AIRLINE CREW SCHEDULING:
A GROUP THEORETIC APPROACH

by

HERVE MARCEL PIERRE THIRIEZ
Ingenieur Mathematiques Appliquees
Grenoble IMAG
(1966)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
(October, 1969)
i.e. FEBRUARY 1970

Signature of Author_____

Certified by:_____
Department of Aeronautics
Thesis Supervisor

_____
Department of Civil Engineering

Accepted by_____
Chairman, Departmental Graduate
Committee

AIRLINE CREW SCHEDULING:
A GROUP THEORETIC APPROACH

by

Herve Thiriez

Submitted to the Department of Civil Engineering on October 10, 1969 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

The problem of airline crew scheduling is studied, the different problems composing it are formulated and solution techniques are offered.

Special emphasis is given to the set covering problem appearing in the rotation selection phase. An approach is presented, based on the group theoretic method, which allows the fast solution of problems with large sizes.

Thesis Supervisor: Robert W. Simpson

Title: Associate Professor of Aeronautics and Astronautics

ACKNOWLEDGEMENTS

My deepest appreciation goes to Professor Robert W. Simpson for his constant supervision and helpful suggestions. To Professor Alan M. Hershdorfer who encouraged me to pursue the Doctoral Program and assisted me through most of it, I want to express my gratitude; also, to Professor Jeremy F. Shapiro who introduced me to group theory and made this thesis possible.

Finally, my thanks go to Mrs. Betsy Gaudreau who typed, erased, and re-typed this thesis with almost constant good humor.

TABLE OF CONTENTS

INTRODUCTION

1.1 The Airline Crew Scheduling Problem

The Airline Crew Scheduling problem is the problem airlines have of building monthly assignments for crews which minimize the total cost of the operation. The problem may be partitioned into different parts; the most important part, ie. the part where the possible savings an optimal solution would bring are the largest, is the selection of an optimal set of rotations. A rotation is a duty assignment a crew receives which may last a number of days and at the end of which the crew returns to its base. Each rotation therefore covers several flights of the schedule. The optimal set of rotations covers all scheduled flights at a minimal cost.

1.2 Mathematical Formulation of the Rotation Selection Problem

The Mathematical formulation is well known as the "set

covering" problem.

$$\text{Min } z = cx$$

$$\text{where } Ax = 1 \quad \text{or } Ax \geqslant 1$$

$$x \in N \text{ i.e. } x \text{ is a non-negative integer vector}$$

A is a 0-1 matrix where rows correspond to flights and columns to rotations.

x is an integer row vector constrained to be non-negative

1 is a column of ones.

Constraint $Ax \geqslant 1$ is used for some airlines who will accept scheduling two or more crews on a flight if the resulting operation costs less.

The mathematical difficulty resides in the integrality condition and in the size of the normal problem since 1000 rows and 10000 columns is a normal size; this is within current computational capabilities for a non-integral problem; but not for most integer solution methods.

1.3 Survey of Previous Work

Most airlines are still solving the problem manually or using heuristic non-optimal solution methods. The size of the problem has made it impossible for previous

integer optimization codes to find an optimal integer solution within an acceptable computer time.

Basically, five approaches have been taken to solve the problem:

1) The first approach groups all the non-optimal heuristic methods, thousands of which may easily be developed; some of the best ones start from the continuous LP (linear programming) optimum (references 2,3,5,13,21).

2) The second approach is the cutting plane approach. Gomory (10, 11) started this a long time ago and many developments have taken place since; better cuts have been found and primal algorithms are now used so that feasible solutions are available before optimality is reached.

3) Another approach is based on branch and bound techniques; Land and Doig (15) did the ground work for these methods and, amazingly enough, are just being "discovered" now by a number of airline O.R. analysts. Branch and bound solves the continuous problem and successively fixes the value of non-integer variables at their adjacent integer levels at each node of a tree (see section 2.2.4).

4) Balas (4) and, later, Geoffrion (8, 9) developed the fourth approach, implicit enumeration. This method also forces variables to integer levels at each node of a tree but the

progression through the tree is organized; this provides very easy table management (to keep track of the progression in the tree), but there is a lack of flexibility in the way the optimization takes place (see section 2.2.2).

5) The fifth approach corresponds to an extension by Shapiro (19,20), Glover, White, and others of the ideas of group theoretic methods expressed by Gomory in papers (10,11).

This thesis presents an extension of the method coupled with several modifications tailored to the Rotation Selection problem. The airlines are attempting to use computers and optimal models to solve this problem. A good review of their current capabilities is given in the recent Transportation Science paper[2].

1.4 Outline

The dissertation presents the crew scheduling process in the first chapter and discusses the different parts of the process, the problems associated with them, and the solution methods available. The second chapter describes in its first section the different mathematical formulations of the rotation selection problem; the second section shows the different solution methods one may use; those methods are then briefly evaluated in a third section. The group theoretic method is explored in detail in the third chapter, along with the modifications and assumptions.

applicable to the rotation selection problem; a sample problem of small size is also completely solved to show the workings of the method. The fourth chapter offers an extensive description of the computational experience obtained in this research on problems provided by the airlines. A conclusion summarizes the work and projects possible developments.

CHAPTER I

THE CREW SCHEDULING PROCESS

This chapter will present the crew scheduling process. Several introductory definitions will be offered, followed by a presentation of the total analysis and a description of the approaches several airlines have developed to solve the different problems related to crew scheduling.

1.1. Definitions

1.1.1 Crew: The pilot and co-pilot(s) needed for a commercial flight.

1.1.2 Flight Leg: A flight leg is a flight from a city to another on a given day at a given time; example: Boston-Chicago, 8:30 a.m. on Mondays.

1.1.3 Composite Flight: A set of two or more flight legs arbitrarily put together - some carriers consider return flights to be the smallest unit for crew scheduling studies;

in that case, composite flights are return flights or composites of return flights. This is more valid in Europe where most airlines are national and have an operation completely centered around one city (usually the capital). Composite flights are generated when the carrier would incur too high a penalty (e.g. long layover at an airport) by allowing the crew to take a different flight leg after the first one.

1.1.4 Segment: The smallest element considered by crew schedulers; a segment is a flight leg or a composite flight.

1.1.5 Rotation: A trip, or sequence of segments, flown by a crew which originates and terminates at the crew base and which satisfies the restrictions imposed by safety regulations, union requirements, company policy. Each rotation has specific amounts of total flight time, total duty time, and time away from base.

1.1.6 Duty Period: A period during which a crew flies a set of segments without checking out (ie. without a rest period).

1.1.7 Duty Time: Lapse of time between the moment a crew arrives for the briefing of a duty period and the moment it ends the debriefing period; e.g. duty time may spread from

an hour before the first departure of the period to 15 minutes after the arrival of the last flight; the pilots' union contract generally does not allow duty times in excess of 12 or 14 hours in any duty period.

1.1.8 Flight Time: Time between the departure and the arrival of a flight leg. The flight time for a composite flight is the sum of the flight times for the flight legs composing it.

1.1.9 Time away from base: Time between the moment a crew checks in for the first flight of a rotation and checks out of the last one.

1.1.10 Overnight: A crew overnights when they must spend a rest period at a location different from their base.

1.1.11 Deadheading: More than one crew may be allocated to a segment. One crew (or more) must then occupy revenue seats as passengers in order to fly the rotation they have been assigned to ; some carriers allow this to happen in their models when it seems economically justified.

1.1.12 Bid (block): A sequence of rotations building up a monthly assignment for a crew in respect with safety and union regulations and company policy.

1.2 The Crew Scheduling Process

The crew scheduling process is a very large-scale operation for most airlines.

Generally the crew scheduler is given a predetermined schedule of flight legs by the airline schedule department for his planning period; e.g. a week or a month or more. His problem is to create a crew schedule which covers all their flight legs, and which uses the least crews, or incurs the least cost.

The problem is immediately decomposed into a schedule for each aircraft fleet since crews are limited to operate one aircraft type in any planning period. This reduces the dimensions of the problem, since a scheduling process can be performed for each fleet independently.

The output from the crew schedule process is a set of monthly blocks which the crews may bid on. For most American carriers, the pilots' only decision level is in his selection of the monthly bids which is made according to seniority.

The schedule process is carried out monthly at every airline at present, as far as known. The computer approach taken by various airline OR groups is described in figure 1-1. It is the approach used in this report.

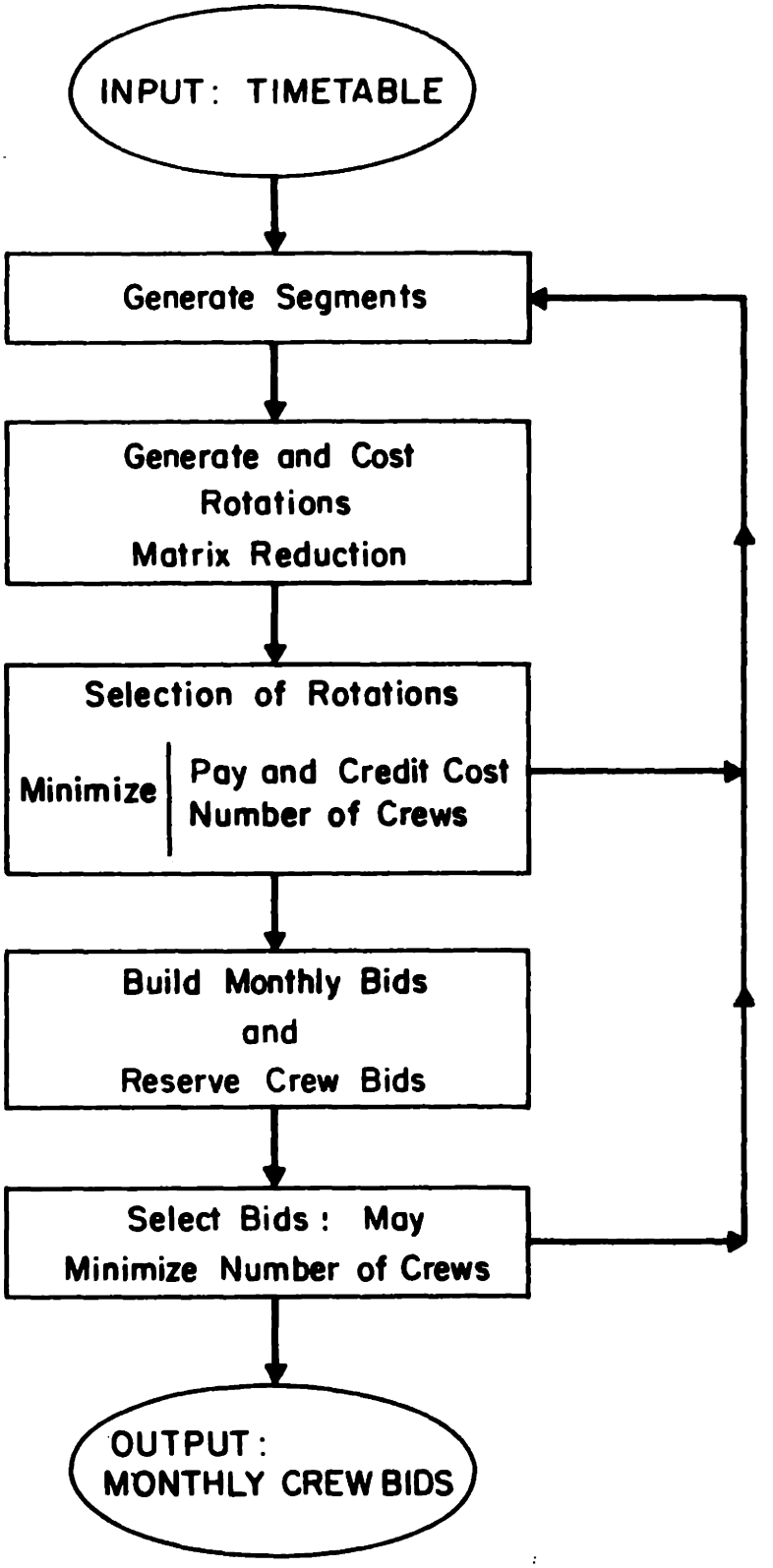


Figure 1.1 THE CREW SCHEDULING PROCESS

The following pages will now present the different steps of the crew scheduling process and the problems associated with each step. The process is described in greater detail in reference (2).

1.3 Segment Generation

The scheduler chooses to accept the flight legs as given or to use his judgement to aggregate subgroups of flight legs into composite flights called segments. This step may be necessary because a first flight leg sends a crew to a city from where the only originating flight on that aircraft type within several hours is the return flight; in this type of situation, creating a composite flight is easily justified. (The OR analyst may also aggregate flights in cases where the economy is not obvious since this may be the only way he can obtain a number of rotations small enough to be accepted by the capacity of his computer model.)

Those flight legs and composite flights are the segments which form the basic element of the mathematical models.

1.4 Rotation Generation

The problem here is to generate and cost all, or a most interesting (e.g. cheapest) subset of the possible rotations. Most airlines have built themselves a computer generator (See reference 2).

There are regulations, legal and union, which limit the

choice of possible rotations. Even then, for an airline operating only 700 segments in a week, over a million feasible rotations may easily be built. To reduce this number, some company rejection rules will often be arbitrarily used and the more expensive or less efficient rotations will not be accepted for selection in the third step of the flow-chart. Analyst judgement again enters the process at this stage.

The cost structure is of significant importance to the rotation generation and selection. There are two common ways to cost rotations:

a) The airline pays a fixed salary to the crew; when bids are offered, the senior pilots therefore choose the bids totaling up to the smallest workload. Each rotation is then flagged with a cost of unity.

b) Most American carriers have a complex costing system called pay and credit which offers the crew flight time credit to compensate for a number of possibly unpleasant situations that would arise in the rotation. A formula may be used to express the credit cost of a rotation. Let us define the components:

FT(R) is the total flight time of the rotation

TAFB(R) is the time away from base of the rotation

FC(R) is the flight time credit of the rotation

N(R) is the number of days of the rotation

FT(i) is the flight time on day i of the rotation

DT(i) is duty time on day i of the rotation

All times are expressed in minutes. For example:

$$FC(R) = \text{Max}\left[\frac{TAFB(R)}{3.5} - FT(R);\ \sum_{i=1}^{N(R)} P(i);\ 0\right] \quad \text{where:}$$

$$P(i) = \text{Max}\left[\frac{DT(i)}{2} - FT(i);\ \ 240 - FT(i)\ ;\ 0\right]$$

This example formula guarantees a minimum of four hours of flight time per duty period, that flight time will be at least ½ duty time in any duty period, and that flight time will be at least 1/3.5 of time away from base for the rotation.

The formula may change each time a new contract is signed between the pilots and the company.

The rotation cost is the sum of that pay and credit (ie. compensation) cost and of the hotel and limousine costs if overnights are involved (although some airlines ignore these latter costs). The time spread of the rotation varies with the airline; the values for several of them, as well as their respective cost structure, may be found in figure 1-3.

