SCHEDULING OF MULTIPLE VEHICLE TYPES: THE ALLOCATION OF LOCOMOTIVES TO TRAINS

A thesis presented for the degree
of Doctor of Philosophy

by

Karen Maria Reddington

Department of Operational Research
London School of Economics and Political Science

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

Each year railway network organisations concern themselves with the complete specification of a commercial timetable. Once a timetable is established for a region and a limited number of locomotives of different types are allocated to the region, the stock diagramming exercise commences. The problem is to schedule locomotives to work the timetabled trains such that every train is worked on a daily basis and no more than the available locomotives are used. The stock diagramming problem can therefore be characterized as a scheduling problem. The objective is to minimize the light-run time incurred by locomotives having to perform unproductive runs, i.e. not working a train, between stations in the region.

The proposed solution procedure uses a column generation technique. The problem is decomposed into two parts, the master problem and the subproblems. The master problem, formulated as a set-partitioning problem, selects a subset of locomotive schedules from a set of known feasible schedules. The subproblems generate the feasible schedules. The linear programming relaxation of the master problem is solved and a constraint branching technique used to locate an integer solution.

Due to the size of the problems encountered in practice, the solution methodology is essentially a heuristic. At the branch and bound stage, an optimal solution is not pursued. Instead, a 'good' integer solution, strongly based on the optimal solution to the linear programming relaxation of the set-partitioning problem, is found.

The computational results presented refer to problems drawn from real data.

ACKNOWLEDGEMENTS

My first word of thanks must go to my supervisor Dr. S. Powell whose door was always open, and whose advice and assistance was always delivered with patience and a friendly smile.

I should also like to thank all the staff of the Operational Research Department who helped me out along the way, especially Dr. G. Appa and Dr. D. Connolly. Thanks to my fellow students for their friendship and for the good memories they have given me.

Thanks are due to British Rail for providing me with the problem as well as real-world data sets. I would also like to thank Dr. M. Wright of Lancaster University for the additional data sets. I am grateful to the SERC for funding this course of study. Also, I owe a great debt of gratitude to Dr. A. Maxwell who lent me his computer for the duration of my work, and in doing so made life a good deal easier for me.

To Cath, you are a wonderful person to live with and laugh with. A special thank you to my best friend Angus for supporting me both financially and morally, but most of all for just putting up with me.

In conclusion, this thesis is dedicated to my parents who, as always, gave me their unquestioning love and support throughout.

LIST OF CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 Outline of the Problem

An effective distribution or transportation system is based on decision problems at all levels of operations, from decisions about the levels of capital investment to the day-to-day planning of the movements of vehicles and crews. Given the growth of such sectors the potential for large scale savings at every level of this planning process is widely recognized. An organisation such as British Rail concerns itself every year with these decisions when specifying the commercial timetable. The final timetable results from the processes, in order of development, of: train timing; stock diagramming; crew diagramming; and, local area scrutiny. It is the problem of stock diagramming which is considered in this thesis.

The output of the train timing exercise is a train timetable specifying trains which must travel at fixed times between fixed locations. The stock diagramming problem involves the scheduling of a heterogeneous set of locomotives to work the trains. The timetable must be covered at minimum cost using no more than the available locomotives. The costs involved are the costs of locomotives making unproductive journeys from the end location of one train to the start location of another. As the locomotives are of different types, it must also be ensured that the locomotive assigned to a train is compatible with that train. The information on the locomotive types permitted to work a train is supplied in the timetable. The output from the stock diagramming exercise is a set of schedules. Each schedule describes the order in which the trains in the schedule are to be worked and has an associated locomotive type compatible with all trains in the schedule. The purpose of this thesis is to devise a method for finding optimal or near-optimal solutions to British Rail's stock diagramming problem.

1.2 Structure of the Thesis

Chapter two gives a detailed description of the stock diagramming problem.

Chapter three reviews solution approaches to the stock diagramming problem. This problem is a member of a broader class of routing and scheduling problems. Therefore, the discussion in this chapter extends to a consideration of solution methods for such problems.

Chapter four discusses the initial solution methods attempted in this thesis. The chapter concludes by proposing that a mathematical programming approach which uses a column generation technique is adopted.

Chapter five introduces the theory of the column generation technique and in chapters six, seven and eight a description of how the column generation technique is applied to the stock diagramming problem is given. Chapters six and seven describe the method used to obtain optimal solutions to the linear programming formulation of the stock diagramming problem. Chapter eight describes a branching strategy used to resolve any fractionalities occurring in these solutions.

Chapter nine summarizes the final method proposed and presents the results of applying the method to a number of real world stock diagramming problems.

In chapter ten some concluding remarks are made and suggestions for further research into this problem are given.

CHAPTER TWO

PROBLEM OUTLINE

2.1 Problem Description

Before outlining the stock diagramming problem it is necessary to introduce some basic terminology. The output from the train timing exercise is a train timetable which provides the starting point for the stock diagramming process. It is usually the case that within British Rail there is a weekday timetable for Tuesday through to Friday and separate timetables for Saturday, Sunday and Monday. However, in this thesis, it is assumed that the same trains are timetabled every day and the solutions found for the stock diagramming problem represent schedules which can be repeated day after day. In practice this is how British Rail proceed with the problem. Once a set of repeatable schedules have been found they then make adjustments to tackle the changes which occur at the weekends. A typical train timetable covers trains which have a start time within a one-day period. Each line of the timetable represents a job or task which has to be executed on a daily basis. The tasks are referred to as 'trains'. Each train has associated characteristics as follows: a start-time; an end-time; a start location; an end location; and a set of allowed locomotive types. The 'locomotives' are the vehicles used to pull the trains, and only a subset of the types of locomotives available are compatible with a particular train. An example of such a timetable is given in table 2.1.i. The key to the table is:

'HEAD CODE'	-	this is a four character identification code for the train.
'DEPART'	-	this is the departure time for the train.
'FROM'	-	this is a four character code used to indicate the station from which the train departs.
'TO'	-	this is a four character code used to indicate the station at which the train arrives.

'ARRIVE' - this is the arrival time for the train.

'TYPE' - the different locomotive types are distinguished by a number. This column indicates which locomotives are compatible with the train.

HEAD CODE	DEPART	FROM	TO	ARRIVE	TYPE
5T02	00.20	LDSC	LDSC	01.05	1
1D18	02.40	LDSC	HULL	11.56	1 2
1M16	05.48	DONC	SHEM	13.22	1 2
9T03	12.11	HULL	SHEM	08.50	2
2M68	14.40	HULL	HULL	15.15	1 2
3T50	14.56	DONC	LDSC	23.30	2
7F44	16.55	HULL	LDSC	22.20	1
8M62	17.11	SHEM	SHEM	18.00	1 2
3H45	20.32	DONC	SHEM	13.46	2
4D10	23.13	LDSC	HULL	02.25	1

Table 2.1.i

The start time of a train ('DEPART' in the above table) includes an allowance for the preparation of the train, e.g. the time taken to attach the train to the locomotive. Similarly, the end time of a train ('ARRIVAL' in the above table) includes an allowance for the disposal of the train , e.g. detaching the train from the locomotive, and a performance time which caters for probable late running of the train. So, the start time of the train is the time at which a locomotive must be available and the end time is the earliest time at which the locomotive is free.

In addition to the train timetable, information on the types and number of locomotives of each type available is supplied. An example of this data is given at

table 2.1.ii. The key to the table is:

'TYPE' - a number indicating the locomotive type.

'AVAILABLE' - the number of locomotives of the specified type available.

TYPE	AVAILABLE
1	12
2	6

Table 2.1.ii

The only constraints pertaining to the locomotives are the type and number available. There are no conditions requiring a locomotive to return to a pre-specified depot, nor are there any driver related restrictions.

The stock diagramming problem requires that two decisions are made for each train:

1) which type of locomotive should work the train;

and, 2) after this train which train should the chosen locomotive work next?

Consider a simple problem consisting of three trains and two locomotive types, types 1 and 2. In this problem train 1 is compatible with locomotive types 1 and 2, train 2 is compatible with locomotive types 1 and 2, and train 3 is compatible with locomotive type 1 only. A representation of these options for train 1 is given at figure 2.1.i. In this figure the solid lines represent the train arcs and the dashed lines represent the light-run arcs. 'Tn' indicates train n.

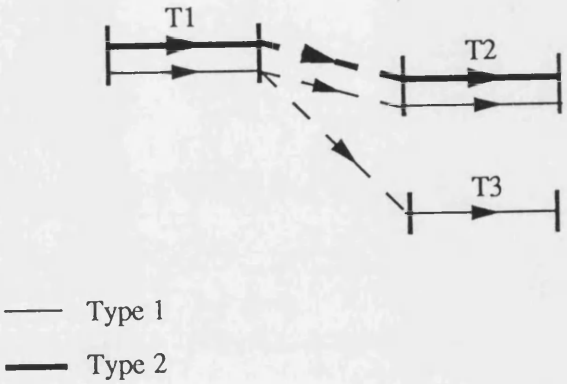


Figure 2.1.i

The allocation of trains to a locomotive type and the decision regarding the order in which the trains should be worked describes schedules for the locomotives. As the timetable has to be worked on a daily basis, these schedules take the form of cycles which can be repeated day after day. All trains in a schedule are compatible with the locomotive type which works the schedule, and over all schedules the total number of locomotive types used should satisfy the availability constraints. A cycle may take one day or a number of days to traverse. In the latter case, the cycle of length n days can be considered as n locomotives working a one-day cycle. To illustrate this, an example of a locomotive schedule is given at figure 2.1.ii. In this figure the solid lines represent the train arcs and the dashed represent the light-run arcs. 'Tn' indicates train n.

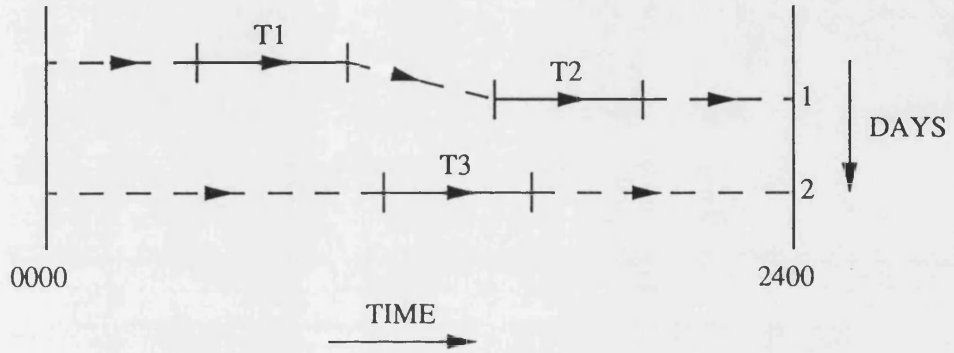


Figure 2.1.ii

In this example two locomotives are required to work the schedule. The periodic nature of such scheduling problems is considered in Carraresi and Gallo [14].

The objective of the stock diagramming problem is to schedule the locomotives to work all trains in a given timetable at minimum cost. A cost is incurred when a locomotive makes an unproductive journey from the end location of one train to the start location of the following train. This cost is known as the 'light-run' cost and is expressed in terms of a journey time which is independent of the type of locomotive used.

2.2 Problem Formulation

Armed with a set of constraints and a declared objective it is now possible to formulate the problem mathematically.

