Vol. 7, pp401-405.

Lawler E.L. [1976], ***Combinatorial Optimization : Networks and Matroid***, Holt Rinehart and Winston, New York.

Lawler E.L. and Wood D.E. [1960], "Branch-and-Bound Method: A Survey", ***Operations Research***, Vol. 14, pp699-719.

Lee K. H. [1990], ***Algorithms for routing problems in distribution***, Ph.D. Thesis, Imperial College, London.

Lemke C.E., Salkin H.M. and Spielberg K. [1971], "Set Covering by Single-branch Enumeration with Linear Programming Subproblems", ***Operations Research***, Vol. 19, p.998-1022.

Lenstra J.K. and Rinnooy Kan A.H.G. [1979], "Computational Complexity of Discrete Optimization Problems", ***Annals of Discrete Mathematics***, Vol. 4, Discrete Optimization I, pp121-140.

Lenstra J.K. and Rinnooy Kan A.H.G. [1981], "Complexity of Vehicle Routing and Scheduling Problems", in Proceedings of the International Workshop on Current and Future Directions in the ***Routing and Scheduling of Vehicles and Crews*** edited by Golden B.L. and Bodin L.D., *Networks*, Vol. 11, No. 2, pp221-228.

Lessard R., Rousseau J.M. and Dupis D. [1981], "HASTUS I: A mathematical programming approach to the bus driver scheduling problem", ***Centre de recherche sur les transports, université de Montréal***, Canada.

Lewis H. and Papadimitriou C. [1981], ***Elements of the Theory of Computation***, Prentice -Hall, Englewood Cliffs, New Jersey.

Lin S. [1965], "Computer Solution of the TSP", ***Bell System Technical J.***, Vol. 44, pp22-45.

Lin S. and Kernighan B.W. [1973], "An Effective Heuristic Algorithm for the Travelling Salesman Problem", ***Operations Research***, Vol 21, pp498 -516.

Loo J., O'Donald B.T. and Whiteman I.R. [1970], "CREATION : A Heuristic Program the Crew Allocation", ***AGIFORS 10 Proc.***, pp1-39.

Lovász L. [1975], "On the Ratio of Optimal Integral and Fractional Covers", ***Discrete Mathematics,*** Vol. 13, pp774-390.

Lubow B. [1981], ***Aircraft Scheduling : An Interactive Graphics Approach***, M.S. Thesis, Department of Aeronautics and Astronautics, M.I.T.

Lucena F.A.P. [1986], ***Exact solution approaches for the vehicle routing problems***, Ph.D. Thesis, Department of Management Science, Imperial College, London.

Maffioli F. [1979], "The Complexity of Combinatorial Algorithms and the Challenge of Heuristics", in ***Combinatorial Optimization*** edited by Christofides N. Mingozzi A. Toth P. and Sandi C., John Wiley & Sons, pp107-129.

Marsten R.E. [1974], "An Algorithm for Large Set-Partitioning Problems", ***Management Science***, Vol. 20, pp774-787.

Marsten R.E. [1981a], "The Design of the XMP Linear Programming Library", ***ACM Transactions on Mathematical Software***, Vol. 7, No. 4, pp481-497.

Marsten R.E., Muller M. and Killion C. [1978], "Crew Planning at Flying Tiger: A Successful Application of Integer Programming", ***Technical Report No 553, Management Information Systems Dept., University of Arizona***, Tucson, AZ.

Marsten R.E. and Shepardson F. [1981], "Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications", ***Networks***, Vol 11, No. 2, pp165-177.

Martello S. and Toth S. [1983], "A heuristic approach to the bus driver scheduling problem", ***European Journal of Operational Research***, Vol. 24, pp106-117.

Martin G.E. [1981], "Aircraft Scheduling Considered as an N-Task, M-Parallel Machine Problem with Start-Times and Deadlines", ***INFOR***, Vol. 19, No. 2.

Martin G.T. [1963], "An Accelerated Euclidean Algorithm for Integer Linear Programming", in ***Recent Advances in Mathematical Programming*** edited by Graves and Wolfe.

Martin-Lof A. [1970], "A Branch and Bound Algorithm for Determining the Minimal Fleet Size of a Transportation System", ***Transportation Science***, Vol. 4, pp159-163.

Mayer M. [1980], "Monthly Computerized Crew Assignment, ***AGIFORS 20 Proc.***, pp93-124.

Meyer A.R. and Stockmeyer L.J. [1972], "The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Time", ***Proc. 13th Ann. Symp. on Switching and Automata Theory***, IEEE Computer Society, Long Beach, CA, pp125-129.

Middleton W.J. [1984], "Bidding for Aircraft - A Pilot's right to choose", ***AGIFORS 24 Proc.***, pp355-368.