The problem created by the pay and credit system is the

non-linearity of the cost. When building a rotation, one may decrease the unit cost of a partial rotation by adding a segment: it is easy to understand this if one considers the rule guaranteeing a crew a credit of four hours of flight time per duty period. Adding a one-hour flight to a partial rotation totaling 3 hours of flight time in the current duty period may save one hour of credit compensation. The existence of these guarantees means that each individual rotation must be generated and costed. It is not possible to attach costs to the basic elements (segments) of the mathematical model. Instead these segments costs are generally taken as zero and only penalties caused by guarantee violations of the complete rotation are of interest as costs. It also makes it impossible to generate rotations by increasing order of pay and credit cost although this would prove very attractive to the OR analyst.

1.5 Rotation Matrix Reduction

When rotations are generated automatically, one may come up with more than a million acceptable rotations, ie. evidently more than any mathematical model would care to use. Some techniques generally must be applied to reduce that number of

rotations in order to make it manageable.

The next step in the crew scheduling process is to select the set of rotations of minimal total cost which cover all segments. To formulate the problem, a 0-1 matrix, A, will be used where:

$$A = (a_{ij}) \quad : \quad a_{ij} = 1 \text{ if rotation } j \text{ included segment } i$$
$$= 0 \text{ otherwise}$$

Columns are rotations and rows represent segments. Fig. 1.2 shows the type of matrix used. The matrix reduction techniques discussed here will reduce the size of A.

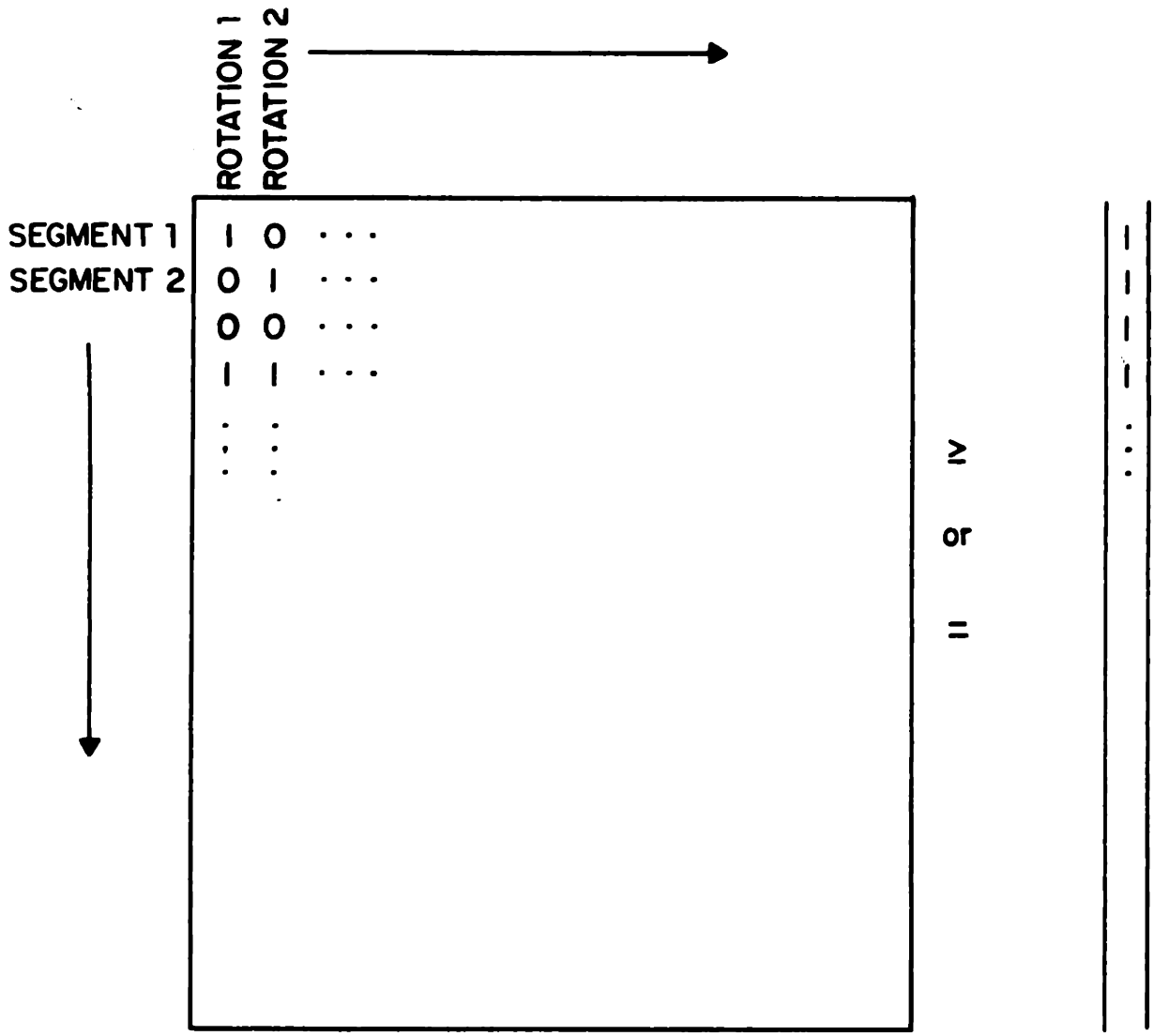
a_j will represent the $j^{\underline{th}}$ column of A

Let us define S_i as the set of columns covering row i, ie. the set of rotations including segment i:

i.e. $$S_i = \left\{ j/ \quad a_{ij} = 1 \right\}$$

Define $$\bar{S}_i = \left\{ j/ \quad a_{ij} = 0 \right\} = \left\{ \text{all columns} \right\} - S_i$$

The following paragraphs describe a number of possible matrix reduction rules, R1 through R6.



Ex: SEGMENT 1 is MONDAY: LOGAN – LA GUARDIA at 8:30 AM

$$\text{Min } z = cx$$

$$Ax \left| \begin{matrix} \geq \\ = \end{matrix} \right| 1$$

$$A, x \text{ BOOLEAN}$$

Figure 1.2 THE ROTATION SELECTION PROBLEM

1.5.1 Deadheading Allowed

The constraint requiring each segment to be covered will be expressed as: $Ax \geq 1$ where $x_j = 1$ if the jth rotation is chosen in the solution and 0 otherwise. $\mathbf{1}$ is a column vector of ones.

R1: if $a_j \subset a_k$ and $c_k \leq c_j$, the optimal solution will not deteriorate when the jth column is deleted from the matrix.

Proof: if column j had belonged to the optimal solution, it could have been replaced by column k at no higher cost.

Remarks: the cost would not be higher if costs were exactly equal to the values in the cost vector. However, if a "deadheading cost" were accounted for, the real-life total cost would be higher if column k replaces column j in the optimal solution, since deadheading will increase. BEA reduces the matrix size from 250 x 25000 to 250 x 3000 with R1 (See reference 2).

R2: if $(a_{j_1} \cup .. \cup a_{j_n}) \subset a_k$ and $c_k \leq \sum_{i=1}^{n} c_{j_i}$, the optimal solution will not deteriorate when the n columns a_{j_i} are deleted from the matrix.

Remarks: the preceding proof and remarks are still valid. R2 shows that reduction may take place on the first level, as in R1, or at any higher level. The problem is that the expected number of columns which will be reduced per second of computer time decreases very sharply as higher levels of reduction are used.

1.5.2 No Deadheading Allowed

The constraint is now $Ax = 1$.

R3: if $S_i \subset S_k$, row i and all columns belonging to $S_k \cap \bar{S}_i$ may be deleted without increasing the cost of the optimal solution.

Proof: if a column covers k and not i, it cannot be included in a feasible solution. If it were, to satisfy row constraint i, one of the column of S_i would have to be selected and that would add a second '1' to row k ($S_k \subset S_i$); row k's constraint could not then be satisfied any more (since $x_j \geqslant 0 \; \forall \; j$).

Row i may be deleted since it is now (after the column reduction) identical to row k; as a consequence, it is a redundant constraint: all solutions satisfying row constraint

k in the reduced problem will implicitly satisfy row constraint i.

R4: if $(S_{i_1} \cup .. \cup S_{i_n}) \subset S_k$, rows $i_1, i_2, ... i_n$ and all columns belonging to $S_k \cap \bar{S}_{i_1} \cap ... \cap \bar{S}_{i_n}$ may be deleted without increasing the cost of the optimal solution.

Remark: the preceding proof applies in the same way. Here again, one must notice that if higher levels of reduction are used, ie. $n > 1$, the operation becomes more costly and the benefit decreases; the benefit/cost ratio therefore deteriorates sharply as n increases.

1.5.3 Non-optimal Reductions

It may happen that, after these reductions, there are still too many columns for the mathematical model. Reductions must then be applied which will force out enough columns to bring the size down to manageable levels in such a way that the cost of an optimal solutions is unlikely to be increased. There are numerous heuristics which may be defined for that purpose; let us just state two of them.

R5: in the automatic generation of rotations, include

tests likely to force out expensive rotations.

Example: refuse rotations for which a duty period has less than, say, four hours of flight time. Several tests of this type may be introduced; another one would be to reject connecting times between segments of more than a certain number of hours, since otherwise the duty time/flight time ratio would probably be bad. This rejection rule is attractive since it stops generating and costing "bad" rotations.

R6: exclude rotations costing more than a given value if a pay and credit cost structure is used. Some segments are "bad" and all rotations including them may be costly; this rule would favor rotations containing the "nice" segments and the matrix would not be balanced. Of course, the cost bound could be a function of the segments included. The reduction would then become more elaborate.

1.5.4 Conclusions on Reduction

One type of reduction was not discussed here, the logical reduction. For example, when only one '1' appears in a row, the column containing it must be included in the solution and all rows covered by it may be deleted. This

type of reduction is both obvious and unlikely to happen considering the large number of rotations which may be generated.

There were three classes of reduction techniques:

a) a class of reduction techniques for $Ax > 1$ which do not increase the cost of the optimal solutions (as defined in the cost vector).

b) a class of reduction techniques for $Ax = 1$ which only delete infeasible columns.

c) a class of heuristic reductions which greatly (and cheaply) reject rotations (partial or complete) and may, although with a small probability, limit the problem to less-than-optimal solutions.

In each of the first two classes, a first-level and multi-level reduction technique were demonstrated. First level reductions delete columns and rows faster and at a smaller cost than multi-level reductions.

One must keep in mind that, in the process of generating, reducing and selecting a covering set of rotations, the analyst should be minimizing the total cost of computation and crew schedule costs :

$$C = CCG + CCR + CCS + CSR$$

where CCR, CCS are the computer costs of these generation, reduction and selection runs, and CSR is the pay and credit costs of the crew schedule.

In fact, the real problem is to minimize C over different computer methods or even manual methods. However, monthly CSR costs are generally large enough that, if computer methods save a small percentage of CSR costs, it would pay for days of computer time. For this reason, the author is persuaded that a completely manual or even heuristic operation will not be a better answer for most airlines. For example, saving 5% of the pay and credit cost using computer methods may represent $250,000/year.

Figure 1-3 indicates the type of problem solved by each airline, the size before reduction, the size after, ... It was obtained from a poll of the airlines engaged or seriously interested in the use of computers for crew scheduling; the poll was made during the summer of 1969. The author is very grateful to the airlines for their cooperation and the speed with which they answered.

Figure 1.3 MATRIX GENERATORS AND REDUCTION

Airline	Number of Days	Matrix Reduction During Generation?	Matrix Reduction After Generation?	Size Before Reduction	Size After Reduction	Matrix Density
Air Canada	1 or 2	No	Yes (and Before)	150 x 50,000	150 x 3000	< 7%
Air France	1	No	No	?	70 x 600	4%
American Airlines	1 to 4	Yes	No	$\simeq 600 \times 2.10^6$	60 x 4000	5%
B E A	1	Yes	No	140 x 3200	100 x 1300	4.5%
K L M (Europe)	1	Yes	No	?	20 x 70	10%
Lufthansa	1	Yes	No	?	180 x 700	5%
S A S	1 to 12	Yes	No	$\simeq$ 200 x 1500	200 x 1000	5%
Swissair	1	Yes	No	?	150 x 1500	?
United Airlines	1 to 4	Yes	Yes	$\simeq$ 120 x 242,000	120 x 5000	6%

1.6 Rotation Selection

The next step in the crew scheduling process is the problem of selecting the set of rotations which covers each segment at the smallest total cost.

That is, a set of columns (rotations) must be selected from the rotation matrix A (see figure 1-2) which will cover each row (segments) at a total minimal cost. Each row must be covered at least once ($Ax \geq 1$) if deadheading is allowed, or exactly once ($Ax = 1$) if it is not.

This mathematical problem is generally referred to as the set covering problem. It is here, where the largest amount of money seems to be involved, that the airlines require a good optimal solution technique. This thesis therefore has given special emphasis to solving the problem better and faster than other computational methods have done in the past. Rotation selection is the only part of the crew scheduling process where all airlines must solve a similar mathematical problem. In other parts of the process, the differences in the contract agreements do not allow a general approach: one cannot write a general purpose rotation generator or even one usable by a reasonable percentage of the carriers. The three following chapters will discuss this problem and different

solution methods along with computational experience.

The formulation of the rotation selection described in figure 1-2 is very useful for the planner trying to find out how many crews he should have at each crew base to operate at an optimal level at a time when he has freedom of action, e.g. when the fleet is not in operation yet. If, however, the planner is operating with an existing fleet, his crews have less mobility and adequate constraints must be included. There are two ways these constraints may be expressed.

1.6.1 Detailed Crew Base Constraints

At each base, a certain number of crews is available, each of them representing a potential number of flight hours which may be flown from the base during the rotation planning period.

The problem becomes: $\text{Min } cx$

such that $Ax = 1$, or $Ax \geq 1$

$$G\,x \leq g$$

$$x \in N$$

where a row constraint in G is added for each crew base, the right hand side being the product of the number of crews in the base by the number of flight hours a crew may fly in the rotation planning period.

Remarks

There are two drawbacks to using this formulation. First, and this is of course the main objection, the problem will lose its nice 0-1 structure and therefore determinant values will blow sky-high (chapter 3 will explain how this presents a problem). Secondly, this formulation would still not involve the time at which the rotation is flown and the flight time load of rotations. A set of rotations may be accepted for a base which requires twice as many crews as available on the first day and no crew on the second day if the planning period is two days. Or a set of rotations may be selected requiring more crews than available to cover rotations which have little flight time and add up to less than the available number of flight hours.

1.6.2 Daily Constraints

Let us now add a different set of constraints to the original problem: one per base per day of the rotation planning period. If there are three bases and a one week period, 21 constraints will extend the problem into:

$$\begin{aligned} &\text{Min } cx \\ &\text{such that} \quad Ax = 1 \text{ or } Ax \geq 1 \\ &\qquad\qquad\quad Hx \leq h \\ &\qquad\qquad\quad x \in N \end{aligned}$$

where H is a 0-1 matrix.