Let the set of trains be $\tau=\{i \mid i \text{ is the } i^{th} \text{ train in the timetable}\}$. Let the set of locomotives be $\Lambda=\{k \mid k \text{ is the number of a locomotive type}\}$. Then, define P(k) to be the set where $P(k)=\{i \mid \text{train i is compatible with locomotive type k}\}$

SD

$$\text{MIN} \quad \Sigma_{i\in\tau}\Sigma_{j\in\tau}\Sigma_{k\in\Lambda}\, c_{ij}\, x_{ij}^{k}$$

s.t.

$$\Sigma_{j\in P(k)}\, x_{ij}^{k} = y_i^k \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD1}$$

$$\Sigma_{j\in P(k)}\, x_{ji}^{k} = y_i^k \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD2}$$

$$\Sigma_{k\in\Lambda}\, y_i^k = 1 \qquad \forall\, i\in\tau \qquad \textbf{SD3}$$

$$\Sigma_{i\in\tau}\Sigma_{j\in\tau}\,(m_i\, y_i^k + M_{ij}\, x_{ij}^k) \le U_k \qquad \forall\, k\in\Lambda \qquad \textbf{SD4}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, i,j\in P(k),\, k\in\Lambda \qquad \textbf{SD5}$$

$$y_i^k \in \{0,1\} \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD6}$$

where the zero-one variables are:

x_{ij}^k - 1 if train j follows train i on locomotive type k,
- 0 otherwise.

y_i^k - 1 if train i is worked by locomotive type k,
- 0 otherwise.

The data for the problem are:

c_{ij} - light-run time from end location i to start location j;

m_i - the number of times a locomotive working train i crosses midnight;

M_{ij} - the number of times a locomotive performing a light-run from end location i to start location j crosses midnight;

U_k - the total number of locomotives of type k available to work the timetable.

The variables and data are related to the options shown at figure 2.1.i in Figure 2.1.iii. In this figure the solid lines represent the train arcs and the dashed lines represent the light-run arcs.

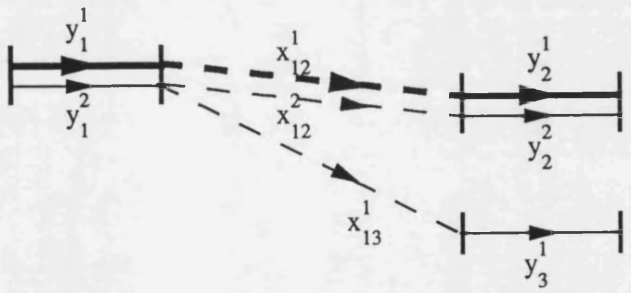


Figure 2.1.iii

Constraint set SD1 ensures that the total number of light-runs from end location i to any start location j on a locomotive type k is equal to the number of times locomotive type k works train i. Constraint set SD2 ensures that the total number of light-runs from any end location j on locomotive type k to the start location of i is equal to the number of times locomotive type k works train i. Constraint set SD3 ensures that train i is only worked once by one type of locomotive k, as required. Constraint set SD4 ensures that no more than the available number of locomotives are used.

Constraints SD1 and SD2 ensure a conservation of flow and constraint SD3 ensures that a train is worked exactly once. These constraints are straightforward, but constraint SD4 warrants some further explanation. As the schedules take the form of cycles the number of locomotives required to work a timetable of a one-day period can be calculated by counting the number of times a locomotive crosses midnight. Referring back to the example in figure 2.1.ii, $m_1 = 0$, $m_2 = 0$, $m_3 = 0$, $M_{12} = 0$, $M_{23} = 1$ and $M_{31} = 1$, and so $(m_1+m_2+m_3+M_{12}+M_{23}+M_{31})=2$, showing that two locomotives are required to work the schedule shown in this example. Constraint SD4 checks that the number of locomotives of each type used in a solution does not exceed the number available by counting the number of times the schedules worked by a particular locomotive type cross midnight.

2.3 Extensions to the basic Stock Diagramming Problem

As with most practical problems the statement of the problem can over-simplify the reality of the situation. Actually, British Rail divides the total area in which trains have start and end locations into regions. The effect of this is that, although the problem is solved on a region-wise basis, trains will cross-over from one region to another. The details of this facet of the problem are discussed later in chapter four.

CHAPTER THREE

REVIEW OF ROUTING AND SCHEDULING PROBLEMS

3.1 Placing the Stock Diagramming Problem in Context

Stock diagramming is a specific instance of a class of problems which occur practically and theoretically. In order to appreciate the driving characteristics of the problem and the solution methodologies which may be applicable, it is worth introducing and discussing the broad-based class of problems known as "routing and scheduling" problems.

In common, routing and scheduling problems have the same basic output: for each vehicle or driver providing a service to a set of entities a route or schedule is given. A route specifies the order in which the entities are to be serviced and a schedule specifies the times at which the entities are serviced. Bodin and Golden [9] attempt to define routing and scheduling problems. The definition they give for a routing problem is: "If the entities to be serviced have no temporal restrictions and there are no precedence relations among these entities then we have a *routing* problem." A well-known example of a routing problem is the Travelling Salesman problem. They define a scheduling problem as follows: "If each entity has a definitive service time, then a *scheduling* problem results." In accordance with this definition, the stock diagramming problem is a member of the class of scheduling problems. For completeness it should be noted that a further class of combined routing and scheduling problems exists. Typically such problems arise when time windows and/or precedence relationships exist so that both routing and scheduling problems need to be considered. Further to these definitions Golden *et al.* [38] outline the main characteristics of routing and scheduling problems which can be used to identify the nature of the problem being tackled. This taxonomy is given in Figure 3.1.i.

CHARACTERISTICS	POSSIBLE OPTIONS
1. Size of available fleet	a. one vehicle b. multiple vehicle
2. Type of vehicle fleet	a. homogeneous (only one vehicle type) b. heterogeneous (multiple vehicle types) c. special vehicle types (compartmentalized,etc.)
3. Housing of vehicles	a. single depot (domicile) b. multiple depot
4. Nature of demands	a. deterministic demands b. stochastic demands c. partial satisfaction of demand allowed
5. Location of demands	a. at nodes (not necessarily all) b. on arcs (not necessarily all) c. mixed
6. Underlying network	a. undirected b. directed c. mixed d. euclidean
7. Vehicle capacity restrictions	a. imposed (all the same) b. imposed (different vehicle capacities) c. not imposed (unlimited capacity)
8. Maximum route times	a. imposed (same for all routes) b. imposed (different for different routes) c. not imposed
9. Operations	a. pick-ups b. drop-offs (deliveries) only c. mixed (pick-ups and deliveries) d. split deliveries (allowed or disallowed)
10. Costs	a. variable or routing costs b. fixed operating or vehicle acquisition costs c. common carrier costs (for unserviced demands)
11. Objectives	a. minimize total routing costs b. minimize sum of fixed and/or variable costs c. minimize number of vehicles required d. maximize utility function based on service or convenience e. maximize utility function based on customer priorities

Figure 3.1.i

Different combinations of these characteristics can give rise to a plethora of routing and scheduling problems, but how does the stock diagramming problem fit into this picture? The stock diagramming problem is a multiple vehicle problem with a heterogeneous fleet of locomotives and so categories 1.b. and 2.b. apply. The locomotives are not considered to be housed and so category 3 does not apply. Demand is deterministic, as given by the timetable, and is located on the arcs so categories 4.a and 5.b apply. The underlying network is directed and category 6.b applies. There are no vehicle capacity restrictions, only type restrictions, and no maximum route times, so categories 7 and 8 do not apply. The operations are mixed as the locomotive picks up and delivers a train at pre-specified times and locations so category 9.c applies. The costs are routing costs and the objective is to minimize the total cost, so categories 10.a and 11.b apply.

Any slight variation in the characteristics between one problem and another can significantly alter the nature and complexity of a problem, but this does not preclude a comparison of available solution strategies for routing and scheduling problems.

3.2 Solution Techniques

3.2.1 Heuristic Approaches

Consider the formulation SD given in chapter two. For a problem consisting of 10 trains and 2 locomotives types the formulation SD generates up to 200 x_{ij}^k variables. This size of problem is manageable, but in practice a problem may involve 500 trains and 10 locomotive types and the number of x_{ij}^k variables could rise to a maximum of 2.5 million. This prohibitively large number of variables suggests that exact solution techniques are impractical. For this reason heuristic approaches have tended to dominate the solution techniques used to solve vehicle routing and

scheduling problems. In many instances they attack these combinatorially complex and often large problems by breaking up the problem into manageable components. This is the principle behind British Rail's solution technique.

British Rail currently use a heuristic procedure to solve the stock diagramming problem. The heuristic produces good solutions which are then improved upon by the intervention of a British Rail analyst who, using intuition and experience, inserts or deletes specific connections with a view to improving the solution. The heuristic is based on a scheme which manages the size and complexity of the problem by considering each locomotive type k in turn. The partial problem associated with locomotive type k only considers the subset of trains which is compatible with locomotive type k. Before outlining the algorithm used it is useful to introduce the theory underpinning the method.

From the formulation SD, if only one locomotive type is considered the problem reduces to an assignment problem with the additional constraint that there is a limit on the number of locomotives available. Suppose that k is the locomotive type being considered, then the following formulation arises:

SD_k

$$\text{MIN } \Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\, c_{ij}\, x_{ij}^{k}$$

s.t.

$$\Sigma_{j\in P(k)}\, x_{ij}^{k}=1 \qquad \forall\, i\in P(k) \qquad \mathbf{SD_k1}$$

$$\Sigma_{j\in P(k)}\, x_{ji}^{k}=1 \qquad \forall\, i\in P(k) \qquad \mathbf{SD_k2}$$

$$\Sigma_{i\in}\Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\, (m_i+M_{ij})x_{ij}^{k} \le U_k \qquad \mathbf{SD_k3}$$

$$x_{ij}^{k} \in \{0,1\} \qquad \forall\, i,j\in P(k) \qquad \mathbf{SD_k4}$$

The zero-one variables are:

x_{ij}^k - 1 if train j follows train i on locomotive type k;
- 0 otherwise.

The data elements are:

c_{ij} - the light-run time from end location i to start location j;
m_i - the number of times a locomotive working train i crosses midnight;
M_{ij} - the number of times a locomotive performing a light-run from end location i to start location j crosses midnight;
U_k - the total number of locomotives type k available.

Now consider the Lagrangean relaxation which arises from relaxing constraint SD_k3.

LSD_k

$$\text{MIN } \Sigma_{i\in P(k)}\Sigma_{j\in P(k)} (c_{ij}+\mu m_i+\mu M_{ij})x_{ij}^k - \mu U_k$$

s.t.

$$\Sigma_{j\in P(k)} x_{ij}^k=1 \qquad \forall\, i\in P(k) \quad \mathbf{LSD_k1}$$

$$\Sigma_{j\in P(k)} x_{ji}^k=1 \qquad \forall\, i\in P(k) \quad \mathbf{LSD_k2}$$

$$x_{ij}^k \geq 0 \qquad \forall\, i,j\in P(k) \quad \mathbf{LSD_k3}$$

where the variables and data are as defined in SD_k and μ is the Lagrangean multiplier. For a given value of μ LSD_k is an assignment problem solvable in polynomial time. The variables x_{ij}^k only take zero-one values in a solution to LSD_k and so constraint SD_k4 is redundant in the formulation LSD_k. The interpretation of μ is that it represents the marginal cost of a locomotive in this system. So, as the cost of the assignment "train j follows train i" is $(c_{ij}+\mu m_i+\mu M_{ij})$, any assignment of the trains considers a trade-off between the light-run cost and the cost of employing an

additional locomotive. If μ is positive and small compared to the light-run costs in the problem the schedules produced will use a large number of locomotives. Conversely, if μ is positive and large compared to the light-run costs involved, the solution will use fewer locomotives. Therefore, there is a value of μ which produces a "good" solution, i.e. a solution which incurs a low light-run cost without exceeding the limit on the number of locomotives available. The question is: how to calculate such a value of μ? In practice, British Rail make an educated guess for the value of μ based on a procedure known as "The Investigation of Extra Locomotive Cost". This procedure is carried out before the assignment problems are solved and the value of μ used in the assignment problems is chosen to be the same for each locomotive type. British Rail's heuristic proceeds by solving a series of assignment problems similar to LSD_k. The procedure can be outlined as follows.