Minoux M. [1984], "Column generation techniques in combinatorial optimization: A new approach to crew pairing problems", ***AGIFORS 24 Proc.***, pp17-29.

Mitra G. and Darby-Dowman K. [1985], "CRU-SCHED A Computer Based Bus

Scheduling System Using Integer Programming", in ***Proceeding of the Third International Workshop on Transit Vehicle and Crew Scheduling in Montreal***, edited by Rousseau J.M., North Holland.

Mitra G. and Welsh A.P.G. [1981], "A Computer Based Crew Scheduling System Using a Mathematical Programming Approach", in ***Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*** edited by Wren A., North Holland Publishing Company, pp281-296.

Mole R., Johnson D. and Wells K. [1983], "Combinatorial Analysis of Route First-Cluster Second Vehicle Routing", ***Omega***, Vol. 11, pp507-512.

Moore R., Evans J. and Ngo H. [1978], "Computerized Tailored Blocking", ***AGIFORS 18 Proc.***, pp343-368.

Murty K.G. [1968], "An Algorithm for Ranking All the Assignments in Order of Increasing Cost", ***Operations Research***, Vol. 16, pp682-690.

Nemhauser G.L. [1966], ***Introduction to Dynamic Programming***, John Wiley & Sons, New York.

Nicoletti B. [1973], "Automatic Crew Rostering", ***AGIFORS 13 Proc.***, pp158-181.

Nijenhuis A. and Wilf H.S. [1978], ***Combinatorial Algorithms***, Academic Press.

Orlin J. [1982], "Minimizing the Number of Vehicles to Meet a Fixed Periodic Schedule: An Application of Periodic Posts", ***Operations Research***, Vol 30., pp760-776.

Orloff S.C. [1976], "Route Constrained Fleet Scheduling", ***Transportation Science***, Vol. 10, No. 2, pp149-168.

Orloff S.C. [1974], "A Fundamental Problem in Vehicle Routing", ***Networks***, Vol. 4. pp35-64.

Owen G. [1973], "Cutting-Planes for Programs with Disjunctive Constraints", ***Journal***

of Optimization Theory and its Applications, Vol. 11, pp49-55.

Padeberg M.W. [1975], "Characterizations of Totally Unimodular, Balanced and Perfect Matrices", in ***Combinatorial Programming, Methods and Applications*** edited by Roy B., Reidel Publishing Company, Dowrecht-Holland.

Paixão J. [1983], ***Algorithms for Large Scale Set Covering Problems***, Ph. D. Thesis, Department of Management Science, Imperial College, London.

Paixão J., Branco I.M., Captivo E. and Pato M.V. [1986], "Bus and Crew Scheduling on a Micro Computer", in ***OR Models on Microcomputer*** edited by Coelho J.D. et al, North Holland.

Paixão J. Pato M. [1989], "A structural lagrangean relaxation for two-duty period bus driver scheduling problem", ***European Journal of Operations Research***, Vol.39., pp213-222.

Parker M.E. and Smith B.M. [1981], "***Two approaches to computer crew scheduling***", Operational Research Unit, Department of Computer Studies, University of Leeds, Leeds, U.K.

Papadimitriou C. and Steiglitz K. [1982], ***Combinatorial Optimization***, Prentice-Hall, Englewood Cliffs, New Jersey.

Psaraftis H.N., Solomon M.M., Magnanti T.L. and Kim T. [1990], "Routing and Scheduling in a Shoreline with Release Times", ***Management Science***, Vol. 36, No. 2, pp212-223.

Rinnooy Kan A.H.G. [1976], ***Machine Scheduling Problems***, Martinus Nijhoff, Hague.

Rousseau J.M. [1984], "Crew scheduling methods in the operation and planning of transportation systems", in ***Transportation Planning Models***, Florian M. (ed.).

Rardin R.L. and Lin B.W. [1978], "What Makes Integer Programming Problems Hard to Solve: An Empirical Study", ***Operations Research***, Vol. 15, No. 2 & 3, pp65-77.

Roy B., Benayoun R. and Tergny J. [1970], "From S.E.P Procedure to the Mixed Ophelie Program", in ***Integer and Nonlinear Programming*** edited by Abadie J., American Elsevier, pp419-436.

Rubin D. [1973], "A Hybrid Cutting-Planes Enumeration Algorithm for Integer Programming", ***43rd National Meeting of ORSA***, Milwaukee.

Rubin J. [1973], "A Technique for the Solution of Massive Set Covering Problems with Application to Airline Crew Scheduling", ***Transportation Science***, Vol. 7, pp34-48.