Example:

Rotation #	1	2	3	4. . .	Right hand Side h
Day 1 Base 1	1	1			3
Day 2 Base 1	1	1	1		3
Day 3 Base 1		1	1		3
. . .		. . .			. . .
Day 7 Base 1					3
Day 1 Base 2				1	8
Day 2 Base 2				1	8
. . .		. . .			. . .
Day 7 Base 2					8
Day 1 Base 3					5
. . .		. . .			. . .
Day 7 Base 3					5

If rotation i runs from day 3 to day 5 from base j, column i will have a one in rows $\left[7\ (j-1) + 3\right]$ to $\left[7\ (j-1) + 5\right]$ and zeroes otherwise. The right hand side h contains the number of crews available at the corresponding base.

Remarks

This type of constraint is at the same time more adequate to our method (0-1 matrix) and more realistic, in a sense. The drawback is that, contrary to what could happen with the preceding constraints, the assignments may require an acceptable number of crews to fly a higher than acceptable number of hours.

1.7 Monthly Bids (blocks)

Once the rotations have been selected for every day, or every week, of a month, they must be put together to form the monthly assignments the pilots will be offered. A set of rules limiting the ways rotations may be built into blocks is found in the contract between the pilots and the carrier; these rules vary with the airlines.

These rules guarantee rest periods and vacation periods of some number of consecutive days. The airline must also schedule training periods for the pilots.

1.7.1 Manual Block-Building

This might be the best solution when, for example, the contract requires that a pilot receive the same weekly

assignment each week, or the same daily assignment each duty day of the block. The problem then is only to fit rest periods and training periods in the month, together with duty periods in such a way that a crew flies the maximum number of hours allowed minus a safety measure which can be used up by delays, holding periods over airports, etc.

1.7.2 Heuristic Automatic Block-Building

A cheap and efficient way to build blocks is to create a template of good blocks, where duty periods are alternated with rest periods according to contract rules and in a manner attractive to the crews. A program then maps the duty periods into blocks from the template in such a way that the maximum use of a crew is found, with some margin to take care of unforeseen events.

1.7.3 "Optimal" Automatic Block-Building

The payoff in this case has a smaller expected value, according to experienced airline OR analysts. Some people feel that much money may be saved by very sensitive block-building. Then, a block generator, similar to the rotation generator, could be used and an integer optimization program

run. The author doubts this will be a valid approach since the problem is more difficult to solve than the problem for rotations (larger dimensions and more intricate rules). It would pay off if the expected operational savings over one of the two other approaches were larger than the expected difference in their computational costs for the airline.

1.8 Reserve Crew Assignment

A number of crews must be on reserve to fly charters, replace crews who are sick, crews who, due to delays, have reached their maximum flight time for the month or who have missed a connection.

Two costs are involved in the reserve crew operation: first, the reserve crews are paid the monthly minimum of flight time hours; if too many reserve bids are offered, the airline will have to pay flight hours which were not actually flown. If not enough appear, some flights would have to be cancelled or regular crews would have to be re-scheduled at a high cost. It is therefore important for the airline to properly organize its reserve crew scheduling.

It turns out that reserve crew needs may be much more

predictable than one might be led to think. Investigations have shown that:

1.) There are cities from where crews call sick more often than from others, even when this is prorated to the number of departures from the city;

2.) Crews happen to call sick more often on a Friday morning with a three-day rotation or on a Saturday with a two-day rotation than on other weekdays.

An intelligent survey of field data will probably provide the scheduler with a good feel for reserve crew needs. He may then try to model the needs by simulation or regression analysis. He should develop a model which will inform him on reserve crew needs, and adapt it to include a given level of reliability (e.g. cancel less than one flight/month because of lack of pilots).

Using such an approach, models have been developed which provide a better reliability with less reserve crews. This can occur when the reserve crews may be found at, or flown in time to, the right airport rather than stay inactive where they are not needed.

1.9 Other Types of Crew Scheduling

Apart from airline crew scheduling, another area of crew scheduling presents some interesting problems, the area of transit systems. A typical difference is that, for transit systems, the penalties really appear in the block-building (or bid-building), whereas it happened in the rotation selection problem with the airlines.

A good paper on rotating rosters was given in Transportation Science (ref. 6). It says that the sensitive area of transit crew scheduling is in packing assignments and rest periods, subject to penalties created by "bad" blocks. Making the rotations is not a problem since a crew usually remains on the same transit line for each assignment; there is therefore no flexibility in the rotation generation and selection phase.

Scheduling of crews of stewards and stewardesses has also been studied; it is however, not as sensitive as our problem since there is not such an expensive penalty system in that case. The problem is mainly one of minimizing the amount of personnel needed.

CHAPTER II

ROTATION SELECTION

This chapter will present different possible formulations of the set covering problem which arises in rotation selection and classify groups of solution methods currently used to solve it.

2.1 Formulations

2.1.1 General Set Covering Formulation

This mathematical formulation was given in the introduction. Figure 2-1 presents the set covering formulation of a test problem to be used as an example in this thesis.

In fact, the x vector may only be required to be integer rather than 0-1. Since the cost vector is nonnegative, values greater than one will never provide a better solution than values of one.

In the description of a solution, $x_j = 1$ when rotation j will be flown; if $x_j = 0$, it will not be used.

$$\text{Min } Z = 2x_1 + 4x_2 + 5x_3 + 4x_4 + 3x_5$$

Such that:

$$x_1 + x_2 = 1$$

$$x_1 + x_3 + x_5 = 1$$

$$x_2 + x_3 + x_4 = 1$$

$x \in \bar{B}$, i.e. Boolean (0-1)

$$A = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{vmatrix}$$

$$C = \begin{pmatrix} 2 & 4 & 5 & 4 & 3 \end{pmatrix}$$

$$x = \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{vmatrix} \qquad 1 = \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix}$$

Problem $\quad \text{Min } Cx$

with $\quad Ax = 1$

$x \in \bar{B}$

Figure 2.1 SET COVERING FORMULATION OF THE TEST PROBLEM

This LP formulation is the one most used currently since good integer solutions may easily be found by manually transforming the LP optimum.

2.1.2 Knapsack Formulation for Ax = 1

An equivalent formulation results in a knapsack problem. This formulation is only applicable to the case where dead-heading is not allowed.

Let us multiply each row of the rotation matrix of A by 2^{i-1} where is is the row index. Let us define a row vector b:

$$b_j = \sum_{i=1}^{m} 2^{i-1} a_{ij}$$

Let us also define a counter row vector k:

$$k_j = \sum_{i=1}^{m} a_{ij}$$

which counts the number of segments each rotation has.

Theorem: there is a one-to-one correspondance between feasible sets Ax = 1 and $(bx = 2^m-1) \cap (kx = m)$ for $x \in \bar{B}$.

Proof: it is clear to see that Ax = 1 will result in both $bx = 2^m-1$ and $kx = m$ if $x \in \bar{B}$.

$$\begin{array}{c} 2^0 \\ 2^1 \\ 2^2 \end{array} \left| \begin{array}{ccccc} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{array} \right|$$

$$b = \left| 3 \;\; 5 \;\; 6 \;\; 4 \;\; 2 \right|$$

$$k = \left| 2 \;\; 2 \;\; 2 \;\; 1 \;\; 1 \right|$$

KNAPSACK FORMATION: min cx

such that: $3x_1 + 5x_2 + 6x_3 + 4x_4 + 2x_5 = 7$

$2x_1 + 2x_2 + 2x_3 + x_4 + x_5 = 3$

$x \in \bar{B}$

Figure 2.2 KNAPSACK FORMULATION FOR THE TEST PROBLEM

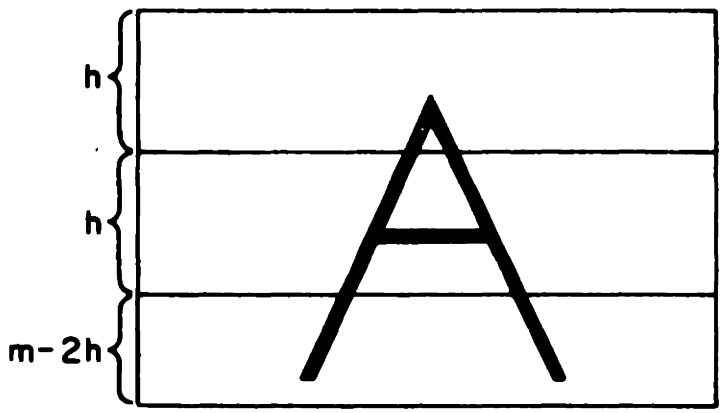
Now the reverse must be proved true. With m bits added at different positions in a binary word (since $kx = m$ and $x \in \bar{B}$), a value of 2^m-1 (since 2^m-1 can only be obtained by putting one 1 in 2°, one in 2^1,...)

So, the knapsack formulation will be

$$\begin{aligned} &\text{Min } c\,x \\ &bx = 2^m-1 \\ &kx = m \\ &x \in \bar{B} \end{aligned}$$

Most rotation selection problems would be too large to be solved by current knapsack codes. Figure 2-2 shows the knapsack formulation of the test problem. The problem has been transformed into a two-dimensional knapsack problem. It is not a knapsack problem in the traditional sense since the constraints are equalities instead of inequalities. As of now, there seems to be no method which solve problems of this type with several thousand variables; in fact, even 500 variables would be too much for a fast solution.

Remark: if m is large, the b_j's will be too large to fit machine words. In that case, A may be divided into horizontal blocks of h rows where h is the highest number of



Treat each row group of A as a complete matrix would be treated if m were small.

Figure 2.3 KNAPSACK FORMULATION WITH m LARGE

rows for which the b_j's will be acceptable. Let $u = \lfloor m/h \rfloor$, ie. the largest integer less than or equal to m/h. Then, vectors $b^1, b^2, \ldots b^{u+1}$ and $k^1, k^2, \ldots k^{u+1}$ may be defined as before where a horizontal block of A will play the role A played. The problem becomes:

$$
\begin{array}{lll}
\text{Min } c\,x & & \\
\text{such that} & b^i x = 2^h - 1 & \text{for all} \\
 & k^i x = h & i = 1, 2, \ldots, u \\
 & b^{u+1} = 2^{m-uh} - 1 & \\
 & k^{u+1} = m - uh & \\
 & x \in \bar{B} &
\end{array}
$$

The matrix structure would then be as in figure 2.3. Of course, solving a six-dimensional or an 8-dimensional knapsack problem is much more difficult than solving a two-dimensional problem.

2.1.3 Network Flow Formulation

This is not an equivalent formulation of the set covering problem. It was tried by the author when he started using a branch and bound approach to solve the problem.

In the network formulation, a network consisting of a set of origin nodes connected to a main source S is linked to a set of destination nodes connected to a main sink T. The origin nodes represent rotations and the destination nodes segments. Arcs link a rotation node to all the nodes of segments included in it.

The test problem would be expressed as shown in figure 2-4 where the numbers on each arc represent the upper bound, the lower bound and the cost: (u, l, c). The cost on each origin arc is the cost of the rotation divided by the number of segments in the rotation. Let us define:

A_j = set of origins of arcs with destination j

A_T = set of origins of arcs with destination T

B_i = set of destinations of arcs with origin i

B_S = set of destinations of arcs with origin S

K_i = number of segments in rotation i

The mathematical formulation is:

$$\text{Min } z = \sum_{i \in B_S} c_{Si} \; x_{Si}$$

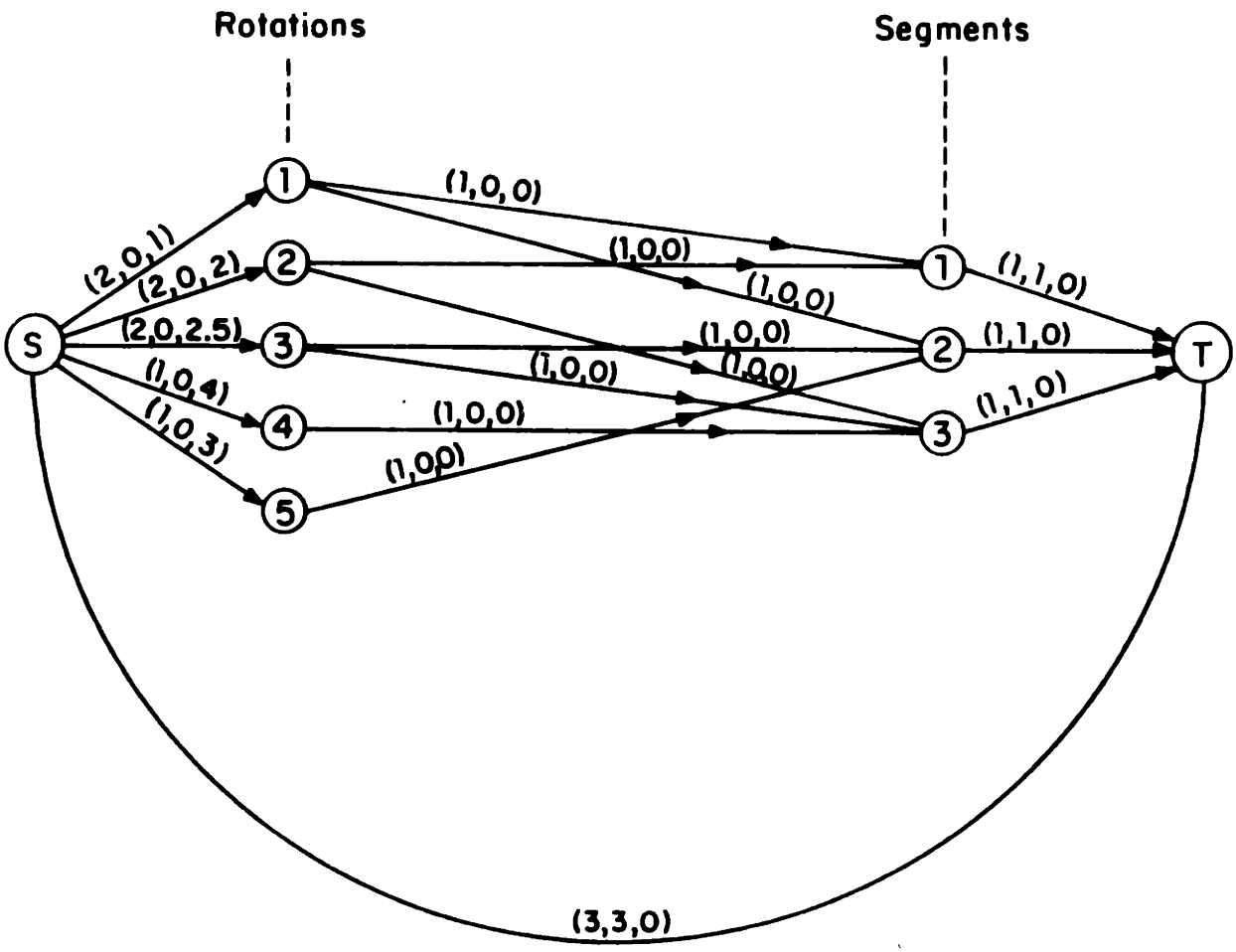


Figure 2.4 NETWORK FLOW FORMULATION OF THE TEST PROBLEM

subject to: 1) $x_{jT} = 1$ (no deadheading)

or $x_{jT} > 1$ (deadheading allowed)

2) $x_{TS} - \sum_{i \; B_S} x_{Si} = 0$

3) $x_{Si} - \sum_{j \; B_i} x_{ij} = 0$

4) $x_{jT} - \sum_{i \; A_j} (x_{ij}) = 0$ for j=1, m

5) $x_{TS} - \sum_{j \; A_T} (x_{jT}) = 0$

6) $0 \quad x_{Si} \leq K_i \quad \forall\, i \in B_S$

7) $0 \quad x_{ij} \leq 1 \quad \forall\, j \in B_i$ for i=1, n

Constraints (2) to (5) guarantee flow conservation at each node. Constraints (1), (6), and (7) define upper and lower bounds. The type of constraint in (1) shows whether deadheading is allowed or not. If c_i is the cost of rotation i, c_{Si} may be defined as:

$$c_{Si} = c_i/K_i$$

The problem may also be formulated in a symmetrical way,

with the origin nodes representing segments and the destination nodes rotations.