Let the set $K_o=\{k \mid$ locomotive type k has been considered$\}$, initially $K_o=\emptyset$. Let $MLSD_k$ be a modified version of LSD_k with the fixed cost μU_k removed from the objective function.

i) Select a locomotive type $\overline{k}$ not yet considered and add $\overline{k}$ to the set K_o.

ii) Set $c_{ii}=0$ for all i such that $i \in P(\overline{k})$ and $i \in P(k)$ for some $k \notin K_o$.

iii) Solve the assignment problem $MLSD_{\overline{k}}$.

iv) From the solution at iii):

- if $x_{ij}^{\overline{k}}=1$ for $i \neq j$ then fix $x_{ij}^{\overline{k}}=1$ and remove train i and train j from each set P(k) where $k \notin K_o$.
- if $x_{ii}^{\overline{k}}=1$ and $i \notin P(k)$ for any $k \notin K_o$ then fix $x_{ii}^{\overline{k}}=1$.
- if $x_{ii}^{\overline{k}}=1$ and $i \in P(k)$ for some $k \notin K_o$ then reset c_{ii} to its original value.

Steps ii) and iv) are not necessary when considering the final locomotive type. The variables x_{ij}^k fixed at value 1 during the above procedure indicate that a locomotive type k should be used to work train i and that train j should follow train i. Using this

information it is then a simple matter to construct a set of schedules to work the timetable. There is no guarantee that the solution found by using this procedure is optimal. More importantly, it may happen that for one or more of the locomotive types there are insufficient locomotives available to work the schedules. It is for these reasons that the British Rail analyst may need to intervene during this procedure to move the solution towards feasibility or improve the solution value by banning or allowing certain connections. Clearly, the order in which the locomotives are considered will, in general, produce different solutions. Also, as already stated, the value of μ chosen affects the number of locomotives used in the solution and for this reason it may be worthwhile using a different value of μ for each locomotive type.

An outline of this procedure is also given by Wright [83] and referred to as the "Deterministic Algorithm". Wright seeks to override the need for manual intervention by use of stochastic algorithms. The first algorithm proposed by Wright is a simple local improvement method and the second uses the theory of simulated annealing. The starting point for both algorithms involves randomly assigning a locomotive type to a compatible train. Both algorithms attempt to improve the solution by banning connections and then evaluating the change in the value of the solution. In the case of the local improvement algorithm, only improvements in the cost of the solution are accepted. For the simulated annealing method, almost any perturbation to the solution is accepted initially, then towards the end almost no perturbation which increases the cost of the solution is accepted. The random element introduced at the start of the algorithms means that the algorithms can be repeated as many times as desired and only the best solution kept at each stage. Wright compares the results of the deterministic algorithm with those of the stochastic algorithms and concludes that, for larger problems, the simulated annealing approach outperforms the other methods. However, Wright points out that it is impossible to compare the solutions with an optimal solution as the optimal solutions are not known. Wright does not say how the solutions compare with those found by British Rail. The main drawback of Wright's method is that the limit on the number of locomotives of each

type is not considered and so it is not known if Wright's solutions are feasible. Wright does not propose a method for dealing with the situation where the solution found is infeasible.

Ferland and Michelon [29] review a number of heuristic and exact methods which may be used to solve the vehicle scheduling problem with time windows and multiple vehicle types. Although the problem they consider differs from the stock diagramming problem, some of the heuristic procedures they propose typify the way in which heuristic procedures are used to tackle combinatorially complex or large problems. Ferland's and Michelon's model includes time-windows for the start times of the tasks, temporal precedence constraints and depots for the vehicles. In the context of the Golden *et al.* [38] characterization of routing and scheduling problems, the problem might better be described as a combined routing and scheduling problem.

The mathematical formulation of the problem is based on the quasi-precedence graphs (N(k),A(k)) of the tasks associated with the vehicle type k where:

i) $i \in N(k)$ if $q_i \leq h_k$, where q_i is the vehicle capacity required to execute task i, and h_k is the capacity of vehicle k. Also, two nodes, s and t, are associated with the depot to indicate the start and finish of a vehicle sequence.

ii) arc $(i,j) \in A(k)$ if task j can be executed after task i, i.e. if

$$a_i + D_i^k + t_{ij}^k \leq b_j,$$

where:

- a_i is the earliest start time for task i;
- b_j is the latest start time for task j;
- D_i^k is the time for vehicle k to execute task i;
- t_{ij}^k is the time for vehicle k to travel from the end location of task i to the start location of task j.

Also, for each task $i \in N(k)$ there exists arcs $(s,i),(i,t) \in A(k)$.

Based on these graphs and denoting by K the set of vehicle types and by N the set of tasks, the mathematical formulation follows:

FM

$$\text{MIN } \Sigma_{k\in K}\Sigma_{(i,j)\in A(k)} c_{ij}^k x_{ij}^k$$

s.t.

$$\Sigma_{i\in P(j,k)} x_{ij}^k = y_j^k \qquad \forall\, j\in N(k)-\{s,t\},\ k\in K \qquad \textbf{FM1}$$

$$\Sigma_{j\in S(i,k)} x_{ij}^k = y_i^k \qquad \forall\, i\in N(k)-\{s,t\},\ k\in K \qquad \textbf{FM2}$$

$$\Sigma_{k\in Q(i)} y_i^k = 1 \qquad \forall\, i\in N \qquad \textbf{FM3}$$

$$x_{ij}>0 \Rightarrow ST_i+D_i^k+t_{ij}^k \le ST_j \qquad \forall\, (i,j)\in \bar{A}(k),\ k\in K \qquad \textbf{FM4}$$

$$a_i \le ST_i \le b_i \qquad \forall\, i\in N \qquad \textbf{FM5}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, (i,j)\in A(k),\ k\in K \qquad \textbf{FM6}$$

$$y_i^k \in \{0,1\} \qquad \forall\, i\in N,\ k\in K \qquad \textbf{FM7}$$

where $Q(i)=\{k \mid q_i \le h_k,\ k\in K\}$, $\bar{A}(k)=\{(i,j)\in A(k) \mid i\neq s,t;\ j\neq s,t\}$. For $i\in N(k)$, $P(i,k)=\{j\in N(k) \mid (j,i)\in A(k)\}$ and $S(i,k)=\{j\in N(k) \mid (i,j)\in A(k)\}$. The variables are defined as follows:

x_{ij}^k - 1 if a vehicle of type k traverses arc $(i,j)\in A(k)$;
- 0 otherwise.

y_i^k - 1 if task i is executed with a vehicle type k;
- 0 otherwise.

ST_i is a candidate start time for task i which is not fixed but must belong to the given time window $[a_i,b_i]$. The costs c_{ij}^k may be routing costs and/or investments costs.

Comparing the above problem to the stock diagramming problem, it is apparent that the graphs which underpin Ferland's and Michelon's problem can be made acyclic by the inclusion of the nodes s and t. This contrasts with the stock diagramming problem where the associated graphs are cyclic. The importance of this characteristic will become apparent as the heuristics presented by Ferland and Michelon are discussed. Also, provided that task i is a member of at least one of the sets N(k), there will be a feasible solution to the problem FM. For instance, a feasible solution would be to assign a separate vehicle to execute each task. This is not necessarily the case with the stock diagramming problem.

Ferland and Michelon discuss three possible heuristic procedures for solving the problem FM, though they do not report any results. It is clear that, as with the stock diagramming problem, the formulation FM gives rise to a large number of variables in any sizeable problem and it is therefore worthwhile considering heuristic methods.

The first heuristic procedure is based on a model which simplifies the problem FM by discretizing the time window interval. Swersey and Ballard [75] solve this discretized problem for a single vehicle type where the objective is to minimize the total number of vehicles used. They solve the discretized problem by first solving the linear programming relaxation of the integer programming problem. This method may yield continuous solutions and in such instances Swersey and Ballard achieved integer solutions by the addition of constraints or by manually adjusting the output. For a 102 task problem, Swersey and Ballard observed that an increase in the number of candidate start times in each of the time windows from 3 to 4 gave rise to problems with computer storage capacity. This suggests that the a discretized model of the problem FM may only be practical for smaller problems. An additional drawback of this method is that it may be necessary to call upon the intervention of an experienced scheduler to locate an integer solution.

The second heuristic method is based on a procedure known as the "two-phase method", originally proposed by Orloff [56]. Orloff only considers a single vehicle type, but Ferland and Michelon suggest a means of extending this method to the multiple vehicle case. Ferland and Michelon tackle the size of the problem by using cost considerations to decide which vehicle type should be assigned to a task and then implementing Orloff's two-phase method for a single vehicle type. The procedure is as follows:

i) For each task $i \in N$, associate a node i.

ii) With each pair of nodes i,j, associate two arcs:

- arc (i,j) with cost

$$c_{ij} = MIN_{k \in Q(i) \cap Q(j)} \{ t_{ij}^k + D_i^k \mid a_i + D_i^k + t_{ij}^k \leq b_j \},$$

- arc (j,i) with cost

$$c_{ji} = MIN_{k \in Q(i) \cap Q(j)} \{ t_{ji}^k + D_j^k \mid a_j + D_j^k + t_{ji}^k \leq b_i \},$$

where the minimum over an empty set is equal to ∞.

Phase one commences by solving a matching problem based on this graph. If an arc (i,j) with cost $c_{ij} < \infty$ exists in the matching, then an updated graph is constructed by collapsing the nodes i and j to form a single node i' and re-evaluating the costs on the arcs for this updated node set. The time-window for node i' is the intersection of the closed intervals $[a_i,b_i]$ and $[a_j,b_j]$, and $Q(i') = Q(i) \bigcap Q(j)$. This procedure is repeated until no further matchings are possible. At the end of the procedure, each node represents a feasible solution to the problem FM. This procedure is short-sighted in the sense that it makes choices based on immediate cost considerations and so it is possible that towards the end of phase one the costs on the arcs will become large. In phase two Orloff proposes an improvement procedure based on Lin's [48] "3-opt" heuristic for the Travelling Salesman Problem. A schedule is 3-opt if it is not possible to obtain a schedule with a better cost by replacing any 3 of its matchings with any other set of 3 matchings. The solutions produced are feasible, but not optimal in

general.

It is possible to apply this heuristic to the stock diagramming problem by associating a cost with the use of a locomotive type to control the number of locomotives of each type used. However, as with Wright's [83] methods, there is no guarantee that the solution found will be feasible.

In the third heuristic proposed by Ferland and Michelon a transportation problem is constructed at each iteration. As with the second heuristic the costs are used to decide which vehicle type should be assigned to a task. The precedence relationships in the model allow for the specification of source and sink nodes at each iteration. The set of source tasks includes the tasks most recently executed by the vehicles now available for other tasks. The set of sink tasks includes those tasks which have not yet been assigned, but cannot be executed after any of the other non-assigned tasks. As with the matching method, a series of transportation problems are solved until all tasks have been assigned.

It is not possible to apply this heuristic to the stock diagramming problem, as the cyclic nature of this problem means that such precedence relationships do not exist.

It is clear from the discussion so far that heuristic procedures such as Wright's or those proposed by Ferland and Michelon may not be adequate for solving the stock diagramming problem. For a problem with tight constraints on the number of locomotives of each type available these heuristics may fail to find feasible solutions. British Rail's procedure is used in practice, but relies heavily on the intervention of an experienced scheduler. These considerations suggest that it is necessary to investigate exact procedures for solving the problem. Orloff [56] and Swersey and Ballard [75] reject exact procedures on the grounds that the amount of computer time and storage required to solve such problems makes them impractical. Orloff states

that "...research on general fleet scheduling must focus exclusively on heuristic methods." However, Bott and Ballou [11] point out that developments in mathematical programming software and computer hardware have been encouraging, and methodologies which combine heuristic and exact procedures have led to success in obtaining optimal solutions to routing and scheduling problems. Also, in a paper on fleet routing and scheduling problems, Levin [47] states that "...the solution methods developed here and the extensive computational evidence gathered by the author in the course of his work lead to the rather gratifying conclusion that obtaining the optimal integer solutions to these large problems is indeed practicable." Encouraged by these comments, the following section reviews applications of mathematical programming techniques for solving routing and scheduling problems.