Rubin J. [1973], "A Survey of Technique for Airline Crew Scheduling", ***Working Paper, Cambridge Scientific Center***, Cambridge, Mass., USA.

Ryan D.M. [1989], "The Solution of Massive Generalized Set Partitioning Problems in Air Crew Rostering", ***Center for Applied Mathematics, Cornell University.***

Ryan D.M. and Falkner J.C. [1988], "On the integer properties of scheduling set partitioning models", ***European Journal of Operational Research***, Vol. 35, pp442-456.

Sandi C. [1976], "A Direct Method for Linear Inequalities", presented at ***EURO II***, Stockholm, Sweden.

Sandi C. [1979], "Subgradient Optimization", in ***Combinatorial Optimization*** edited by Christofides N. Mingozzi A. Toth P. and Sandi C., John Wiley & Sons.

Salkin H. [1975], ***Integer Programming***, Addison-Wesley.

Shapiro J. [1968], "Group Theoretic Algorithms for the Integer Programming Problem: Extension to a General Algorithm", ***Operations Research***, Vol. 16, pp928-947.

Shapiro J. [1979], "A Survey of Lagrangean Techniques for Discrete Optimization". ***Annals of Discrete Mathematics***, Vol. 5, Discrete Optimization II, pp113-138.

Shepardson F. and Marsten R.E. [1980], "A Lagrangian Relaxation Algorithm for the

Two Duty Period Scheduling Problem", ***Management Science***, Vol. 26, pp274-281.

Sklar M.G., Armstrong R.D. and Samn S. [1990], "Heuristics for the Scheduling Aircraft and Crew during Airlift Operations", ***Transportation Science***, Vol. 24. No. 1. pp63-76.

Smith B. and Wren A. [1981], "VAMPIRES and TASC: Two Successfully Applied Bus Scheduling Programs", in ***Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*** edited by Wren A., North Holland Publishing Company, pp97-124.

Solomon M. [1987], "Algorithms for the vehicle routing and scheduling problems with time window constraints", ***Operations Research***, Vol. 35, No. 2, pp254-265.

Solomon M. and Desrosiers J. [1988], "Time window constrained routing and scheduling problems", ***Transportation Science***, Vol. 22, No. 1, pp1-13.

Spitzer M. [1961], "Solution to the Crew Scheduling Problem", ***AGIFORS 1 Symposium.***

Steiger F. [1965], "Optimization of Swissair's crew scheduling by an integer linear programming model", ***AGIFORS 5 Symposium.***

Steiger F. [1967], "Activity report of AGIFORS study-group crew scheduling", ***AGIFORS 7 Symposium.***

Swersey A.J. and Ballard W. [1984], "Scheduling School Buses", ***Management Science***, Vol. 30, No. 7, pp844-853.

Taha H.A. [1975], ***Integer Programming: Theory, Applications and Computations***, Academic Press.

Tarjan R. [1978], "Complexity of Combinatorial Algorithms", ***SIAM***, Rev. 20, pp457-491.

Thiriez H.M. [1969], "Airline Crew Scheduling: A Group Theoretic Approach", ***Flight Transportation Laboratory, Report R69-1, Department of Aeronautics and Astronautics, MIT, Cambridge***, Mass.

Tingley G.A. [1979], "Still another solution method for the monthly aircrew assignment problem", ***AGIFORS 19 Symposium.***

Tobin N.R. and Butfield T.E [1970], "OR Models and then Implementation in the Airline's Planning Cycle", ***AGIFORS 10 Symposium.***

Toregas C. and Revelle C. [1972], "Location under Time or Distance Constraints", ***Papers of the Regional Science Association***, Vol. 28, pp138-143.

Ward R., Durant P. and Hallman A. [1981], "A Problem Decomposition Approach to Scheduling the Drivers and Crews of Mass Transit Systems", in ***Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*** edited by Wren A., North Holland Publishing Company, pp297-312.

Wolters J.A.M. [1979], "Minimizing the Number of Aircraft for a Transportation Network", ***European Journal of Operational Research***, Vol. 3, pp394-402.

Wren A. [1981], "General Review of the Use of Computers in Scheduling Buses and their Crews", in ***Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*** edited by Wren A., North Holland Publishing Company, pp3-16.

Yen J.T. [1971], "Finding the k-shortest loopless paths in a network", ***Management Science***, Vol. 17, pp712 -716.

Young R.D. [1971], "Hypercylindrically-Deduced Cuts in Zero-One Integer Programs", ***Operations Research***, Vol. 19, pp1393-1405.

Zadeh N. [1973], "A Bad Network Problem for the Simplex Method and Other Minimum Cost Flow Algorithms", ***Mathematical Programming***, Vol. 5, pp255-266.