Remarks: This problem could be solved by direct inspection: the minimum cost solution is the solution for which: $x_{ij} = 1$ for i such that:

$$c_{Si} = \min_{k \in A_j} c_{Sk}$$

for all j=1, m

All other x values are obtained directly from the flow conservation equations. Such a solution is evidently feasible and optimal.

In the network flow formulation the solution may send flow in some arcs of a rotation but not all. For this reason, some constraints should be added requiring a capacity-or-nothing flow in each arc leading out of the main source S. Branch and bound (see section 2.2.3) should be used, but it does not converge fast enough, compared to branch and bound with LP.

2.2 Solution Methods

2.2.1 Heuristic Methods

For a long time heuristic solution methods have been used which do not guarantee optimality. Some of them are still in use where people have problems too large to solve otherwise, or when they do not know of the existence of methods good enough to solve their problem satisfactorily.

These methods obtain feasible integer solutions based on a limited search of the feasible space; to find these solutions, selection criteria are applied which "should" bring one close to an optimal integer answer.

One typical heuristic method is to solve the problem by an LP code and manually round up the non-integer values in the solution vector to obtain a feasible integer solution, hopefully close to an optimal integer answer but quite possibly far from optimal.

Another heuristic method would be to fix to unity the activity of all variables with unit activity in the LP optimum: this reduction of the feasible set limits the size of the search, but again may result in a feasible solution more expensive than the optimal integer solution.

2.2.2 Implicit Enumeration

Since each of the n variables may be set to 0 or 1, there are 2^n possible solutions. This number is generally too large to find the best solution through an exhaustive search. Implicit enumeration methods reject subsets of solutions which are known a priori not to be feasible or optimal.

A number of elements must be defined first. We follow Geoffrion's approach (Reference 8) in explaining the method.

- A partial solution S is an assignment of binary values to a subset of the n variables.

- A free variable is a variable not assigned any value by S.

- A completion of a partial solution is a solution determined by S together with a binary specification of the values of the free variables.

- A partial solution is "fathomed" if all its completions have been considered implicitly or explicitly. (S, z) represent a partial solution S and its cost z.

- Notational convention: j denotes $x_j = 1$ and -j denotes $x_j = 0$. Example: if n = 4 (4 variables), S = (3,-4,1) is a partial solution for which $x_1 = 1$, $x_3 = 1$, $x_4 = 0$ and x_2 is free. There are two possible completions: $S_1 = (3,-4,1,2)$ and $S_2 = (3,-4,1,-2)$.

A sequence of partial solutions is generated and all their possible completions are considered. The best current feasible solution is stored together with its cost. Partial solutions are progressively completed. At each step, one of three situations arises:

a) a better feasible solution is found; it then replaces the current optimal solution S*. The next partial solution is then considered.

b) or it is clear that all completions of a partial solution will be infeasible or more expensive than S*; go to the next partial solution.

c) or nothing can be said about S; assign a binary value to one of the free variables which therefore augments S. Test to find out whether (a), (b), or (c) is now valid.

At some point, there will be no partial solution left to be considered. All solutions will have been implicitly or explicitly covered. The optimal solution is the final S*.

The representation of S must be such that it is possible to recognize whether its other binary value has already been assigned to a given variable, the other variables being equal. For example, a variable will be underlined if the partial solution formed by the variables preceding it in S with their current value and the variable at its other binary value

has already been fathomed. Example: S = (3,-4,$\underline{1}$) indicates that (3,-4,-1) has already been fathomed.

The next partial solution is obtained by complementing the rightmost not underlined variable of S and dropping all elements to its right. To complement, underline the variable and assign to it its other binary value. The next partial solution of (3,-4,$\underline{1}$) is (3,$\underline{4}$). It is clear that, through this procedure, the whole set of solutions has been fathomed when a partial solution has been evaluated for which all variables are underlined.

This procedure allows a complete search with a minimum of backtracking effort and table management. Computational speed is sacrificed for this advantage.

The flow chart in figure 2-5 indicates the general outline of an implicit enumeration method.

Many different codes have been built, using different types of tests to fathom S; many different criteria have also been found for the choice of the variable which must augment S. The author started his work on Crew Scheduling by writing and programming an implicit enumeration code (ref 22) based on simplifications of Geoffrion's method (ref 8) allowed by the combinatorial structure of the problem. Special emphasis was given to obtaining a good initial solution

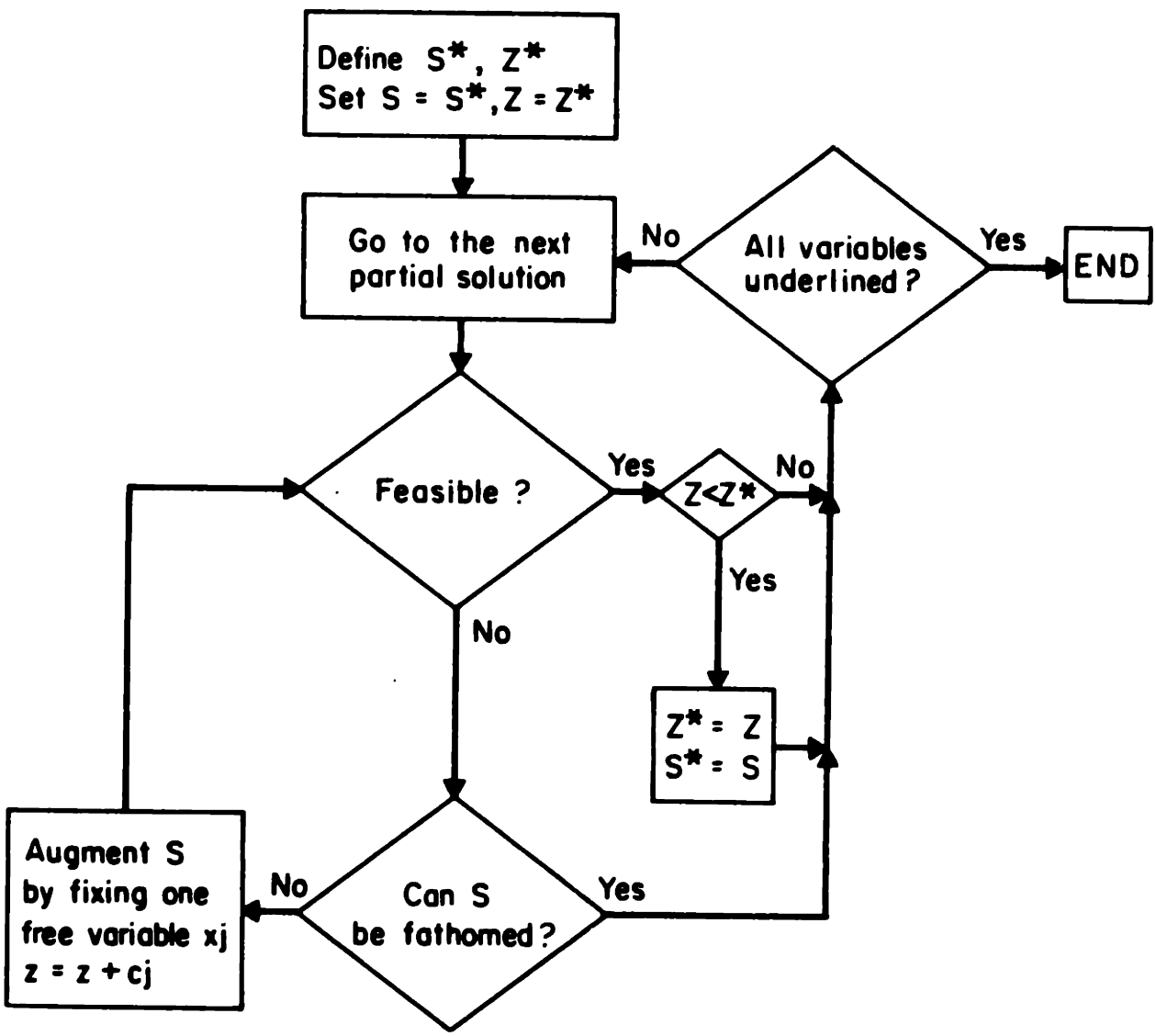


Figure 2.5 IMPLICIT ENUMERATION

(S*, z*) and ordering it in an efficient way; obtaining a good S* is important, since one may only progress sequentially from one solution to another. A good ordering is also of major importance since the leftmost variable will stay in S very long and having fixed that variable at a "bad" level originally will leave the method with a bad bound (z*) for most of the time; as a consequence, many solutions will be considered explicitly which would have been directly fathomed with a better bound.

Experience with this code soon indicated that for problems of the size encountered in crew scheduling the computational times would be excessive. For small problems, it seemed very fast and efficient.

2.2.3 Branch and Bound

Basically, branch and bound corresponds to first solving the LP problem without integrality constraints. If the solution is integer, it is an optimal integer solution. Otherwise, select a non-integer variable x_j and solve two problems where the constraints $x_j = 0$ and $x_j = 1$ have been added.

The feasible set for the continuous problem can be partitioned into three sets, $x_j = 0$, $x_j = 1$ and $0 < x_j < 1$. The third set contains no feasible integer solution and

therefore does not interest us. The optimal integer solution is the minimum of the integer solutions for $x_j = 0$ and $x_j = 1$.

A tree is built where the original node corresponds to a continuous optimum. Each node either branches off to two nodes corresponding to opposite values of a variable not integer in its solution; or is a terminal node. At each iteration, the terminal node with the smallest solution cost is used for branching, until such a node has an integer solution. This will be an optimal integer solution.

The author has experimented with branch and bound using LP (linear programming) and the Out-of-Kilter method (network flow formulation) as a subproblem. Convergence towards the integer optimum was much faster with the LP formulation.

The MPS (Mathematical Programming Systems) for IBM 360 was used as a subprogram by branch and bound to find the optimal integer solution to two sample problems. On the smaller problem, 104 rows by 132 columns, the continuous optimum cost z = 8817.5. The tree in figure 2.6 is what the branch and bound process resulted in.

On the larger problem, 104 x 236, it took seven nodes and around three minutes of MPS time to obtain the optimal integer solution of 14145. Figure 4-6 on the computational results references the different problems tested during this thesis research.

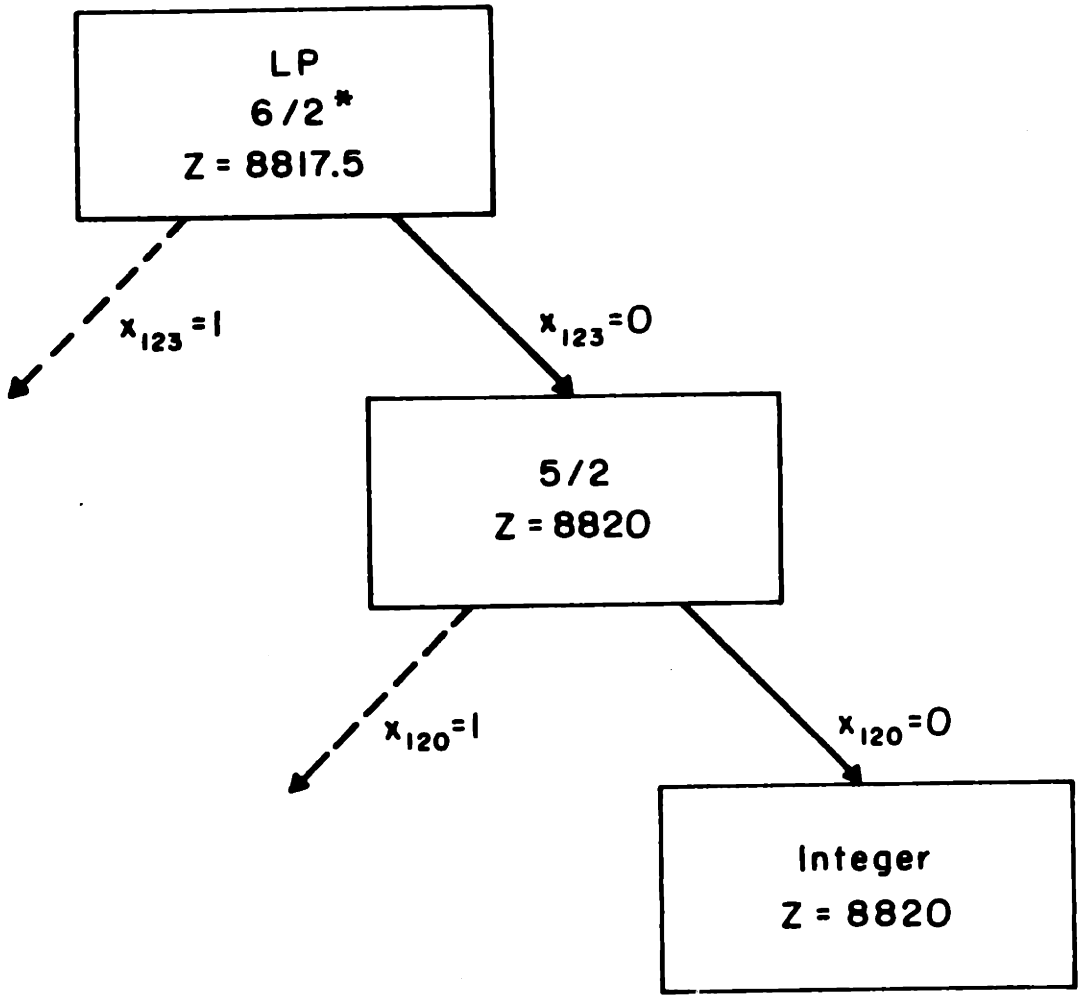


Figure 2.6 BRANCH AND ROUND WITH PROBLEM AA-I

The column selected for branching was the non-integer column with the largest number of non-zero elements. At each level, only one branch was executed; since all column costs were multiplicands of 5 and a feasible integer solution was found at a cost of 2.5 more than the continuous optimum, it was clear no better feasible integer solution would be found. If this had not been the case, two more nodes should have been formed.