3.2.2 Mathematical Programming Approaches

Sub-Optimal Methods

Ryan [66] outlines an optimization technique for the solution of the air-crew scheduling problem. The problem involves the sequencing of crew movements in space and time so as to staff the airplane movements. The crew scheduling problem is essentially similar to vehicle scheduling problems, but the former usually involves numerous restrictions related to rest-periods, airline regulations, crew mix, etc. The crew scheduling problem is usually broken down into two stages.

The first stage of the process involves the production of *trips*. A trip is composed of a series of flights and rest periods which cover the airline timetable, and may run over a number of days. Trips originate and terminate at a crew base. The trips should be constructed so as to comply with all conditions imposed by the airline regulations and crew restrictions.

The second stage is referred to as "Rostering". In this stage the trips are

assigned to individual crew members to produce a line of work (LoW) over the rostering period for each crew member. It is this second stage procedure which is discussed in Ryan's paper.

The procedure starts by constructing a set of feasible LoW's for each crew member from which exactly one must be chosen. The candidate LoW's are the variables in the mathematical model of the rostering problem. The rostering problem can be formulated as a set partitioning problem as follows:

CSP

$$\text{MIN } \Sigma_j c_j x_j$$

s.t.

$$A \underline{x} = \underline{b} \qquad \text{CSP1}$$

$$x_j \in \{0,1\} \qquad \forall j \quad \text{CSP2}$$

where A is a 0-1 matrix partitioned to correspond to the LoWs for each crew member and the crew and trip constraints. So, for p crew members and t trips,

$$A = \begin{bmatrix} C_1 & C_2 & C_3 \ldots & C_p \\ L_1 & L_2 & L_3 \ldots & L_p \end{bmatrix}$$

Where $C_i = \underline{e}_i \underline{e}^T$ is a $(p \times n_i)$ matrix with $\underline{e}_i$ the i^{th} unit vector and $\underline{e}^T = (1,1,\ldots,1)$. The n_i LoWs for crew member i form the columns of the $(t \times n_i)$ matrix L_i with elements l_{qk} defined as $l_{qk}=1$ if the k^{th} LoW for crew member i covers the q^{th} trip and $l_{qk}=0$ otherwise. The right-hand side vector $\underline{b}$ is such that $b_i=1$ $(i=1,\ldots p)$ for the crew constraints and $b_i=r_i$ $(i=p+1,\ldots,p+t)$ for the trip constraints, where r_i = the number of crews required to work trip i.

The solution method for this set partitioning problem first solves the linear programming relaxation of the problem CSP and then applies a branching scheme to achieve integer solutions.

Ryan reports that the solution of the linear programming relaxation has proved to be a "computational bottleneck". One difficulty is the number of variables in the formulation CSP. Even with the use of sensible choice strategies to limit the number of LoW's generated, the number of variables entering into the linear programming model can often exceed 200000. Ryan deals with the size of the linear program by dividing up the variables into classes which are ranked in decreasing order of attractiveness. 'Attractiveness' may be a function of the objective value of an LoW, but could also be based on other considerations such as crew preferences. This division by class is then used during the solution of the linear program to control the problem size by only allowing certain classes of variables to be considered during the entering variable pricing stage of the primal convergence. If the solution is primally infeasible then variables from other classes are allowed to enter until primal feasibility is achieved. In effect, variables can be turned on and off as required.

The branching scheme used to resolve fractional solutions is a stronger tool than the conventional procedure which only fixes one variable along the 1-branch and the 0-branch at each node of the tree. In this situation a number of variables are set to zero on the 1-branch, but only one variable is fixed at zero along the 0-branch. Instead Ryan proposes constraint branching.

For an optimal fractional solution at any node of the branch and bound tree a crew constraint s and a trip constraint t can be identified such that,

$$0 < \Sigma_{j \in J(s,t)} x_j < 1$$

where $J(s,t) = \{ j \mid a_{sj} = 1 \text{ and } a_{tj} = 1 \}$. Then on the 1-branch set

$$\Sigma_{j \in J(s,t)} x_j = 1$$

i.e. crew member s must perform trip t, and on the 0-branch

$$\Sigma_{j \in J(s,t)} x_j = 0$$

i.e. crew member s must not perform trip t. Ryan implements the 0-branch by fixing all variables with an index in J(s,t) to take the value 0. The 1-branch is imposed by fixing all variables with an index in $\overline{J}(s,t)=\{j \mid (a_{sj}=1 \text{ and } a_{tj}=0) \text{ or } (a_{sj}=0 \text{ and } a_{tj}=1)\}$ to 0.

In practice, s and t are chosen at each node so that $\Sigma_{j \in J(s,t)} x_j$ is maximized and the branch and bound procedure is implemented by evaluating the 1-branch at each node and leaving the 0-branch unfathomed. The rationale behind this is that the 1-branch reflects the preference indicated by the linear programming optimum of a crew member s for a trip t at the current node. Therefore, the 0-branch can be disregarded as it is unlikely that a better solution will be found on a 0-branch. Ryan reports that the method used to solve the crew scheduling problem imposes a structure on the problem which means the relaxed linear programming solutions exhibit strong integer properties. The effect of this is that the constraint branching technique is effective in finding integer solutions, as any fractions which occur during branching must involve more than one crew member.

Ryan's solutions cannot be guaranteed optimal as all possible LoW's are not considered. However, this optimization technique has been successful in constructing feasible rosters for a real world problem. Ryan reports that the optimization approach is an improvement on heuristic methods. When using heuristic methods to solve this problem it was believed that the rostering problems were close to infeasibility, whereas Ryan's method shows that it is possible to find a number of alternative feasible solutions. An added benefit of the optimization technique is that it identifies

infeasibility with certainty. With the heuristic method it is unclear whether or not the problem is infeasible or the heuristic inadequate. This result is important when evaluating the case for using an optimization technique for the stock diagramming problem. For both Wright's heuristics and British Rail's heuristic it cannot be proven whether infeasibility is due to an insufficient availability of locomotive types or the limitations of the heuristic procedures.

Many crew-scheduling problems adopt a similar approach to that described by Ryan. These methods proceed by: generating a set of feasible schedules; solving the linear programming relaxation of a set-partitioning or set-covering problem over the restricted feasible region; implementing a branch and bound procedure to find an integer solution. See Wren [78] and Rousseau [64] for a number of examples of this approach.

In an earlier paper, Foster and Ryan [34] consider the Vehicle Scheduling Problem (VSP) formulated as:

VSP

$$\text{MIN } \Sigma_{j\in J} (V+m_j)\, x_j$$

s.t.

$$\Sigma_{j\in J}\, a_{ij}x_j = 1 \qquad i=1,\ldots,n \quad \textbf{VSP1}$$

$$x_j \in \{0,1\} \qquad \forall\, j\in J \quad \textbf{VSP2}$$

where the variables represent feasible routes with:

x_j - 1 if route j is in the schedule;
- 0 otherwise.

The data elements are:

V	-	mileage equivalent cost of each vehicle;
m_j	-	the total mileage of route j;
a_{ij}	-	1 if delivery i is covered in route j,
	-	0 otherwise.

J is the set of feasible routes and n is the number of deliveries to be made.

A feasible route in the set J may be required to comply with a number of restrictions, for example, vehicle capacity or route duration constraints. Nevertheless, if J contains all possible feasible routes this can give rise to a problem VSP with an extremely large number of x_j variables, rendering standard integer programming techniques impractical. Foster and Ryan attack the size of the problem by imposing additional constraints on the composition of J and thereby restricting the feasible region of the problem VSP. The method then proceeds by solving the linear relaxation of the problem VSP for the over-constrained set J. Integrality is maintained throughout by restricting the number of vehicles used to be a fixed integer and by introducing cutting planes to exclude fractionalities as they occur. The over-constrained feasible region is then successively relaxed to allow other promising routes to enter the problem. This approach is shown to be computationally attractive because of the near-integer nature of the over-constrained linear program. Also, Ryan and Foster find that the over-constrained approach produces good solutions as the limited set J contains the most promising routes. As with the crew-scheduling problem described above, the solutions found using this method cannot be guaranteed optimal as the entire feasible region is not considered when solving the problem. Foster and Ryan discuss the possibility of using a column generation technique to find optimal solutions to the VSP but they conclude that such a method is only practical for very small problems as it is "...slow to converge and rarely gives rise to near-integer or integer solutions at the linear programming optimum."

Optimal Methods

In contrast to Foster's and Ryan's experience, Desrosiers *et al.* [24] have reported success with the column generation method when applied to the problem of vehicle routing with time windows. Swersey and Ballard tackled this problem by solving a discretized version of the problem and thereby reducing the feasible region(see section 3.2.1). In Desrosiers *et al.* the continuous problem is considered and an optimal solution found over the entire feasible region.

The Desrosiers *et al.* formulation of the problem is constructed as follows. Let P be the set of trips i to be executed and I be the set of intertrip arcs, where an intertrip is an unproductive run between the end of one trip and the start of another. The network underpinning the model can be defined by a set of nodes N=PU{s,t}, where s and t represent the start and end depots for the vehicle, and a set of directed arcs A=PU{s}xPUPx{t}. Then:

RTWP

$$\text{MIN } \Sigma_{(i,j)\in A}\, c_{ij}\, x_{ij}$$

s.t.

$\Sigma_{j\in N}\, x_{ij} = \Sigma_{j\in N}\, x_{ji}$	$\forall\, i\in P$	**RTWP1**
$\Sigma_{j\in N}\, x_{ij} = 1$	$\forall\, i\in P$	**RTWP2**
$x_{ij} \geq 0$	$\forall\, (i,j)\in A$	**RTWP3**
$x_{ij} > 0 \Rightarrow t_i + t_{ij} \leq t_j$	$\forall\, (i,j)\in I$	**RTWP4**
$a_i \leq t_i \leq b_i$	$\forall\, i\in P$	**RTWP5**
$x_{ij} \in \{0,1\}$	$\forall\, (i,j)\in A$	**RTWP6**

The solution method involves constructing a set of feasible routes, then solving a set-partitioning problem to find an optimal subset of feasible routes which includes each trip exactly once. The set-partitioning problem is formulated as follows.

Let Ω : the set of routes, r, which satisfy the scheduling constraints.

y_r : a binary variable; 1 if route r is used, 0 otherwise

δ_{ri} : binary constant; 1 if route r includes trip i, 0 otherwise

c_r : cost of route r

MRTWP

$$\text{MIN } \Sigma_{r\in\Omega} c_r y_r$$

s.t.

$$\Sigma_{r\in\Omega} \delta_{ri} y_r = 1 \qquad \forall i\in P \quad \textbf{MRTWP1}$$

$$y_r \in \{0,1\} \qquad \forall r\in\Omega \quad \textbf{MRTWP2}$$

Desrosiers *et al.* do not consider all possible routes at the outset as this can give rise to a large number of variables y_r in the problem MRTWP. Instead a column generation procedure is used and the routes are constructed as required. The network constraints of RTWP are divided between the master problem and the subproblem. The master problem MRTWP deals with the requirement that each trip is performed exactly once and the corresponding subproblem is a routing problem with the addition of scheduling constraints. The subproblem is formulated as follows:

SRTWP

$$\text{MIN } \Sigma_{(i,j)\in A} (c_{ij} - \sigma_i) x_{ij}$$

s.t.

$$\Sigma_{j\in N} x_{ij} = \Sigma_{j\in N} x_{ji} \qquad \forall i\in P \quad \textbf{SRTWP1}$$

$$x_{ij} > 0 \Rightarrow t_i + t_{ij} \le t_j \qquad \forall (i,j)\in I \quad \textbf{SRTWP2}$$

$$a_i \le t_i \le b_i \qquad \forall i\in P \quad \textbf{SRTWP3}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j)\in A \quad \textbf{SRTWP4}$$

where σ_i are the dual values retrieved from the solution of the linear programming

relaxation of master problem.