Remarks: A number of selection criteria may be tested. The suggestions by Healy in (ref. 12) are quite interesting: using values in the simplex tableau, one may project lower bounds on the objective for the two branches out of a node if a variable were pushed to zero or one. Using a good integer solution as a bound, it is possible to force several variables to integer values in one step, lest the lower bound on the objective be greater than the good integer solution. Using this technique on the preceding problem, the author fixed five variables to integer levels directly after running the continuous LP (before the branching); the resulting optimum was the optimal integer solution. It took .26 min. to have the continuous optimum, .09 min. to send the relevant information from MPS to a file, .13 min. to access the FORTRAN program on file which fixed some variables to zero or one,

execute it and return, and .10 min. to obtain a new solution which was integer (and therfore optimal integer). The total time was therefore .58 min., ie. very fast for the optimal integer solution to a 104 x 132 problem.

2.2.4 Cutting Plane Methods

The principle underlying these methods is to run a continuous linear program, add a constraint which will cut off part of the convex polyhedron of the continuous feasible set without cutting off any integer solution which could be optimal. In fact, many methods could be described as "cutting plane" which bear different names. This is probably the most extensively studied area of integer programming.

The problem is first solved as a continuous linear program. If the optimal solution is not integer, a constraint is added which is a linear combination of the regular constraints from $Ax \geq 1$ or $Ax = 1$ and of some of the integrality constraints. This process takes place until an integer solution is found to the continuous linear program. This solution is the optimal integer solution since it was obtained through an intersection of the original feasible set with some of the integrality constraints.

Now that very good cuts and primal algorithms have been

found, permitting intermediate feasible solutions, it seems that, for many 0-1 problems, cutting plane methods should be second best to only the group theoretic method; only branch and bound methods compare to cutting plane methods and sometimes can provide a faster optimum.

2.2.5 Group Theoretic Method

This method will be explained in detail in the following chapter. It transforms the regular integer linear programming problem into a knapsack-type problem. The procedure is to obtain the continuous optimum and, from there, to formulate the problem as a simple one-dimensional knapsack problem whenever possible. The optimal integer solution is then calculated through a limited search; it is so simple that in many cases, the optimal integer solution is found by direct inspection of some output from the solution by MPS of the continuous problem.

2.3 Evaluation of the Methods

A description of the different methods the author tried will provide some basis for evaluation. The list of the different methods is given in figure 2-7, the highest one

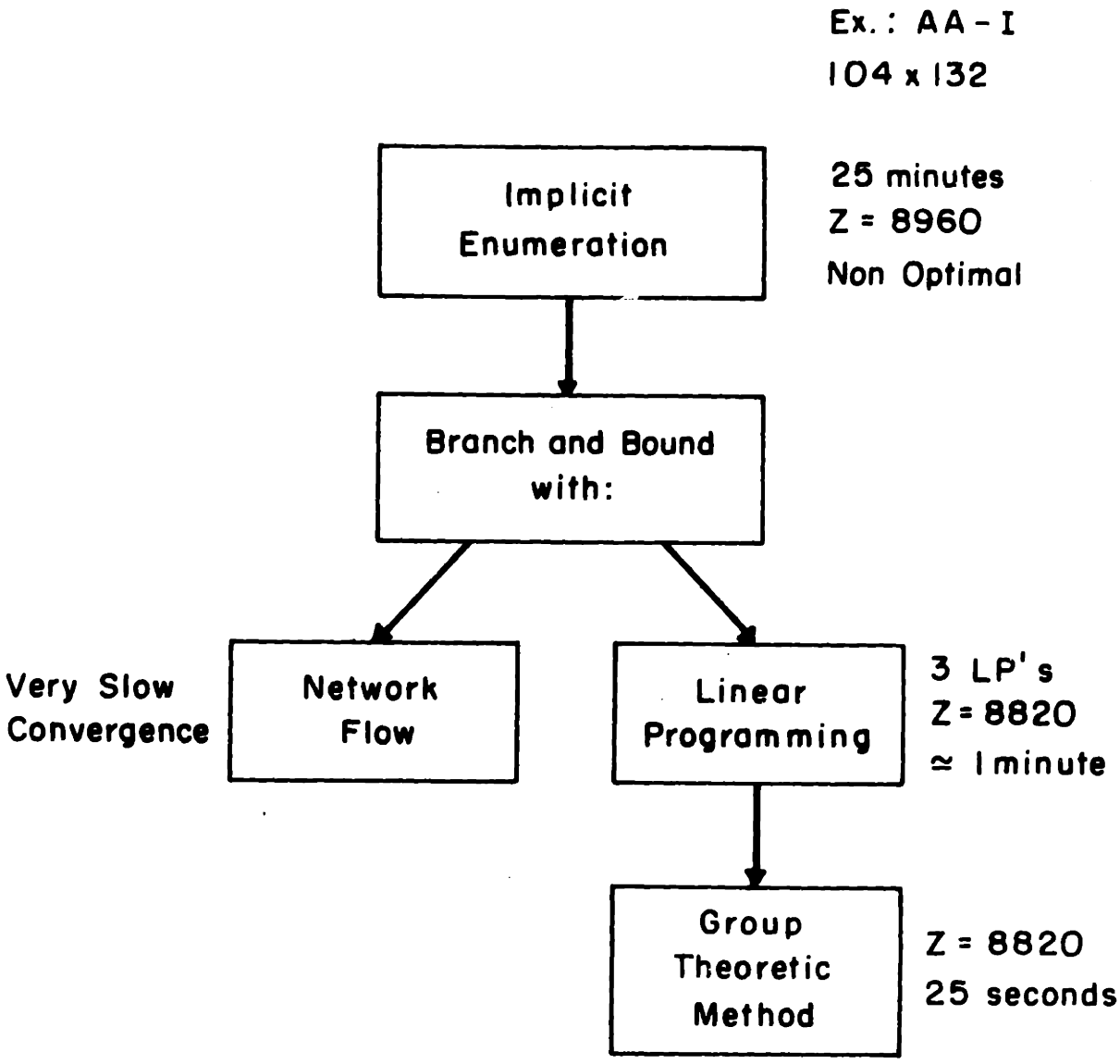


Figure 2.7 PROGRESSION OF RESEARCH

being the first method experimented with, and so on. The first technique used was implicit enumeration where the algorithm described in (ref. 22) was used. The computation time was very small for small problems but seemed to increase combinatorially with size. This is quite logical when considering the way implicit enumeration operates.

The idea then occurred that branch and bound should be more efficient for crew scheduling problems; the same problem was solved optimally by branch and bound in around one minute and group theory in 25 seconds. After 25 minutes with implicit enumeration, the best solution reached was not optimal. Of course, better implicit enumeration codes exist although they would probably all take more than 30 seconds to reach the optimum.

CHAPTER III

GROUP THEORETIC METHOD

3.1 Introduction

This chapter will introduce the mathematical formulation of the group theoretic method and discuss the relations and applicability of this method to the solution of the rotation selection problem.

An effort was made to present this chapter in an easily understandable manner since there are already several articles describing the group theoretic method in a compact and mathematically elegant but not very readable manner.

The following section of this chapter derives the mathematical formulation of the problem. A small sample problem is then completely treated. A discussion of the determinant values one may expect from the continuous optimum to the crew scheduling problem follows. An explanation of the assumptions made in this thesis and of their implications conclude the chapter.

3.2 Mathematical Formulation

3.2.1 Introduction

Let us denote by $\bar{B}$ the set of Boolean values: an element, a vector or a matrix belonging to B is exclusively made up of 0's and 1's. Denote by N the set of non-negative integers.

The rotation selection problem is written in canonical form:

$$\text{P1.} \quad \begin{aligned} &\min \quad z = cx \\ &\text{subject to} \quad Ax = 1 \\ &\qquad\qquad\quad x \in N \end{aligned}$$

where A is an mx (m+n)-dimensional matrix, $A \in \bar{B}$

c is an (m+n)-dimensional vector: $c_j > 0 \ \forall j$

x is an (m+n)-dimensional vector: $x \in N$

1 is an m-dimensional vector of 1's

It is possible to consider A (m,n)-dimensional and c and x n-dimensional. Adding a unit matrix only guarantees the existence of a feasible solution to P1. P1 will be called the BLP, binary linear programming problem. When $x \in N$ is relaxed in the BLP, we are reduced to P2., the simple continuous linear programming problem, or LP problem.

$$\text{P2.} \quad \begin{array}{l} \text{Min } z = cx \\ \text{subject to } Ax = 1 \end{array}$$

Now, if B represents the basis of the optimal solution to P2, P1 can be rewritten as:

$$\text{P1.a)} \quad \begin{array}{l} \text{Min } c_B\, x_B + c_R\, x_R \\ \text{s.t.} \quad Bx_B + Rx_R = 1 \\ \text{where} \quad x_B,\ x_R \in N \end{array}$$

Since B is the optimal LP basis, a set of conditions holds true:

B is a non-singular m x m matrix

$$\text{(1)} \quad B^{-1}\ 1 \geqslant 0$$

$$\bar{c}_j = c_j - c_B B^{-1}\, a_j \geqslant 0 \quad (j=1,2,\ldots,m+n)$$

a_j denotes the jth column of A.

The third condition implies: $c_B x_B \leqslant cx \quad (x \in N,\ Ax = 1)$

Solving for x_B, a new formulation is obtained.

$$\text{P1.b)} \quad \begin{array}{l} \text{Min } c_R x_R + c_B B^{-1}\ (1 - Rx_R) \\ \text{s.t.} \quad x_B = B^{-1}\ (1 - Rx_R) \\ x_R,\ x_B \in N \end{array}$$

For a fixed basis, the value of $z = (c_B\ B^{-1}\ 1)$ is a constant and therefore does not influence the optimization

process. P1b) can then be expressed as a problem in finding the best x_R to:

$$\text{Min } \bar{c}_R x_R = c_R x_R - c_B B^{-1} R x_R$$

P3 s.t. 1) $x_B = B^{-1} 1 - B^{-1} R x_R \geq 0$

2) x_B, x_R integer

3) $x_R \geq 0$

where $\bar{c}_R = c_R - c_B B^{-1} R \geq 0$ because of the optimality condition in (1).

First, let us temporarily drop the condition requiring x_B to be non-negative. Secondly, x_B will be integer if $B^{-1} 1$ and $B^{-1} R x_R$ differ by an integer value, ie. if $B^{-1} R x_R = B^{-1} 1 \pmod 1$.

$$\text{2b)} \quad \left\{ B^{-1} R - \left[B^{-1} R \right] \right\} x_R = \left\{ B^{-1} 1 - \left[B^{-1} 1 \right] \right\} \pmod 1$$

where $\left[B^{-1} R \right]$ is the integer portion of $B^{-1} R$, etc.

Thus, since all elements of $B^{-1} R$ and $B^{-1} 1$ are fractions of D, the determinant of B, the constraints become:

$$\text{2c)} \quad D \left\{ B^{-1} R - \left[B^{-1} R \right] \right\} x_R = D \left\{ B^{-1} 1 - \left[B^{-1} 1 \right] \right\} \pmod D$$

One may now formulate a reduced or group theoretic

problem which finds the cheapest x_R vector to:

$$\text{Min } z = \sum_{j=1}^{n} \bar{c}_j x_j$$

$$\text{P4} \quad \text{s.t.} \quad 1) \quad \sum_{j=1}^{n} \alpha_j x_j = \alpha_0 \pmod{D}$$

$$2) \quad x_j \in N \qquad \forall j = 1,n$$

$$\text{where} \quad \alpha_j = D\left\{B^{-1} a_j - \left[B^{-1} a_j\right]\right\}$$

$$\alpha_o = D\left\{B^{-1} 1 - \left[B^{-1} 1\right]\right\}$$

The following sections show that if x_R is the optimal solution to P4 and if $\left(x_B = B^{-1}(1 - Rx_R); x_R\right)$ is feasible in P1, i.e. non-negative, it will be an optimal solution to the BLP formulated in P1.

3.2.2 Validity of the Group Formulation

Th. I: Any feasible solution (x_R, x_B) to P1 is such that x_R is a feasible solution to P4.

Proof: this is clear since P4 was obtained from P1 by relaxing $x_B \geqslant 0$. In fact,

$$FS(P1) = \left[FS(P4) \cap \{ x/x_B \geqslant 0 \} \right] \subset FS(P4)$$

where FS (Pi) = feasible set of problem (Pi).

Th. II: if a vector (x'_R, x'_B) is cheaper than (x''_R, x''_B) in (1), x'_R will be cheaper than x''_R in P4.

Proof: this is due to the fact that the two objective functions only differ by a constant. They must therefore provide the same ordering for respective feasible vectors and Th. I proved that $FS(P1) \subset FS(P4)$

Th. III: an optimal solution to P1 is a cheapest solution $\bar{x}_R$ to P4 for which the corresponding $(\bar{x}_R, \bar{x}_B)$ vector is feasible in P1, i.e., for which:

$$\bar{x}_B = B^{-1} 1 - B^{-1} R \bar{x}_R \geqslant 0$$

Proof: if there were a better solution(x^*_R, x^*_B) to P1, theorem I proves that x^*_R would be feasible in P4. Theorem II proves that it would be cheaper than $\bar{x}_R$ in P4; so, $\bar{x}_R$ is not the cheapest solution to P4 for which $\bar{x}_B \geqslant 0$!

Some basic remarks can be made about the impact of group

theory on this method. Proofs for the following statements can be found in reference 17.

Remark I: the m-dimensional integer column vectors $\{\alpha_j\}$ of P4 generate a module M in m-space over the ring of integers Z (ref.17 pp. 261-265):

$$M = \left\{ \sum_{j=1}^{n} \alpha_j x_j / x_j \in N \right\}$$

M can be considered to be an abelian group without loss of generality. Let a subgroup P be defined as:

$$P = \left\{ \sum_{j=1}^{m} De_j \; x_j / x_j \in N \right\} \cap M$$

where e_j is the jth unit vector in m-space.

Remark II: Then the factor group G = M/P is abelian and has D elements (ref. 17 , pp. 278-282). Basically, remark II means that there is a one-to-one correspondence between D and the number of possible x_j columns: there are at most D different α_j columns in problem P4.

It can be deduced (ref. 17) that, given M and P, there exists unique positive integers $q_1, \ldots, q_r$ such that $\left\{ q_i \text{ divides } q_{i+1} \; \forall \; i \leq r-1 \right\}$ and $D = \prod_{i=1}^{r} q_i$ and:

$$G = M/P \simeq Z_{q1} \oplus \ldots \oplus Z_{qr}$$

Z_{q_i} = residue class of integers modulo q_i

$A \oplus B$ = direct sum of A and B

"$\simeq$" means "is isomorphic to".