The column generation procedure proceeds as follows:

1) Use an adaptation of the Bellman-Ford algorithm to find a set of least cost feasible routes for the subproblem.

2) Add the routes found at 1) to the master problem and solve the linear programming relaxation of the master problem.

3) Update the costs c_{ij} - σ_i using the dual values retrieved at stage 2). Go to step 1).

If at stage 1) no new routes are found the solution of the master problem is optimal and the procedure terminates.

The solution of linear programming relaxation of the master problem when optimality is achieved may be fractional. The generation of integer solutions is encouraged by introducing cuts which require an integer number of vehicles to be used and travel costs to be rounded up to the nearest integer, but if fractionality occurs an explicit enumeration procedure is used. At each node of the branch and bound tree the column generation technique is used to restore the linear programming relaxation of MRTWP problem to optimality. Therefore, Desrosiers *et al.* find optimal solutions to the problem RTWP.

Desrosiers *et al.* report that the algorithm allows large savings in the size of the fleet required to cover the trips and large problems with fairly wide time-windows can be solved. Compare this result with that found by Swersey and Ballard where problems were encountered with the computer storage space required to handle the linear programming problem.

For a further example of the use of a column generation procedure to solve

a vehicle scheduling problem see Riberio's and Soumis's [63] method for solving the Multiple-Depot Vehicle Scheduling Problem formulated by Carpento, Dell'amico, Fischetti and Toth [12].

The power of the column generation approach has been shown to be effective in dealing with routing and scheduling problems where the number of feasible routes and schedules is large. The results found by the above authors therefore suggest that such a method may be successfully applied to the stock diagramming problem. However, in a recent paper Forbes, Holt and Watt [32] claim to have found an exact procedure for solving the stock diagramming problem which explicitly considers all possible locomotive schedules.

Forbes *et al.* formulate the stock-diagramming problem as follows:

FSD

$$\text{MIN } \Sigma_i\Sigma_j\Sigma_k \, c_{ij}^k x_{ij}^k$$

s.t.

$$\Sigma_i\Sigma_k \, x_{ij}^k = 1 \qquad \forall \, i \qquad \textbf{FSD1}$$

$$\Sigma_j \, x_{ij}^k - \Sigma_j \, x_{ji}^k = 0 \qquad \forall \, i,k \text{ with } \delta_{ik}=1 \qquad \textbf{FSD2}$$

$$\Sigma_i\Sigma_j \, M_{ij}^k \, x_{ij}^k \leq U_k \qquad \forall \, k \qquad \textbf{FSD3}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall \, i,j,k \qquad \textbf{FSD4}$$

where:

- i,j are train indices;
- k is the locomotive type index;
- c_{ij}^k is the cost of train j following train i, with train i assigned a locomotive type k;
- M_{ij}^k is the number of times midnight passes between the end time of train i and the first feasible occurrence of train j as its successor using locomotive type k;

x_{ij}^k is a zero-one variable which is one if train j follows train i on a locomotive type k;

δ_{ik} is an indicator function which is one if train i may legally be operated by locomotive type k, and zero otherwise;

x_{ij}^k and c_{ij}^k exist only if $\delta_{ik}\delta_{jk}=1$.

Constraint FSD1 ensures that each train is worked exactly once and FSD2 ensures a conservation of flow.

Forbes *et al.* solve the linear programming relaxation of FSD, here on referred to as RFSD. A branch and bound scheme is then used to obtain integer solutions. Before solving RFSD, Forbes *et al.* formulate a further relaxation in which the locomotive type restrictions are removed. The formulation of this problem is equivalent to an assignment problem and is given by:

FAP

$$\text{MIN } \Sigma_i\Sigma_j\, e_{ij}z_{ij}$$

s.t.

$$\Sigma_j\, z_{ij}=1 \qquad \forall\, i \quad \textbf{FAP1}$$

$$\Sigma_j\, z_{ji}=1 \qquad \forall\, i \quad \textbf{FAP2}$$

$$z_{ij} \in \{0,1\} \qquad \forall\, i,j \quad \textbf{FAP3}$$

where:

e_{ij} is $\min_k c_{ij}^k$;

z_{ij} is a zero-one variable, which is one if train i is followed by train j.

The solution to FAP can be converted into a dual feasible solution to FSD by setting $x_{ij}^k=1$ if $z_{ij}=1$ and k is the "smallest" γ such that c_{ij}^γ equals e_{ij}. Using this solution to FAP Forbes *et al.* then solve the problem RFSD using the dual simplex algorithm.

This method appears promising at first sight as Forbes *et al.* report pleasing results. However, there are a number of drawbacks and assumptions made which invalidate the method as an exact way of solving a real world stock diagramming problem. The number of variables in a problem with many trains and locomotive types is extremely large. It was this consideration which led Wright and British Rail to use heuristic approaches. Forbes *et al.* describe a procedure for reducing the number of variables which uses a domination argument. They use Wright's data to test their method and observe that for some data sets "...any train which can legally be operated by locomotive type 1 or 2 can also be operated by locomotive type 3. If there are no constraints on the numbers of locomotives, and the objective value is independent of locomotive type, it is therefore possible to disregard locomotive types 1 and 2." There are two problems with this argument. Firstly, although the formulation FSD includes a constraint on the number of locomotives of each type available, they then disregard this constraint and this greatly simplifies the problem. If the locomotive availability constraint is present it is not possible to ignore locomotive types in the way they propose, as in a real world problem there is not usually the level of locomotive availability which allows a scheduler to avoid using two of the locomotives types in a feasible solution. The second problem with this procedure is that they use the fact that the objective value is dependent on locomotive type to solve the problem FAP and this suggests that locomotive type cannot be ignored in the way suggested. Even if the variables are reduced in this way, it can be seen from the results given by Forbes *et al.* that the number of variables remaining is large for any sizeable problem. A further drawback with this method arises when Forbes *et al.* use the solution to FAP to solve the problem RFSD. Recall that they assign the trains to the "smallest" locomotive type k such that c_{ij}^k equals e_{ij}. If the costs c_{ij}^k do not vary according to locomotive type, and this is the case with the stock diagramming problem, then it is likely that in the dual feasible solution to RFSD most of the trains are assigned to the smaller locomotive types. The consequence of this is that it is probable that the constraint FSD3 is violated for small values of k and ineffective for large values of k. In such instances the gap between the optimal

solution of FAP and the optimal solution of RFSD may be significantly larger than the gaps recorded by Forbes *et al.* for their test problems. In addition, if the costs c_{ij}^{k} do not vary according to locomotive type and the constraints on locomotive availability are tight, then it is likely that the gap between the solution to RFSD and FSD is significant, thereby incurring a large amount of branch and bound work.

CHAPTER FOUR

SOLUTION METHODS

4.1 Introduction

This chapter describes a number of solution methods considered for the stock diagramming problem, and draws conclusions from the results which justify the use of the solution method eventually adopted and presented in detail in the succeeding chapters.

In section 4.2 the data sets used to test the methodologies are introduced. Section 4.3 provides some background information on the way in which British Rail prepare data. Sections 4.4, 4.5 and 4.6 describe and discuss the three solution methods dealt with in this chapter.

4.2 Data Sets

British Rail supplied two sets of data along with solutions obtained using their heuristic procedure (outlined in Chapter three). The data sets are referred to as SET33 and SET189. SET33 consists of 33 trains and two locomotive types working in a region consisting of six stations. SET189 consists of 189 trains, including 'gap trains', and 10 locomotive types working in a region consisting of 26 stations. The concepts of 'gap trains' and 'regions' are discussed in section 4.3. It is these data sets which are used to test the solution methods considered in the latter sections of this chapter.

4.3 Regions, Areas and Gap Trains

Regions

As mentioned in Chapter two, British Rail solve the stock diagramming problem by dividing the railway network into regions. Each of the above data sets specifies the trains which must be worked in a particular region along with information on the stations in the region and the locomotives available to work the trains.

Areas

As well as a division of the railway network by region, British Rail may also specify areas within a region. It is normally the case that stations designated to be within an area are in close proximity, in terms of the light-run time between the stations, compared with the other stations in the region. To see what the purpose of this is, consider a region consisting of n stations and suppose that m ($m < n$) of these stations are said by British Rail to be within an 'area'. From the m stations chosen to be within the same area, British Rail also nominate a single station as the main station. Then, when solving the stock diagramming problem for the region, the value of the light-run cost used in the objective function is taken to be 0 for all light-runs between the m stations in the area. For light-runs from a station within the area to a station outside of the area the light-run cost used in the objective function is taken to be the value of the light-run time from the main station in the area to the station outside the area, and vice versa. In addition to designating certain stations to be within an area, British Rail automatically 'area' a station with itself. This means that $c_{ij}=0$ if the end location of a train i is the same as the start location of a train j. In any region a number of areas may be specified, but a station can only lie within at most one area. The interpretation of the use of these divisions by area to specify the objective function is that British Rail are interested in the amount of "out-of-area"

light-run time incurred in any solution. To illustrate the effect of using areas with an example, consider a region with six stations with the light-run times between stations as shown below.

	1	2	3	4	5	6
1	10	70	81	69	108	104
2	70	10	41	44	79	75
3	81	41	10	47	78	74
4	69	44	47	5	43	39
5	108	79	78	43	15	18
6	104	75	74	39	18	5

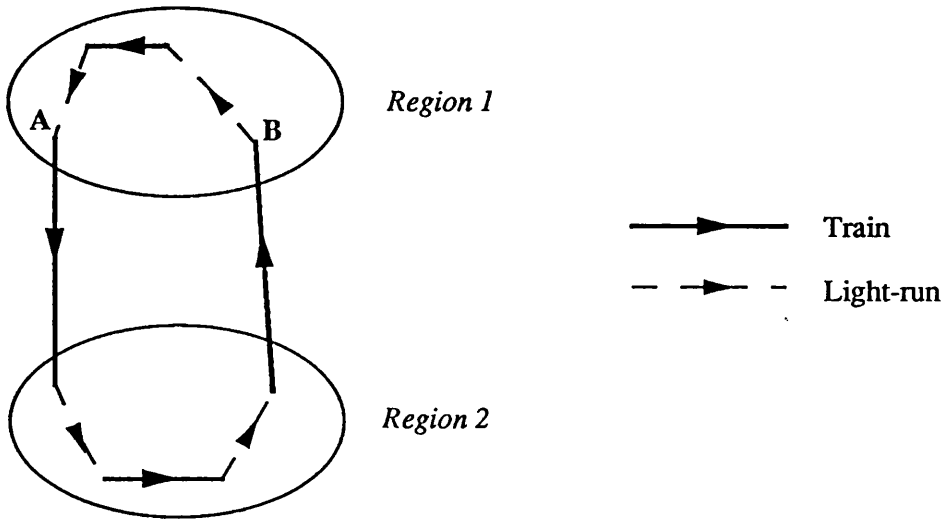
Suppose stations 5 and 6 constitute an area and station 5 is the main station. If the end location of a train i is station 5 and the start location of a train j is station 6, then in the objective function $c_{ij}=0$ rather than 18. If the end location of a train i is station 6 and the start location of a train j is station 1, then $c_{ij}=108$ rather than 104.