3.3 Prime Determinants

3.3.1 Implications

When D is a prime number, G will be isomorphic to a single residue class of integers (modulo D). In that case, there will be only one dimension to the problem and the x_j's will be completely identifiable by one row element. This greatly simplifies the formulation of P4 which will become:

$$\text{Min } z = \sum_{j=1}^{n} \bar{c}_j x_j$$

P5 $$\text{s.t.} \quad \sum_{j=1}^{n} \bar{\alpha}_j x_j = \bar{\alpha}_o \pmod{D}$$

$$x_j \in N \quad \forall j = 1,n$$

where $\bar{\alpha}_j = \alpha_{kj}$

and $k = \text{Min} \left\{ i/\alpha_{io} \neq 0 \right\}$, i.e., k is the first row for which α_o has a non-zero element.

The constraint is now a single equation rather than a set of equations. The optimal solution to P5 is found by inspection using only the $\bar{\alpha}_j$ and $\bar{c}_j$ values. This is a key to the solution methods presented here.

The determinant usually found for the basis of the continuous optimum of the rotation selection problem is very small and often a prime number: 1,2,3,5, and 7 being very common values. The following paragraphs will discuss the reason for small determinant values.

3.3.2 Remark on Determinant Values

Let us consider what type of determinant a 0-1 matrix with a small density will generate.

The number of rows in the real life problems ranges from around 75 rows up to 1000 or more. The maximum number of segments in any rotation is much smaller: from five to fifteen depending on the case. The matrix densities are therefore very small.

Let us call M_n a 0-1 square matrix with n rows. The determinants D_n of M_n matrices have one interesting property.

$$
\bar{M}_n = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & & & \\
0 & 1 & 1 & 0 & 0 & & \vdots & \\
1 & 0 & 1 & 1 & 0 & & & (0) \\
0 & 1 & 0 & 1 & 1 & & & \\
 & \cdots & & & & & & \\
 & & & & & 1 & 1 & 0 & 0 \\
 & & & & \vdots & 0 & 1 & 1 & 0 \\
 & & & & & 1 & 0 & 1 & 1 \\
 & & & & & 0 & 1 & 0 & 1
\end{bmatrix}
$$

Figure 3.1 MATRIX $\bar{M}_n$

Property: $D_n \leqslant U_n$ where U_n is the nth element of a Fibonacci series: $U_1 = 1$, $U_2 = 1$, $U_i = U_{i-1} + U_{i-2} \; \forall \; i \geqslant 3$

U_n	1	1	2	3	5	8	13	21	34	55
n	1	2	3	4	5	6	7	8	9	10

One matrix reaching that maximum determinant is given here as an example: it consists of a diagonal of 1's; the transversal line over the diagonal is also filled with 1's; the upper part of the matrix is 0's. From the diagonal down to the left corner, transversal lines of 0's and 1's alternate. Matrices such as $\bar{M}_n$ (see figure 3.1) have a determinant of U_n. A large set of matrices of determinant U_n can be obtained by transposing rows and columns of $\bar{M}_n$. The U_n series represents upper bounds for determinants of M_n matrices. These bounds will never be reached by bases of the optimal continuous solution of

Min cx, $Ax \geqslant 1$ or $Ax = 1$, A Boolean since the density of 1's in these bases will never be high enough for those high determinant values to be reached.

3.4 Degenerate Groups

A group is cyclic when the determinant is not a

prime number but each α_j column may be expressed as a product of an integer $\leqslant$ D by one α column (modulo D). Then, the knapsack problem, instead of being multi-dimensional, becomes one-dimensional again.

Suppose D = 4, if there are columns of the type

$$\alpha_k = \begin{vmatrix} .25 \\ .5 \\ 0 \end{vmatrix} \quad \text{and} \quad \alpha_l = \begin{vmatrix} .5 \\ .5 \\ .25 \end{vmatrix} \quad , \quad \text{the group}$$

is acyclic.

If, however, all columns are of one of the four following types:

$$\begin{vmatrix} .75 \\ .5 \\ .5 \end{vmatrix} , \quad \begin{vmatrix} .50 \\ 0 \\ 0 \end{vmatrix} , \quad \begin{vmatrix} .25 \\ .5 \\ .5 \end{vmatrix} \quad \text{or} \quad \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix}$$

$$\text{i.e.} \quad \begin{vmatrix} .25 \\ .5 \\ .5 \end{vmatrix} \times 3 \; , \; \begin{vmatrix} .25 \\ .5 \\ .5 \end{vmatrix} \times 2 \; , \; \begin{vmatrix} .25 \\ .5 \\ .5 \end{vmatrix} \times 1 \quad \text{or} \quad \begin{vmatrix} .25 \\ .5 \\ .5 \end{vmatrix} \times 4 \quad (\text{mod } 1)$$

the group is cyclic and the problem one-dimensional. In such a case, the solution method presented here is still applicable and problem P5 may be solved.

It has been said that many groups for which D is not a prime number are still cyclic. This statement is not corroborated by our computational experience. As a matter of fact, it seems to be an exception rather than a frequent situation. Figure 3-2 in section 3.5 lists the

determinant values obtained for several problems and shows only one case of a cyclic group.

3.5 Pseudo-Degeneracy

The values of non-integer vectors at the LP optimum are all fractions with a smallest common denominator $\bar{D}$. It is clear that $D = n\bar{D}$ where n is an integer.

Theoretically, one should operate in the optimization process with D. However, when all non-integer variables in the basis at the continuous optimum are fractions of $\bar{D} < D$, only a minority of the $B^{-1} a_j$ columns will be made of fractions of D and not of $\bar{D}$. Therefore the possibility of needing any of those columns for an optimal integer basis is very slight. In fact, it did not happen in any of the problems we experimented with. Having $\bar{D} < D$ is what we call pseudo-degeneracy. This happened in each problem tested here where the determinant was a non-prime number greater than 4.

Figure 3.2 details the values of $\bar{D}$ and D obtained so far. An * next to the dimension of a problem means that the problem was modified from the original by fixing some variables to given activities. No answer was provided

Figure 3.2 $\bar{D}$ AND D OBTAINED AT LP OPTIMUM

Airline	Size (rows x columns)	$\bar{D}$	D	Cyclic ?
Air Canada : AC	74 x 739	1	2	
Air France : AF	67 x 536	2	2	
American Airlines : AA-I	104 x 132	2	2	
American Airlines : AA-III	104 x 236*	4	4	Yes
American Airlines : AA-II	104 x 236	7	7	
United Airlines : UAL	117 x 4845	12	24	No
British European Airways : BEA-I	98 x 1652*	21	126	No
British European Airways : BEA-II	84 x 854	60	120	No

in the last column when D was prime, since it would then have no information value.

3.6 Example Problem

In order to illustrate the previous analysis, a simple application will be given here.

$$\text{Min } z = 2x_1 + 4x_2 + 5x_3 + 4x_4 + 3x_5$$

such that:

$$\begin{aligned} x_1 + x_2 \qquad\qquad\qquad\qquad &= 1 \\ x_1 \qquad + x_3 \qquad\quad + x_5 &= 1 \\ x_2 + x_3 + x_4 \qquad\quad &= 1 \end{aligned}$$

and $x \in N$

$$A = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{vmatrix}$$

The continuous optimum is reached for $x_1 = x_2 = x_3 = \frac{1}{2}$, $x_4 = x_5 = 0$, $z = 5.5$

Then: $B = \begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$, so: $B^{-1} = \begin{vmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{vmatrix}$

$$\alpha_o = D\left\{B^{-1}\,1 - \left[B^{-1}\,1\right]\right\} = 2\begin{vmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}$$

$$\alpha_4 = 2\begin{vmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{vmatrix} = \begin{vmatrix} -1 \\ 1 \\ 1 \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix} \pmod 2$$

$$\alpha_5 = 2\begin{vmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{vmatrix} = \begin{vmatrix} 1 \\ -1 \\ 1 \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix} \pmod 2$$

So, both x_4 and x_5 are eligible to enter the basis.

$$c_R^* = c_R - c_B\, B^{-1}\, R = (4\ 3) - (2\ 4\ 5)\begin{vmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{vmatrix}\begin{vmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{vmatrix}$$

$$c_R^* = (\tfrac{1}{2}\quad 1\tfrac{1}{2})$$

Therefore, setting $x_4 = 1$ and making the related changes increases the cost by $\frac{1}{2}$. Doing it for x_5 will increase the cost by $1\frac{1}{2}$. There remains to check for feasibility.

$$x_B = B^{-1}\, 1 - B^{-1}\, a_4$$

$$x_B = B^{-1}\left\{\begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix} - \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix}\right\} = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix}$$

So $x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 1,\ x_5 = 0$

$z = 5.5 + .5 = 6$

Optimal Solution $\quad x_1 = x_4 = 1$

$z = 6$

One needs only check that $x_B = B^{-1} 1 - B^{-1} a_5 \geq 0$ to find that ($x_2 = x_5 = 1$, $z = 7$) is another feasible integer solution. This is why it is so easy with the group method to generate a large group of optimal or near-optimal solutions at very little cost over that of finding an optimal solution.

3.7 Practical Group Problem Formulation for $\bar{D}$ Prime

The optimization technique would be evident if one of the first best solutions to P5 were feasible in P1 since those solutions are easily found by direct inspection with our regular D values. However, this unfortunately is almost never the case in crew scheduling. Let us consider the case where $\bar{D}$ is prime. Let us call the zero group the group Z of α_j columns (or $\bar{\alpha}_j$ values) with nothing but zero elements, $Z = \left\{ \alpha_j / \alpha_{kj} = 0 \ \forall \ k = 1, m \right\}$. Let Y be the set of all columns or combinations of columns with a reduced cost of zero (usually not empty!), $Y = \left\{ \alpha_j / \bar{c}_j = 0 \right\}$. Any subset of columns from $Y \cap Z$ could be set to an activity of one without changing the value or feasibility of a solution to P5. As a result, if one finds the optimal solution to P5 infeasible in P1, all other combinations

of this column with any subset of columns from $Y \cap Z$ must be tested for feasibility implicitly or explicitly before rejecting the column. Moreover, finding the optimal solution to P5 when there are several columns in $\bar{Z} \cap Y$ would become, by the same token, more involved than simple inspection. Those columns could also be combined to form additional columns for Y or Z, e.g. two columns j_1 and j_2 of $\bar{Z} \cap Y$ such that $\alpha_{j1} = 2$ and $\alpha_{j2} = 5$ combine to form a column of Z when D = 7 since 5+2 = 0 mod (7). $Y \cap Z$ is defined to contain all such columns as well.

An assumption was made which has proven valid on all examples treated to date. There is however no reason to believe it should always hold; counterexamples may be found or generated easily.

<u>Assumption</u>

When $\bar{D}$ is a prime number, the optimal solution to P1 can be found by solving P5a instead of P5 and otherwise processing as before.

$$\text{Min } z = \sum_{j=1}^{n} \bar{c}_j x_j$$

P5a

$$\text{s.t.} \quad \sum_{j=1}^{n} \alpha_j x_j = \alpha_o \qquad (\text{mod } D)$$

$$x \text{ integer}$$

$$\alpha_j \notin Z \ \forall j = 1,n$$

3.8 Proof of Optimality for $\bar{D}$ Prime

As will be seen in Chapter 4, situations will arise where only a subset of all nonbasic columns will be considered for the solution of P5. Then, if optimality must be proved and if the expense of the additional computer time seems justified, this may be done in two ways.

One solution is to write a new program or modify the current program so that every possible combination of columns is covered. This is the expensive approach since one would waste much time searching dead ends. The other approach is to formulate a new linear program which proves optimality once the "optimal" integer solution is found with the current program.

Let c* be the sum of the reduced costs of the columns sent into the basis for the "optimal" integer solution;

i.e., this integer solution costs c* more than the continuous optimum.

To formulate the problem proving optimality, one must start from the set of columns for which $\bar{c}_j < c^*$;

let S be the set of such columns:

$$S = \{j / \bar{c}_j < c^*\}$$

Associate with each element of S the $\bar{\alpha}_j$ defined in section 3.3 and its reduced cost $\bar{c}_j$.

Let us partition S into subset S_i where:

$$S_i = \{j \in S / \bar{\alpha}_j = i\}$$

so, $$S = S_0 \cup S_1 \ \ldots \cup S_{D-1}$$

Let t be the number of elements of S.

Let C be the matrix formed by the $B^{-1}a_j$'s for $j \in S$, where the columns are ordered by increasing $\bar{c}_j$'s.

The problem is:

$$\text{Min} \sum_{i=1}^{t} \bar{c}_i \, u_i$$

such that:

1. $Cu < B^{-1} 1$

2. $\sum_{i=1}^{t} \bar{\alpha}_i \, u_i - Dv = \bar{\alpha}_o$

P6 $$3. \quad \sum_{i=1}^{t} \bar{c}_i \, u_i < c^*$$

$$4. \quad u \in \bar{B} \text{ and } v \in N$$

The first constraint set expresses the condition of feasibility in the original problem ($x_B \geq 0$).

The second constraint is equivalent to $\sum \bar{\alpha}_j x_j = \bar{\alpha}_o \pmod{D}$ in P5a.

The third constraint says that we are looking for a solution cheaper than the best solution yet. Practically, since v has a cost of zero, executing P6 without the constraint $v \in N$ would be meaningless since optimal solutions would be found in large numbers with v fractional.

It seems that this problem would be best avoided by solving:

$$\text{Min} \quad \sum_{i=1}^{t} \bar{c}_i \, u_i$$

$$\text{such that} \quad C \, u \leq B^{-1} \, 1$$

P6a $$\sum_{i=1}^{t} \bar{\alpha}_i u_i = \bar{\alpha}_o + D.k$$

$$u \in \bar{B} \text{ and } \sum_{i=1}^{t} \bar{c}_i \, u_i < c^*$$

$$\text{for } k = 0, 1, 2, \ldots$$

If no feasible solution is found to any of these problems, the best integer solution yet found is proved optimal. Otherwise, the optimal solution to P6a would provide a better integer solution: $(x_B = B^{-1}1 - Cu,\ x_R = \{u\})$.

CHAPTER IV

SOLUTION TECHNIQUES & COMPUTATIONAL EXPERIENCE

4.1 Introduction

This chapter will show the different approaches developed here in the search for the optimal integer solution to the rotation selection (set covering) problem. All of them are based on the group theoretic method but correspond to solving problems of different sizes or with different types of cost vectors.

4.2 Solution Techniques for the Integer Optimum

Based on the group theoretic approach described in Chapter III, five approaches to finding the integer optimum have been found in this research, each of which applies to a type of rotation selection problem.

4.2.1 Group Theoretic Method Program

When $\bar{D}$ is a prime or degenerate (cyclic group), if the matrix size is moderate, one should use GTMP, the group theoretic method program. This program has been written and tested for this thesis and provided most of the computational experience. A first part of the program reads all the $B^{-1}a_j$ columns for non-basic columns. A second part tries to find the cheapest solution to P5 for which $\sum_{k=1}^{n} x_k = 1$, ie. for which only one non-basic column from the LP (such that $\bar{\alpha}_j = \bar{\alpha}_o$) provides a feasible integer solution to P1. The cost c* of this solution is an upper bound on the cost of the optimal integer solution. A third part tries to find whether there are cheaper solutions to P5 for which $\sum_{k=1}^{n} x_k = 2$, ie. such that pushing two columns which were non-basic after the LP at an activity of one will help obtain a better integer solution. Theoretically, combinations of 3 and more columns should also be considered. Practically, optimal solutions were always reached up to now in that manner. In case optimality had to be proved, problem P6a would have to be solved. Figure 4-1 flow-charts this procedure.

4.2.2 Binary Linear Inspection Program

For a prime or degenerate $\bar{D}$, BLIP, the binary linear

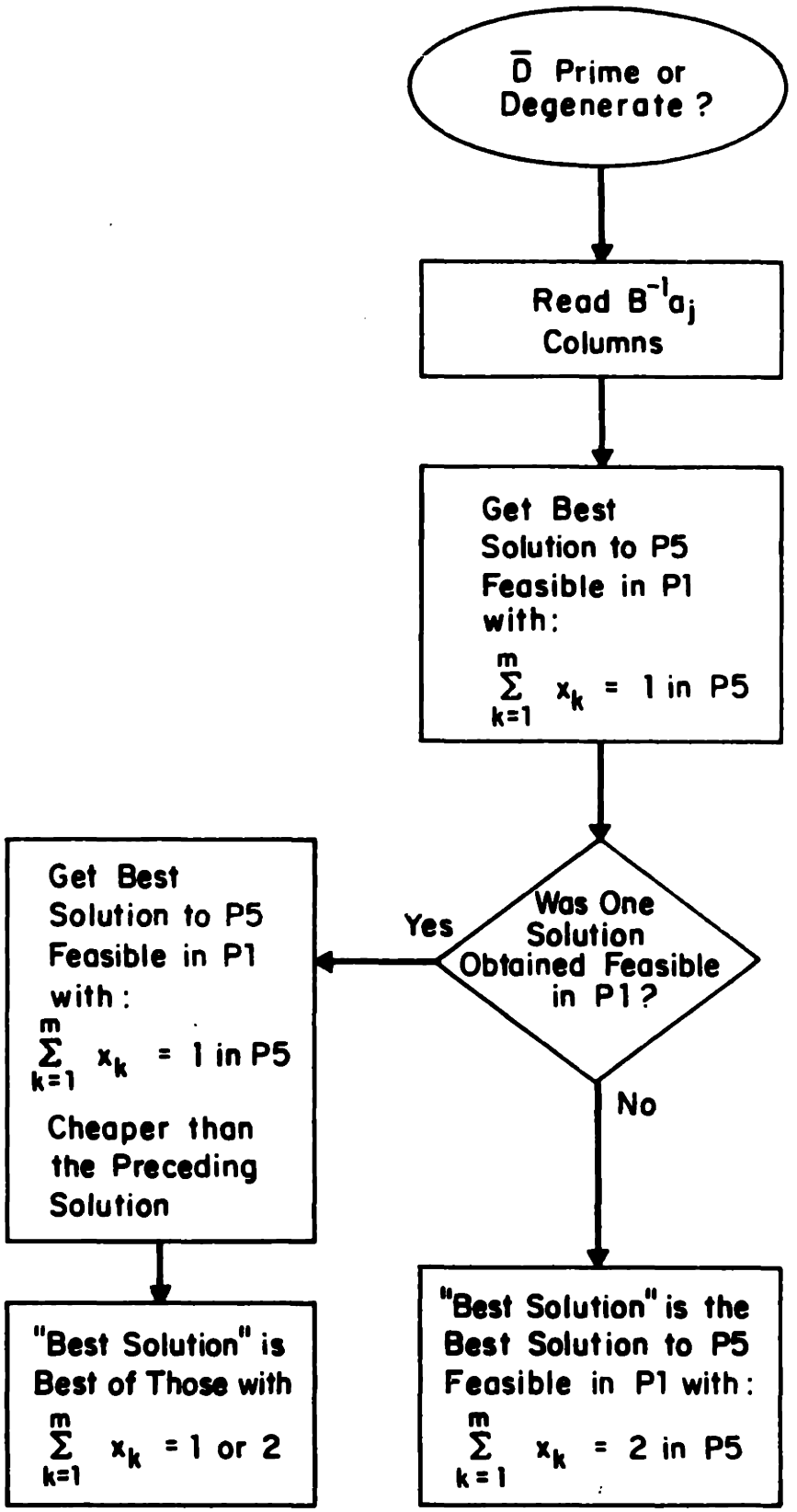


Figure 4.1 GROUP THEORETIC METHOD PROGRAM

inspection program, provides the optimal solution to problems of moderate and large sizes. It is expected to perform better than GTMP for problems of size greater than 100 x 600 (or equivalent sizes). There are three parts in this program: a first part looks for the names of columns for which $\bar{\alpha}_j = \bar{\alpha}_o$. The second reads their $B^{-1}a_j$ vectors and tries to find the cheapest solution to P5 for which $\sum_{k=1}^{n} x_k = 1$, ie. for which setting only one non-basic column to an activity of 1 provides a feasible integer solution. The cost c* of this solution is an upper bound on the cost of the optimal solution. The third part of the program reads the $B^{-1}a_j$ vectors of all columns for which the reduced cost $\bar{c}_j < c^*$ and finds the optimal solution. Through this approach, the original problem is solved while $B^{-1}a_j$ may have to be completely generated for only as few as 1% of the columns. This program has not yet been written; the method has however been used by submitting three different runs and operating the changes manually. But it is then identical to the following method.

4.2.3 Semi-automatic Inspection

If $\bar{D}$ is a small non-prime number, e.g. the product of two prime numbers, semi-automatic inspection should be used. The first two parts are as in BLIP, the third part being solved by

visual inspection rather than by a program.

4.2.4 Automatic Branching for Unit Costs

When the cost vector is unity, determinants are likely to be larger, other things being equal. This is true because all columns are more or less eligible for the continuous optimum, since there is little differentiation between them. If $\bar{D}$ is such that the techniques of 4.2.1 through 4.2.3 do not apply, a new approach must be taken.

An "automatic branching" technique more adequate than "branch and bound" techniques was developed where a non-integer variable is set to one and the problem re-solved until the next integer value is reached by the objective. If the continuous optimum costs 25.71, there is no point developing the whole branch and bound tree down to z = 26 if a solution with that cost may be found by direct branching. The flow chart in figure 4.2 describes the technique we developed for this type of problem.

This technique was used to solve problem BEA-II, 98 x 1652. The branching is described in figure 4-3; z represents the value of the objective function. The problem was solved by direct branching until z reached a value of 26. It was then found that for two nodes, $\bar{D}$ was a product to two prime numbers.

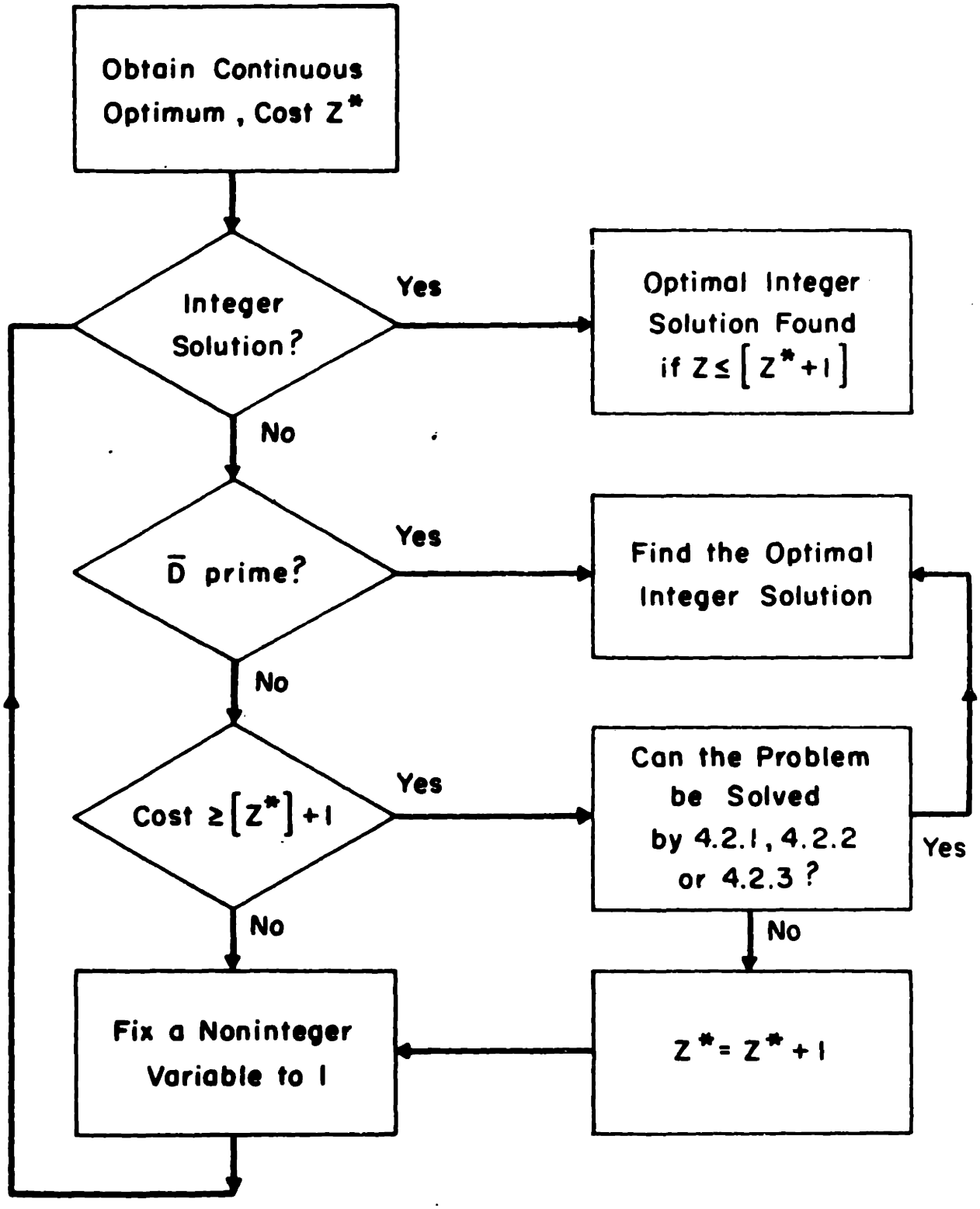


Figure 4.2 AUTOMATIC BRANCHING TECHNIQUE

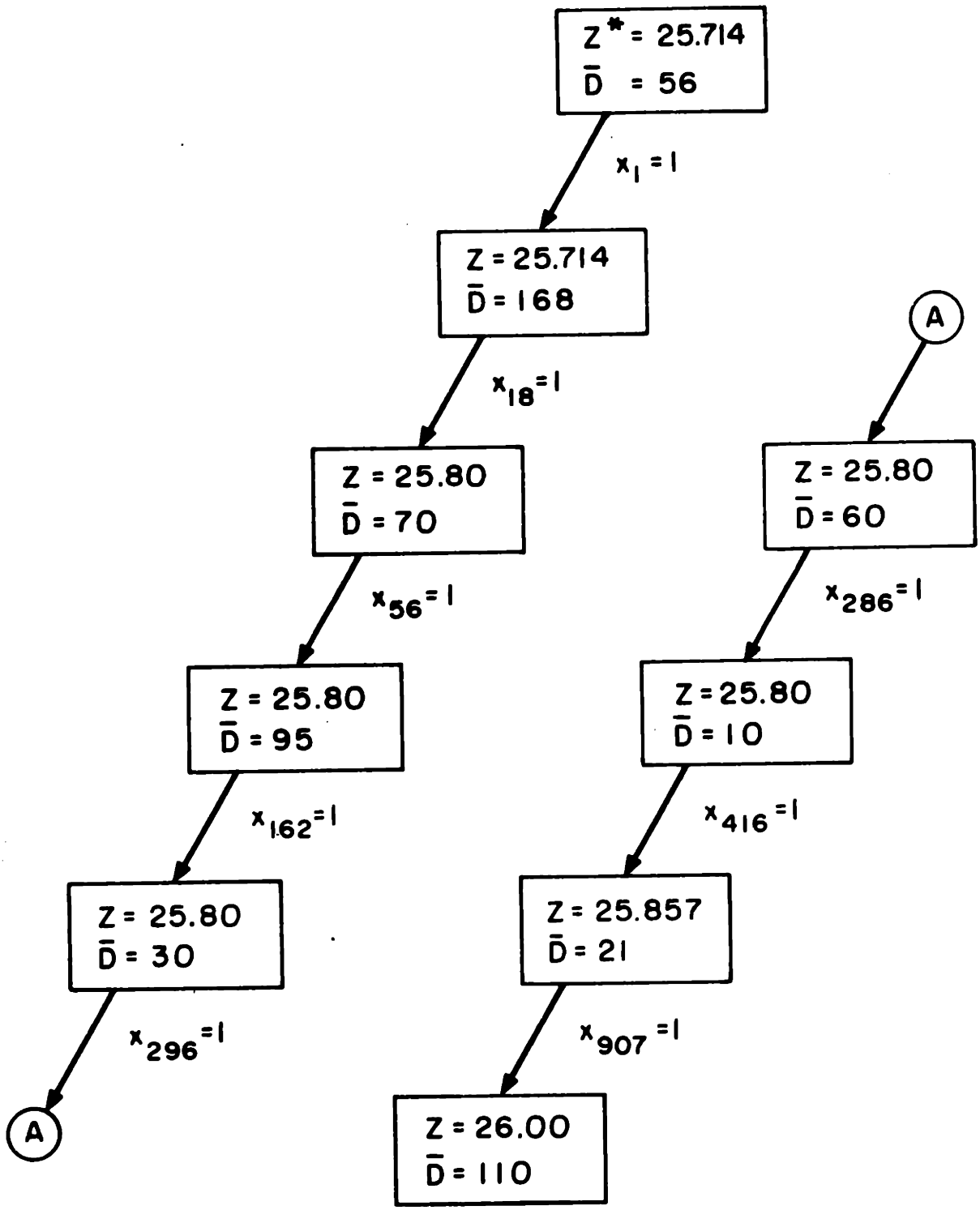


Figure 4.3 DIRECT BRANCHING TREE FOR BEA-II

The last one was selected and the semi-automatic inspection technique was used. This provided several optimal integer solutions at a cost of $z = 26.00$ crews.

If regular branch and bound has been used, it would have taken much longer: the objective value did not increase very fast as one branched to the left ($x_i = 1$); it would have increased even slower towards 26 on branches to the right ($x_i = 0$). The rationale for this behavior is simple: setting $x_i = 1$ reduces the problem greatly since several row constraints are satisfied in one shot. With $x_i = 0$, one of several hundred columns with similar eligibility (same cost of unity) is deleted but the problem is hardly modified.

Judging by the values obtained for $\bar{D}$ in the branching tree (see figure 4.3), the analyst who did go through step ($z^* = z^* + 1$) of the flow chart may want to search other branches if he thinks he has a chance to find a cheaper solution this way. For example, if several relatively small values were found for $\bar{D}$ before that step, there is a better chance of finding such a cheaper solution.