British Rail may also approximate the total light-run time in a solution by rounding down the values c_{ij} to the nearest 10. It should be noted that although the costs in the objective function may be adjusted to record out-of-area or approximate light-run times, the true light-run times are used in the model to ensure that connections between trains are feasible within the time available.

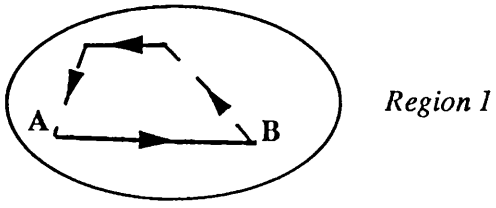
From here on, any solution presented for data sets SET33 and SET189 is based on an objective function in which c_{ij} has been rounded down to the nearest 10 and all stations are automatically 'area-ed' with themselves. Any further specification of areas within the region covered by a data set are made clear during the discussion of results.

Gap Trains

The concept of a gap train is a consequence of the specification of the problem on a region-wide basis. Consideration needs to be given to the spare capacity of those locomotives based outside but working into a region, the aim being to utilize the 'gaps' in such locomotive schedules to best advantage. To see how these gaps are used, suppose that a train has a start location in region 1 and an end location in region 2. This gives rise to the situation illustrated below where a locomotive based in region 1 works into region 2 and, as the schedule for a locomotive is cyclic, returns to region 1.



Consider region 1 only, then the picture 'appears' as:



It is clear from this representation that one can think of the locomotive working a 'train' from A to B with duration given by the time the locomotive takes to perform the section of the schedule between A and B. So, in effect, one can specify the time for which the locomotive is unavailable to work trains within region 1 by introducing the notion of a 'gap train'. The gap train is specified in the train timetable for region 1 in the same way as any other train within the region. The gap train has a start location and start time at A and an end location and end time at B. There is only one locomotive type compatible with this train, namely, the locomotive type which entered region 1 from region 2. Hence, once the gap train has been specified the stock diagramming problem can be solved as if all trains are restricted to running within a region.

The problem of assignment between regions will not be considered in this thesis. British Rail have supplied the data for single regions and the gap trains have been incorporated into the timetable for the region.

4.4 Exact Method

Recall the formulation of British Rail's stock diagramming problem given in Chapter two.

Let the set of trains be $\tau=\{i \mid i \text{ is the } i^{th} \text{ train in the timetable}\}$. Let the set of locomotives be $\Lambda=\{k \mid k \text{ is the number of the locomotive type}\}$. Then define P(k) to be the set of trains where $P(k)=\{i \mid \text{train } i \text{ is compatible with locomotive type } k\}$.

SD

$$\text{MIN} \quad \Sigma_{i\in\tau}\Sigma_{j\in\tau}\Sigma_{k\in\Lambda}\, c_{ij}\, x_{ij}^{k}$$

s.t.

$$\Sigma_{j\in P(k)}\, x_{ij}^{k} = y_i^{k} \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD1}$$

$$\Sigma_{j\in P(k)}\, x_{ji}^{k} = y_i^{k} \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD2}$$

$$\Sigma_{k\in\Lambda}\, y_i^{k} = 1 \qquad \forall\, i\in\tau \qquad \textbf{SD3}$$

$$\Sigma_{i\in\tau}\Sigma_{j\in\tau}\,(m_i\, y_i^{k} + M_{ij}\, x_{ij}^{k}) \le U_k \qquad \forall\, k\in\Lambda \qquad \textbf{SD4}$$

$$x_{ij}^{k} \in \{0,1\} \qquad \forall\, i,j\in P(k),\, k\in\Lambda \qquad \textbf{SD5}$$

$$y_i^{k} \in \{0,1\} \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD6}$$

The zero-one variables are:

x_{ij}^{k} - 1 if train j follows train i on locomotive type k,
- 0 otherwise.

y_i^{k} - 1 if train i is worked by locomotive type k,
- 0 otherwise;

The data elements are:

c_{ij} - light-run cost from the end location of train i to the start location of train j;

m_i - the number of times a locomotive crosses midnight while working train i;

M_{ij} - the number of times a locomotive crosses midnight while performing the light-run from the end location of train i to the start location of train j;

U_k - the total number of locomotives of type k available.

It is this formulation which underpins the exact method discussed here.

There are various commercial packages available which can be used to find exact solutions to integer linear programming problems such as SD. It seemed obvious that the first step in the search for an optimal solution would be to run the sets of test data on one of these packages. The aim of this exercise was to compare the speed, improvement and veracity of the results against those found by British Rail. The tests were performed on a commercial mathematical programming package known as LAMPS [41]. LAMPS is an acronym for Linear And Mathematical Programming System, and is described as '..a general purpose Mathematical Programming code, which is designed to solve problems of many thousands of rows.' For the formulation SD, if n is the number of trains and K is the number of locomotive types then there is a maximum of (n^2K+nK) variables (columns) and ($2nK+n+K$) constraints (rows). A 'maximum' because all trains are not necessarily compatible with all locomotive types. The software required to specify the problem in AMS format, as required by the LAMPS optimizer, was developed from scratch using the programming language FORTRAN.

4.4.1 SET33

The first test is performed for the data set SET33. To evaluate the results obtained by solving SD on LAMPS for SET33, first consider the solution British Rail found using their heuristic procedure (outlined in chapter three).

British Rail's Solution

Before entering the heuristic procedure British Rail perform two preliminary investigations. First they calculate a peak requirement for the number of locomotives. The peak requirement figure gives a lower bound on the number of locomotives needed to work the timetable. The method used to calculate this lower bound is

described in detail in chapter nine. For SET33 it was found that 15 locomotives are needed, 3 less than the total of 18 locomotives available (12 type 1 and 6 type 2). The second investigation calculates a lower bound on the light-run cost for the problem. The lower bound is found by ignoring the type restrictions and solving an assignment problem with input costs $c_{ij}+\mu m_i+\mu M_{ij}$. c_{ij} is the light-run cost incurred when performing a light-run from end location i to start location j. For data set SET33 two stations were designated to be within an area and so c_{ij} was adjusted to reflect the cost of the out-of-area light-running. m_i is the number of times train i crosses midnight and M_{ij} is the number of times a locomotive crosses midnight in performing the light-run from i to j. μ is the marginal cost of a locomotive expressed in terms of light-run time. As explained in chapter three, by adjusting μ it is possible to vary the number of locomotives used. Setting μ equal to 180 light-run minutes for SET33, British Rail found that only 15 locomotives were used and the total light-run cost incurred was 390 light-run minutes i.e. $\Sigma_i\Sigma_j c_{ij}=390$. For SET33 this figure is the lower bound on a solution which uses 15 locomotives.

On completion of these preliminary investigations, British Rail then use their heuristic procedure to find a set of schedules to work the trains in the region at minimum cost. At various stages during the execution of the heuristic an experienced user intervenes to manipulate the schedules and force or block specific connections. This was done for SET33 and a solution which uses 390 light-run minutes and 15 locomotives was found.

Exact Solutions for SET33 Using LAMPS

The maximum possible number of variables for this set is 3,244 and the maximum number of constraints is 167. Some of these variables and constraints may not exist as all trains are not compatible with all locomotive types. The AMS data file required by LAMPS used 257 Kbytes of memory. Four test runs are performed. Each time the linear programming relaxation of SD, referred to as RSD, is solved.

TEST 1

For the purposes of comparison, the values of data c_{ij} are taken to be the same as those used by British Rail. The optimal solution to RSD was found in 29 seconds of CPU time. The solution cost was 140 light-run minutes and all of the 18 locomotives available were used. The solution found was integer but this cannot be guaranteed in general.

TEST 2

The formulation used in test 1 is modified in test 2 so that the objective function of RSD changes from:

$$\Sigma_i \Sigma_j \, c_{ij} \, x_{ij}^k$$

to:

$$\Sigma_i \Sigma_j \, c_{ij} \, x_{ij}^k + \mu \, m_i \, y_i^k + \mu \, M_{ij} \, x_{ij}^k$$

with $\mu = 180$ to coincide with British Rail's parameter for the cost of a locomotive.

The CPU time to find an optimal solution to RSD with the modified objective function was 31 seconds. The objective function value was 390 light-run minutes and the solution used 15 locomotives. In this instance the solution was not integer. LAMPS uses Branch and Bound to solve integer programming problems. By specifying to LAMPS that the variables y_i^k should take integer values, an integer solution was found in a further 30 seconds of CPU time. Four nodes of the branch and bound tree were explored. The optimal integer solution cost 390 light-run minutes and used 15 locomotives. The cost of this solution agrees with the solution found by British Rail and the same number of locomotives are used. The schedules found differed from those found by British Rail, but this is due to the existence of multiple solutions.

TEST 3

The formulation used in test 1 is modified by the addition of a constraint to limit the total number of locomotives used. The new formulation is:

SD_T

$$\text{MIN} \quad \Sigma_{i\in\tau}\Sigma_{j\in\tau}\Sigma_{k\in\Lambda}\, c_{ij}\, x_{ij}^{k}$$

s.t.

$$\Sigma_{j\in P(k)}\, x_{ij}^{k} = y_i^k \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD1}$$

$$\Sigma_{j\in P(k)}\, x_{ji}^{k} = y_i^k \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD2}$$

$$\Sigma_{k\in\Lambda}\, y_i^k = 1 \qquad \forall\, i\in\tau \qquad \textbf{SD3}$$

$$\Sigma_{i\in\tau}\Sigma_{j\in\tau}\,(m_i\, y_i^k + M_{ij}\, x_{ij}^k) = N_k \qquad \forall\, k\in\Lambda \qquad \textbf{SD4}$$

$$N_k \le U_k \qquad \forall\, k\in\Lambda \qquad \textbf{SD5}$$

$$\Sigma_k\, N_k \le \text{TOTAL} \qquad \textbf{SD6}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, i,j\in P(k),\, k\in\Lambda \qquad \textbf{SD7}$$

$$y_i^k \in \{0,1\} \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \textbf{SD8}$$

where:

N_k	-	number of locomotives of type k used;
TOTAL	-	total number of locomotives used.

This modification adds K variables and K+1 constraints to the problem, where K is the number of types of locomotive.

For this test TOTAL is set at 15. The solution to the linear programming relaxation of SD_T was found after 26 seconds of CPU time. The cost of the solution was 390 light-run minutes and, as expected, 15 locomotives were used. The solution

was continuous, but the Branch and Bound algorithm was used as in test 2. The integer solution was found after a further 15 seconds of CPU time and a search of 6 nodes. The integer solution used 390 light-run minutes and 15 locomotives.

TEST 4

Having set up the problem as in test 3, it is easy to vary the value of parameter TOTAL and test the objective value found. The results shown in table 4.4.1.i refer to the integer solutions found, except where the solution is infeasible.

TOTAL	OBJECTIVE VALUE
18	140
17	140
16	220
15	390
14	Infeasible

Table 4.4.1.i

These results show that 14 locomotives are insufficient to work the trains, verifying the peak requirement of 15 locomotives found by British Rail. They also suggest that the cost saving of 170 light-run minutes may make a solution which uses 1 locomotive above the peak requirement more attractive.

The results of these tests on the SET33 data are encouraging in a number of ways. Optimal solutions can be found in a short time and, once the problem has been defined, it is easy to vary the limit on the total number of locomotives used or adjust the number of each type used. LAMPS will also allow the user to modify the problem easily so that certain connections can be banned or forced, or the light-run values changed. If the solution is multiple, i.e. if the reduced cost for some of the non-basic variables is zero, then trains may be alternatively scheduled at no increased cost.