4.2.5 Branch and Bound with Non-unit Costs

In this case regular branch and bound may be used if the techniques described from 4.2.1 to 4.2.3 were not applicable.

The process can however be sped up considerably. If one projects the continuous optimum on the line of values of any variable x_i, one obtains a convex, as shown in figure 4.4.a.

Parametric programming provides us with the values of z_o and z_1, lower bounds on the optimum objective value for $x_i = 0$ and $x_i = 1$. If a feasible solution has been found with a cost of z ($z_o > z > z_1$), it is clear that a better solution may only be found for $x_i = 1$. In the same manner, if a variable has an activity of 0 or 1 at the continuous optimum, there are lower bounds on the minimum costs which would incur if it were pushed to 1 or 0.

With a little experience, one may, given the cost of the continuous optimum, estimate how close from it the optimal integer is most likely to be. This cost is set as the bounding cost Z, unless a better integer solution has already been found. Then, for each variable, after having obtained the continuous optimum, z_o and z_1 are estimated. If only one is less than Z, say z_o (z_1), the variable is fixed to 0 (1). If both are greater than Z, Z was chosen too small; pick up a larger value and re-start. At each terminal node, where the techniques described in sections 4.2.1 through 4.2.3 are applicable, they must be used to provide the best integer solution at that node.

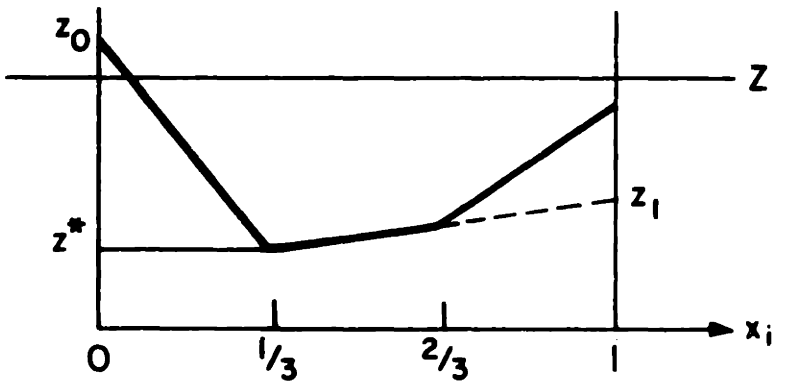


Figure 4.4a

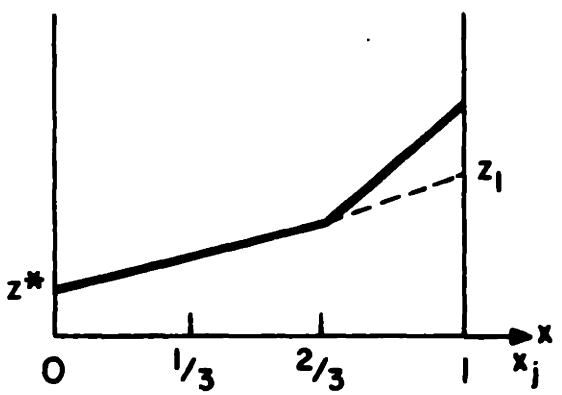


Figure 4.4b

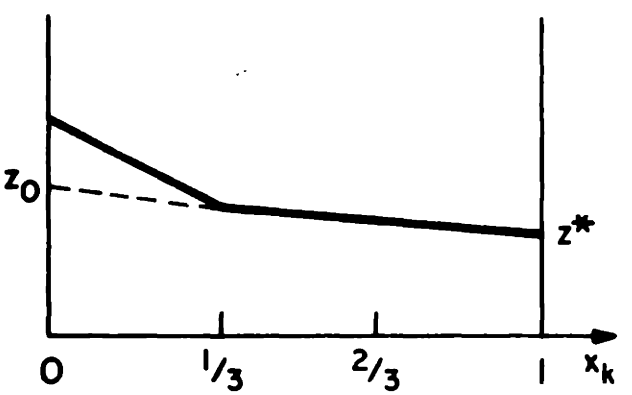


Figure 4.4c

Figure 4.4 BRANCH AND BOUND BOUNDING

4.3 Types of Rotation Selection Problems

All the problems experimented with to date in this research were provided by airlines; it was felt that generating random matrices of zeroes and ones and cost vectors, even with the right density, would not be a good solution. There are irregularities in the matrix structures of the problem sent by the different carriers which are difficult to simulate in a random generator. For example, some typical flight segments (e.g. Boston-New York, 8 a.m.) will be included in many more rotations than others, because there is a dense traffic in both cities.

Figure 4-5 presents some of the experience the airlines have with the continuous LP solution to the crew scheduling problem. It shows the span of application this group theoretic approach has - it must be clear, after having read the third chapter that the group approach is most useful when $\bar{D}$ is a prime number or small.

On the basis of several hundred problems, it can be said that from 50% to 90% of all problems will have an integer solution at the LP. From those which are not integer, $\bar{D}$ is expected to be small or prime for almost all problems. The group theoretic approach described in section 4-2 has therefore a great potential.

Figure 4.5 ROTATION SELECTION PROBLEM TYPES OF AIRLINES

Airline	Constraint	Cost Vector	Sample Problem Solution by Airline				
			Size	Computer Used	Optimum Found	Time	Method Type
Air Canada	Ax = 1	Non-unit	142 x 650	IBM 360/50	No	30 min.	Implicit Enumeration
Air France	Ax = 1	Non-unit	67 x 536*	IBM 360/75	No z = 6106	1.2 min.	LP and Branching
American Airlines	Ax ≥ 1	Non-unit	104 x 132*	IBM 360/65	Yes	250sec.	IBM SCA-I
BEA	Ax ≥ 1	Unit	98 x 1652*	Univac 494	No z = 28	12.24 min.	House, Nelson & Rado (ref. 13)
KLM	Ax = 1	Non-unit	20 x 70	IBM 360/40	Yes	15 min.	Branch and Bound
Luftansa	Ax = 1	Unit	120 x 400	Univac 1107	No	70 sec.	Branch and Bound
SAS	≥ for base to base fleet = elsewhere	Non-unit	93 x 2300	Univac 494	No	12 min.	Cutting Plan and LP rounding
Swissair	Ax = 1	Unit	59 x 543	IBM 360/40	Yes	50 min.	Implicit Enumeration
United Airlines	Ax = 1	Non-unit	117 x 4845*	CDC 3600	Yes	55 min.	Cutting Plane

* Problem was also solved in GTMP and/or BLIP approach. See Figure 4.6

4.4 Summary of Computational Experience

The IBM Mathematical Programming System MPS was used to solve the LP and all other programs in GTMP were written in FORTRAN IV.

Figure 4.6 describes the computational experience gathered to date using the techniques explained in sections 4.2.1 through 4.2.5.

There is no doubt that, when BLIP is programmed, problems of 20000 columns will be acceptable and their optimal solutions will be obtained.

From the present computational experience, it appears that the time needed to reach the optimal integer solution from the LP optimum can be expected to be shorter than the time it took to attain the LP optimum. Not enough problems have been solved yet to allow us to plot a graph of solution time vs. size but, on the basis of preceding experience and, considering the way BLIP operates, it seems valid to say:

Total Solution Time $<$ 2*(LP Solution Time) or

(Time From LP Optimum to Integer Optimum) $<$ Time to LP Optimum

Most methods currently available to solve the set covering problem are unable to solve problems of moderate to large size

Figure 4.6 COMPUTATIONAL EXPERIENCE

COMPUTER: M.I.T. IBM 360/65

			LP SOLUTION			INTEGER OPTIMAL SOLUTION				SEMI-AUTOMATIC INSPECTION
Origin	Size	Type	$Z^* =$	LP Time	$\bar{D}$	Z =	Time After LP	Total Time	Technique Used	MPS Time
AA - I	104 x 132	Ax ≥ 1	8817.5	15.6 sec.	2	8820	9.6 sec.	25.2 sec.	GTMP 4.2.1	20.4 sec.
AA - II	104 x 236	Ax = 1	14000.7	31.8 sec.	7	14145	22.8 sec.	54.6 sec.	GTMP 4.2.1	39.8 sec.
AF	67 x 536	Ax = 1	6041.5	28.2 sec.	2	6049	40.8 sec.	69 sec.	GTMP 4.2.1	50 sec.
UAL	117 x 4845	Ax = 1	217350.6	20 min.	24	217687				105 sec. (After LP)
BEA - I	84 x 854	Ax ≥ 1	22.70	1.57 min.	60	23.0	2.06 min.	4.63 min.	4.2.4	
BEA - II	98 x 1652	Ax ≥ 1	25.714	3.29 min.	56	26.0	6.55 min.	9.84 min.	4.2.4	

(e.g. over 1500 columns) or expect the solution time to increase much faster with size than the LP solution time.

4.5 Possible Improvements

In fact, good as the computation times are, they could be improved by:

- changing MPS to make it more flexible
- or using another LP code offering more flexibility

Assuming one had to use MPS, here are several things that could be changed to permit a much faster execution. The two most important ones are described in the following paragraphs.

4.5.1 MPS Operating System

MPS may not be called as a subprogram by a FORTRAN program. This limitation does not allow us to easily manage core and link from FORTRAN to MPS and vice-versa very efficiently. For example, it would involve relatively complex programming to give the calling FORTRAN program a COMMON area residing permanently in core. This is a large drawback if one wants to write a branch and bound code and needs many transfers between FORTRAN and MPS.

4.5.2 Alphameric Names

All rows and columns are identified by eight-character alphameric names. As a consequence, each element of $B^{-1}a_j$ is given with a row and a column name. To compare columns, test or add them, one must test down two lists of alphameric 8-character names and watch for the appearance of the same names in both. This is a very heavy procedure and speed could be vastly increased if one could refer to a row by its row number rather than its name. Giving a row name to each element when reading it in would not be faster in the long run, since it would involve transforming many elements that are not used in the program; the improvement would involve flagging rows by numbers and/or names rather than just names within the MPS-FORTRAN communications system.

4.6 Conclusion

In judging this group theoretic approach, one must remember that the part of the optimization following the LP could be made reasonably faster by using a more flexible LP code or re-programming some routines in MPS; for example by defining an argument for TRANCOL which would limit the operation to columns for which the revised cost is less than the argument.

However, even in this imperfect form, this approach is believed to be highly competitive with other programs solving the same problem. Other programs tried on some of these example problems had execution times larger by orders of magnitude.

CONCLUSION

An efficient solution method for the set covering problem appearing in Crew Scheduling has been presented. There are two ways this research could be extended.

First, in the area of Crew Scheduling, an integrated approach should be devised, where, given a timetable, one could obtain the monthly bids by just calling a program. This program would generate rotations, select them and build the bids without requiring, but allowing, a human interface during the process.

Secondly, in terms of set covering, BLIP (see section 4.2.2) must be programmed and tested against large problems to show how fast very large problems (e.g. 20,000 columns) may be solved.

APPENDIX A

TABLE OF FIGURES

BIBLIOGRAPHY

1. Agard, J., Arabeyre, J.P., and Vautier, J., "Generation Automatique de rotations d'equipages", Revue d'Informatique et de Recherche Operationnelle, No. 6, 1967.

2. Arabeyre, J.P. (Air France), Fearnley, J. (Air Canada) Steiger, F. (Swissair), and Teather, W. (BEA), "The Airline Crew Scheduling Problem: A Survey", Transportation Science 3, No. 2, May, 1969.

3. Arabeyre, J.P., "Combinatorial Algorithms for Crew Scheduling", AGIFORS Paper, Amsterdam, 1967.

4. Balas, E., "An Additive Algorithm for Solving Linear Programs with 0-1 Variables", Operations Research 13, No. 4, July-August, 1965.

5. Balinski, M.L., "Integer Programming: Methods, Uses, Computation", Management Science 12, No. 3, November, 1965.

6. Bennett, B.T., Potts, R.B., "Rotating Roster for a Transit System",Transportation Science 2,No. 1,February,1968.

7. Fearnley, J. (Air Canada), "Crew Scheduling Developments in Air Canada", AGIFORS paper, Killarney, Ireland, 1966.

8. Geoffrion, A.M., "Integer Programming by Implicit Enumeration and Balas' Method", SIAM Rev. 9, 178-190 (1967).

9. Geoffrion, A.M., "An Improved Implicit Enumeration Approach for Integer Progamming", The Rand Corporation, RM-5644-PR, June, 1968.

10. Gomory, R.E., "An Algorithm for Integer Solutions to Linear Programs", pp. 269-302 in Recent Advances in Mathematical Programming, Graves, R.L. and Wolfe, P. (Eds.) McGraw-Hill, 1963.

11. Gomory, R.E., "On the Relation Between Integer and Non-Integer Solutions to Linear Programs, Proc. Nat. Acad. Sci. 53, pp. 260-295, 1965.

12. Healy, W.C., Jr., "Multiple Choice Programming", Operations Research 12, pp. 122-138, 1964.

13. House, R.W., Nelson, L.D. and Rado, T., "Computer Studies of a Certain Class of Linear Integer Problems", Recent Advances in Optimization Techniques, Lavi, A., and Vogl, T., (Eds.), Wiley, 1966.

14. Kolner, T. K. (United Air Lines), "Some Highlights of a Scheduling Matrix Generator System", AGIFORS paper, Killarney, Ireland, 1966.

15. Land, A. H., and Doig, A. G., "An Automatic Method of Solving Discrete Linear Programming Problems", Econometrica 28, pp. 497-520, 1960.

16. Moreland, J. A., "Scheduling of Airline Flight Crews", S. M. Thesis, Department of Aeronautics, M.I.T., September, 1966.

17. Mostow, G. D., Sampson, J. H., and Meyer, J. P., Fundamental Structures of Algebra, Mc-Graw-Hill, 1963.

18. Pierce, "Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems", IBM Cambridge Scientific Center Report, Cambridge, Massachusetts, July, 1966.

19. Shapiro, J. F.,"Dynamic Programming Algorithms for the Integer Programming Problem-I: The Integer Programming Problem Viewed as a Knapsack Type Problem", Operations Research 16, pp. 103-121, 1968.

20. Shapiro, J. F., "Group Theoretic Algorithms for the Integer Programming Problem-II: Extension to a General Algorithm", Operations Research 16, No. 5, September, 1968.

21. Steiger, F., and Niederer, M., "An Accelerated Enumeration Method for Combinatorial Programming Problems", paper presented at the IFIP Congress, Edinburgh, 1968.

22. Thiriez, H., "Implicit Enumeration Applied to the Crew Scheduling All-binary Problem", Department of Aeronautics, M.I.T., 1968

23. Thiriez, H., "Methods for the Generation of All Circuits or Cycles in a Graph", M.I.T. Memo FTL-M68-10, September, 1968

24. Wren, A., Applications of Computers to Transport Scheduling in the United Kingdom, College of Engineering, West Virginia University, E.C.S. Bulletin 91, March, 1969.