The use of this formulation performs well for the small data set SET33, but how will it perform for the larger data set SET189? Consider formulation SD for SET189. SET189 consists of 189 trains and 10 locomotive types. This suggests a maximum of 392,931 variables and 3,979 constraints. Given that the AMS file required by LAMPS for the SET33 problem used approximately 0.25 Mbytes of computer memory, these values suggest that AMS file for SET189 would be very expensive in terms of storage. Also, LAMPS uses the simplex algorithm to solve the linear programming problem RSD and, as this algorithm is not known to be a polynomial time algorithm, the time to solve a problem of this size could be unacceptably large. It is for these reasons that the following sections investigate alternative methods for solving the stock diagramming problem.

4.5 Reduction of Variables

The premise behind the 'Reduction of Variables' method is that certain connections are 'bad' in that they involve long light-runs which would probably never be used in an optimal or near optimal solution. Based on this observation a number of strategies are devised in the hope of sensibly reducing the number of variables to a manageable size. So, using this method a restricted feasible region is considered.

4.5.1 Data Sets

The data set used to test the strategies is based on SET189 and referred to as SET178. SET178 consists of 178 trains and 5 locomotive types. This 'artificial' data set is constructed by first removing the gap trains. As well as identifying locomotives by type, each locomotive type is a member of a set known as 'locomotive class'. Having removed the gap trains it is possible to partition the locomotive types compatible with the remaining 178 trains into 5 classes. These 5 classes are then used as if they are the locomotive types for the data set SET178. This data set is developed solely to test this method. It was decided that SET33 is too small, but SET189 is too large for the initial stages of testing and SET178 represents a compromise. If the results found for SET178 suggest that the method is viable, then the next step is to test the method for the real data set SET189.

4.5.2 Reduction of Variables Formulation

An alternative formulation to SD is used to test the Reduction of Variables method. The sets τ, Λ and P(k) are as defined for SD. Let A be the set of pairs where $A=\{(i,j) \mid$ the light-run from train i to train j is permitted$\}$. Then define $S(i)=\{j \mid (i,j)\in A\}$ and $\overline{S}(i)=\{j \mid (j,i)\in A\}$. Then the new formulation is:

RV_1

$$\text{MIN} \quad \Sigma_{i\in\tau}\Sigma_{j\in\tau}\Sigma_{k\in\Lambda}\, c_{ij}\, x_{ij}^{k}$$

s.t.

$$\Sigma_{j\in S(i)}\, x_{ij}^{k} = y_i^k \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \mathbf{RV_1 1}$$

$$\Sigma_{j\in \overline{S}(i)}\, x_{ji}^{k} = y_i^k \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \mathbf{RV_1 2}$$

$$\Sigma_{k\in\Lambda}\, y_i^k = 1 \qquad \forall\, i\in\tau \qquad \mathbf{RV_1 3}$$

$$x_{ij}^{k} \in \{0,1\} \qquad \forall\, i,j\in P(k),\, k\in\Lambda \text{ with } (i,j)\in A \qquad \mathbf{RV_1 4}$$

$$y_i^k \in \{0,1\} \qquad \forall\, i\in P(k),\, k\in\Lambda \qquad \mathbf{RV_1 5}$$

The variables and data are as defined for formulation SD.

Notice that there is no limit on the number of locomotives available. This allows for some freedom in the problem at this stage where the methodology is being developed. The specification of the set A controls the number of variables in the problem. The construction of this set is based on the strategy being used to reduce the number of variables. Three such strategies are tested.

Strategy 1

Only allow connections from a station to the nearest three stations i.e. the station itself and its two nearest neighbours. This strategy ensures that only the smaller light-runs enter into any solution. This specification of the set A has the effect of splitting up a region into sectors by clustering the stations. Connections between the sectors are effected by trains working out of one sector and into another. The number of light-run variables x_{ij}^k resulting from this strategy is 18,001.

The linear programming relaxation of RV_1 was solved using LAMPS. The solution found was infeasible with a sum of infeasibilities equal to 12. This figure of 12 indicates by how much the trains are being underworked, or more precisely $\Sigma_i\Sigma_k y_i^k = 166 = (178 - 12)$.

In an attempt to gain greater insight into the structure of the problem an alternative formulation is used on data set SET178 for strategy 1:

RV_2

$$\text{MIN} \quad \Sigma_{i\in\tau}\Sigma_{j\in\tau}\Sigma_{k\in\Lambda}\, c_{ij}\, x_{ij}^{k}$$

s.t.

$$\Sigma_{j\in S(i)}\, x_{ij}^{k} = y_i^k \qquad \forall\, i\in P(k),\ k\in\Lambda \qquad \mathbf{RV_1 1}$$

$$\Sigma_{j\in \bar{S}(i)}\, x_{ji}^{k} = y_i^k \qquad \forall\, i\in P(k),\ k\in\Lambda \qquad \mathbf{RV_1 2}$$

$$\Sigma_{k\in\Lambda}\, y_i^k \geq 1 \qquad \forall\, i\in\tau \qquad \mathbf{RV_1 3}$$

$$x_{ij}^{k} \in \{0,1\} \qquad \forall\, i,j\in P(k),\ k\in\Lambda \text{ with } (i,j)\in A \qquad \mathbf{RV_1 4}$$

$$y_i^k \in \{0,1\} \qquad \forall\, i\in P(k),\ k\in\Lambda \qquad \mathbf{RV_1 5}$$

where the sets, variables and data are as defined in RV_1.

This formulation allows a train to be worked more than once. A train may be overworked in order to ensure that the problem is feasible. That is, a light-run which is not permitted by strategy 1 is mimicked by the overworking of the trains. Alternatively a train may be overworked in order to reduce the total light-run cost. By isolating those trains which are overworked because of feasibility considerations it may be possible to identify where the deficiency in the light-running is occurring, and thereby devise an alternative strategy to strategy 1. The procedure used to do this is as follows.

The linear programming relaxation of the problem RV_2 for strategy 1 is solved and the value of OVERWORK $= \Sigma_i\Sigma_k\, (y_i^k - 1)$ recorded. Attempts are then made to reduce the value of OVERWORK by setting $\Sigma_k y_i^k = 1$ for a particular train i currently overworked, then resolving and recalculating the value of OVERWORK. Where possible train i is chosen if it is worked within a sector. If this is the case, as the available light-run connections are confined to sectors, it is probable that train i is being overworked to reduce the value of the objective function. Given that the sum of infeasibilities is 12, it is known that the minimum possible value of OVERWORK

is 12. The results of this exercise are shown in the table below.

ITERATION	OVERWORK
1	22
2	22
3	20
4	17
5	18
6	16
7	13
8	12
9	15
10	12
11	12

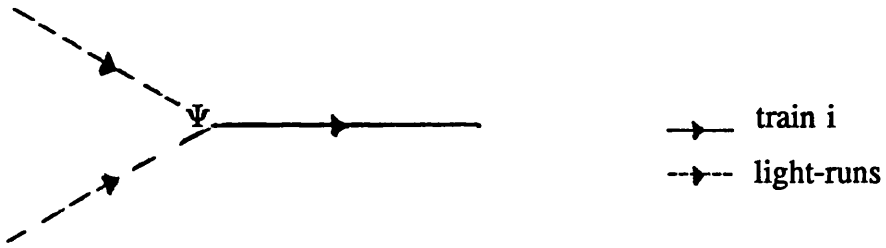
Table 4.5.2.i

It was observed that there is an imbalance in the number of trains working in to and out of each sector. Also, some stations are more 'popular' than others in that there are more trains arriving and departing from these stations, and from the results of iterations 8, 10 and 11 it appeared that the overworked trains were passing through these stations. The second strategy is based on these observations.

Strategy 2

The proposal for the second strategy is that the number of light-runs associated with each station are weighted according to the popularity of the station. This not only allows more light-runs into and out of popular stations, but also allows 'out-of-

sector' light-runs to enter into the problem. Suppose that a train i departs from station Ψ. Order all possible light-runs x_{ji}^k from a train j to the train i for a locomotive type k according to the increasing duration of the light-runs. If the weighting chosen for station Ψ is α, then select the first α of these light-runs as candidate light-runs into the start of train i on locomotive type k. So, if the weighting for station Ψ is 2, for a specific locomotive type the picture is:



This procedure is repeated for all locomotive types compatible with train i. The same procedure is then carried out for the arrival station for train i. Once each train in the problem has been considered the set A is completely specified for strategy 2.

The value of α for a station is chosen to be equal to the total number of trains arriving at and departing from a station. The value of α is then adjusted where necessary so that it lies within specified upper and lower bounds. For the first test of strategy 2 the upper andlower bounds are chosen to be 30 and 5, respectively. So, if $\alpha<5$ then set $\alpha=5$ and if $\alpha>30$ then set $\alpha=30$. Using these weightings the number of x_{ij}^k variables generated is 15,491.

The solution to the linear programming relaxation of RV_1 for strategy 2 was infeasible, but the sum of the infeasibilities was reduced to 3. The linear programming relaxation of RV_2 is then solved for strategy 2. As with strategy 1, the value of OVERWORK is calculated and then successive attempts are made to reduce this value. The results are as follows:

ITERATION	OVERWORK
1	11
2	11
3	13
4	11
5	10
6	10
7	10
8	10
9	9
10	9
11	9
12	6
13	5
14	5
15	3

Table 4.5.2.ii

At this point no further reductions are possible. This exercise did not suggest an alternative strategy for reducing the number of variables. It may be possible to find a feasible solution to RV_1 by increasing the lower and upper limits on α. However, such a step would increase the number of variables without guaranteeing a feasible solution to the problem SD. An alternative strategy is proposed.

Strategy 3

Strategy 3 is a simple modification of strategy 2. This time all 'loop' light-

runs are permitted to enter the problem. That is, for each train i and each locomotive type k compatible with train i, all x_{ii}^k variables are included in the problem RV_1. This means the addition of a maximum possible of 178 x 5 variables (178 trains and 5 locomotive types). The solution to formulation RV_1 is bound to be feasible as no limit has yet been imposed on the number of locomotives. The linear programming relaxation of RV_1 was submitted to LAMPS, but after 36,000 iterations an optimal solution had not been found and therefore the optimization procedure was interrupted.

It was felt at this stage that the Reduction of Variables approach was not proving promising, and this solution method was abandoned and an alternative method tested.

4.6 Generating Sequences

A sequence in this context describes a schedule for a locomotive which works a subset of the trains in the stock diagramming problem. A sequence is characterized by: the trains in the sequence; the order in which the trains are worked in the sequence; and, the locomotive type used to work the sequence. Each train in a sequence must be compatible with the locomotive type which works the sequence. The cost of the sequence is the sum of the light-running between trains in the sequence. The aim of this method is to construct a set of sequences Ω, and from this set choose a subset which works the trains at minimum cost without exceeding the limits on locomotive availability. Only a subset of the set of all feasible sequences is constructed. Therefore, as with the Reduction of Variables Procedure, this method is used to solve the stock diagramming problem over a limited feasible region. Obviously the cost of the optimal solution depends on how 'good' the set Ω is. At the other extreme, a 'bad' set of sequences Ω may yield an infeasible solution. The notional idea is that a 'good' sequence does not include too many long light-runs and is not expensive in terms of the number of locomotives used. A heuristic which takes account of the light-run time and the limit on the number of locomotives

available is used to generate a set of sequences.

4.6.1 Heuristic Procedure used to Generate Sequences

Before using the heuristic procedure to generate sequences, the problem is decomposed according to locomotive type compatibility. It is easiest to describe the method employed using an example data set. Suppose that there are 5 trains to be worked in the data set, so the set of trains $\tau=\{1,2,3,4,5\}$. Also suppose that the data set has two locomotive types, type 1 and type 2. Let the pattern of compatible locomotive types for the trains in set τ be:

train no.	compatible locomotive types
1	1
2	2
3	1
4	1 2
5	1 2

Using this information each train in the data set can be categorized according to the compatible locomotive types in the way shown in table 4.6.1.i.

TYPE 1 ONLY	TYPE 2 ONLY	TYPE 1	TYPE 2	TYPE 1 AND 2
1		1		
	2		2	
3		3		
		4	4	4
		5	5	5

Table 4.6.1.i

There are five different categories shown in table 4.6.1.i. Let these categories be referred to as 'type sets'. As indicated in the table, the trains which can only be worked by a locomotive of type 1 are trains 1 and 3. Trains which can be worked by a locomotive of type 1 but may or may not be compatible with a locomotive of type 2 are trains 1,3,4 and 5. Hence, the type set 'TYPE 1 ONLY'={1,3} is a subset of 'TYPE 1'={1,3,4,5}. Also, 'TYPE 1 AND 2'={4,5} is the intersection of type sets 'TYPE 1'={1,3,4,5} and 'TYPE 2'={2,4,5}. In general, the type sets are constructed as follows. Suppose that a data set has K locomotive types. For a given value of $L \leq K$, construct type sets based on all possible combinations of length L of the K locomotive types. There are

$$\frac{K!}{(K-L)!L!}$$

such combinations. Suppose that one of these combinations of length L contains the locomotive types $k_1,k_2,\ldots,k_L$. Based on these L locomotive types two type sets are constructed. The first type set is given by 'TYPE k_1 AND TYPE k_2...AND TYPE k_L' and includes all trains which are compatible with the L locomotive types $k_1,k_2,\ldots,k_L$, but may also be compatible with other locomotive types besides these. The second type set is given by 'TYPE k_1 AND TYPE k_2...AND TYPE k_L ONLY' and includes those trains which are compatible with the L locomotive types $k_1,k_2,\ldots,k_L$, but are not compatible with any other locomotive types besides these. Therefore, each value of L gives rise to

$$\frac{2K!}{(K-L)!L!}$$

type sets, except when L=K in which case the two type sets described above are equivalent. Varying the value of L from 1 to K generates

$$1+\sum_{L=1}^{L=K-1}\frac{2K!}{(K-L)!L!}$$

type sets. Some of these type sets may be empty and can therefore be ignored.

The heuristic then proceeds by constructing an assignment problem for each of the type sets. The costs in the assignment matrix are the light-run costs c_{ij}. The assignment problem formulation for a type set χ is:

AP

$$\text{MIN } \Sigma_{i\in\chi}\Sigma_{j\in\chi}\, c_{ij}\, a_{ij}$$

s.t.

$$\Sigma_{j\in\chi}\, a_{ij} = 1 \qquad \forall\, i\in\chi \quad \textbf{AP1}$$

$$\Sigma_{j\in\chi}\, a_{ji} = 1 \qquad \forall\, i\in\chi \quad \textbf{AP2}$$

$$a_{ij} \in \{0,1\} \qquad \forall\, i,j\in\chi \quad \textbf{AP3}$$

The zero-one variables are:

a_{ij} - 1 if the locomotive performs a light-run from the end location of train i to the start location of train j,
- 0 otherwise.

The data are:

c_{ij} - the light-run time from the end location of train i to the start location of train j.

χ is the type set being considered where χ is a subset of the trains in the data set considered.

The matrix structure of the problem AP is unimodular, therefore the condition that a_{ij} is zero-one variable can be relaxed.

The solution of the assignment problem gives a valid set of sequences for all locomotive types associated with the type set χ: 'valid' in the sense that each train is compatible with the locomotive types which define with type set χ and the total light-running cost of a sequence is the sum of the costs on the light-run arcs which make up the sequence. As an example, a solution for the type set TYPE 2={2,4,5} might be:

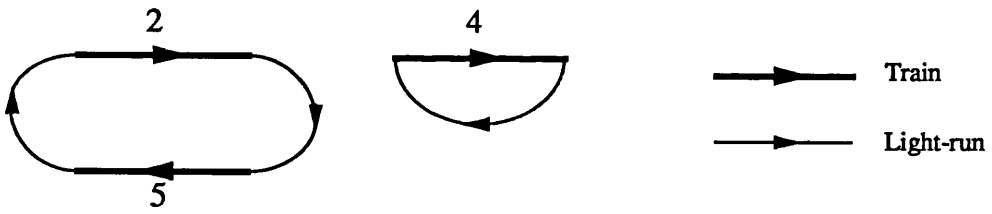


Figure 4.6.1.i

The formulation AP will tend to eliminate long light-runs, but as AP does not consider the number of locomotives available the result is that any solution will reduce the total light-run cost by using more locomotives. This is not in line with the notion of a good sequence. The solution to this problem lies in considering the two schedules below.

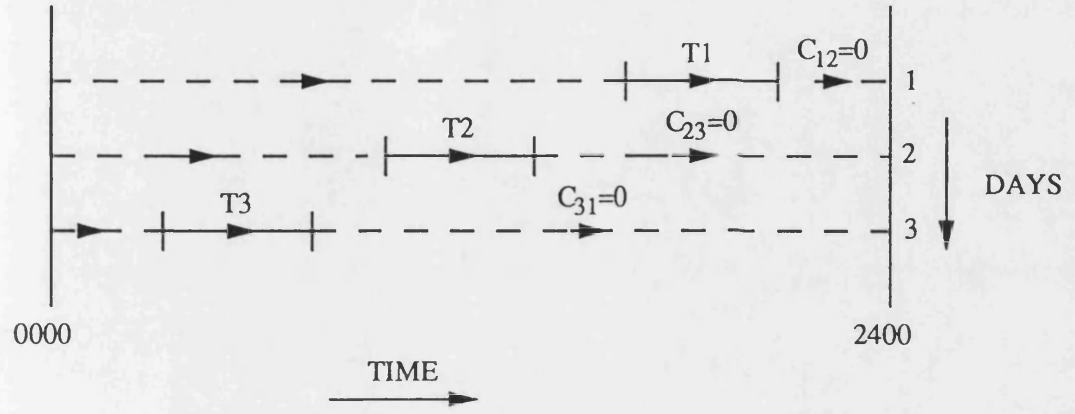


Figure 4.6.1.ii

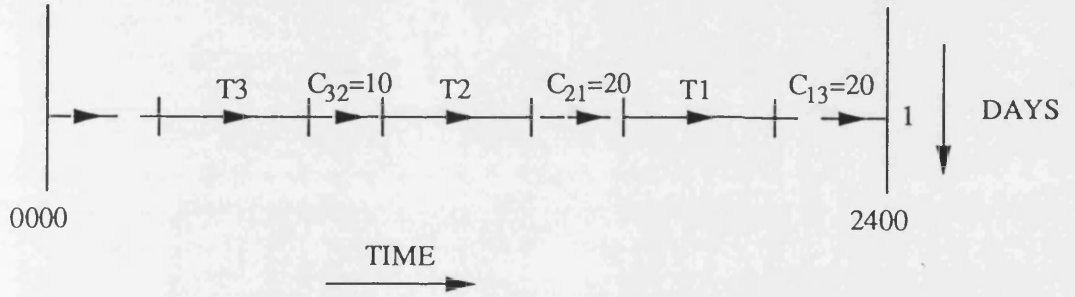


Figure 4.6.1.iii

In the above figures the solid lines represent the train arcs and the dashed lines represent the light-run arcs. 'Tn' indicates the train n.

In figure 4.6.1.ii the assignment cost is 0, but the schedule uses 3 locomotives. In figure 4.6.1.iii the assignment cost is 50, but the schedule only uses 1 locomotive. Both schedules work the same three trains. However, in the second schedule time between connections has been reduced and the result of this is that second schedule requires fewer locomotives. Suppose that the time which elapses

between the end of a train i and the start of a train j when the connection i→j is made is t_{ij} minutes. With a view to reducing the amount of time between connections in the sequences generated by solving AP, the assignment problem is modified to allow only those connections which are possible within a pre-specified time interval. This restriction on the admissible connections is made possible by introducing the condition that $t_{ij} \leq$ INTERVAL, where INTERVAL is a parameter supplied in the heuristic procedure. In the formulation AP the condition is implemented by setting $c_{ij} = \infty$ if $t_{ij} >$ INTERVAL. (In practice c_{ij} is set equal to a large value M.) This restriction on the connection time between trains has two effects which encourage the generation of good sequences. Firstly the idle-time, or the time when a locomotive is not working a train or performing a light-run, is reduced. Secondly, the maximum light-run time which can occur in a schedule is restricted to be no greater than the value of INTERVAL.

It was discovered that in their work on air-crew scheduling Ryan and Falkner [67] used a similar procedure to limit the amount of idle-time in a schedule. The air-crew scheduling problem involves constructing an optimal schedule of "duties" (or sequences in the stock diagramming problem), each of which performs a sequence of "runs" (or trains in the stock diagramming problem). Ryan and Falkner do not consider all possible runs which could follow run r in a duty, instead they construct a list of "next availables" by "..restricting the idle time between runs." This allows them to restrict the number of variables in the problem and, at the same time, encourage a solution which reduces idle-time between duties.

The value of INTERVAL can be increased during the execution of the heuristic procedure to generate different sequences by allowing additional light-run variables to enter the assignment problem. Such an outcome is not guaranteed though as an increase in INTERVAL may not result in a change in the optimal solution of the assignment problem. This means that repeated sequences may be created. A problem arises if INTERVAL is fixed too low. It might happen that it is not possible

to schedule all trains in a type set as too many connections are banned. In this case the value of INTERVAL is increased until a first solution is found, and the corresponding value of INTERVAL is recorded as base value. This base value is characteristic of the type set. The solution to the problem AP for a type set gives valid sequences for each of the locomotive types considered by the type set. So, for a type set which considers the locomotive types $i_1,i_2,\ldots,i_L$, the assignment problem is solved once and copies of the sequences found are then taken for each of the locomotive types $i_1,i_2,\ldots,i_L$.

The heuristic procedure used to generate a set of sequences is known as HAP and summarized as follows. Once all possible non-empty type sets have been constructed for the stock diagramming problem being considered, the heuristic procedure HAP commences. For each type set in turn:

i) For a given value of INTERVAL solve the problem AP.

ii) If the solution to AP is finite go to step iii). Otherwise, increase the value of INTERVAL and go to step i).

iii) Make copies of the sequences found at i) for each locomotive type considered by the current type set. Add the sequences to the set of sequences Ω.

Steps i), ii) and iii) can be repeated for a number of values of INTERVAL.

Once a set of sequences Ω is found for the problem, the next step is to find a subset of sequences from the set Ω which works each train exactly once at minimum cost and uses no more than the available number of locomotives of each type.

4.6.2 Set Partitioning Problem

Given the set of sequences Ω the problem of selecting an optimal subset of the

set of sequences is a set-partitioning problem. The set-partitioning formulation for the stock diagramming problem is:

SPP

$$\text{MIN } \Sigma_{r\in\Omega}\, c_r s_r$$

s.t.

$$\Sigma_{r\in\Omega}\, \delta_{ri} s_r = 1 \qquad \forall\, i\in\tau \qquad \textbf{SPP1}$$

$$\Sigma_{r\in\Omega}\, L_{rk}\, s_r \le U_k \qquad \forall\, k\in\Lambda \qquad \textbf{SPP2}$$

$$s_r \in \{0,1\} \qquad \forall\, r\in\Omega \qquad \textbf{SPP3}$$

The sets τ and Λ are as defined for formulation SD.

The zero-one variables are:

s_r - 1 if sequence r is used,
- 0 otherwise.

The data elements are:

δ_{ri} - is a binary constant; 1 if sequence r includes train i,
- 0 otherwise;

c_r - cost of sequence r equal to the sum of the light-run costs which make up the sequence r;

L_{rk} - the number of locomotives of type k required to work sequence r;

U_k - the total number of locomotives of type k available to work the timetable.

4.6.3 Testing the Generation of Sequences Method

SET33

The testing of 'Generation of Sequences' method proceeds by: constructing the type sets for the data set; using the heuristic HAP to generate a set of sequences Ω; then, solving the linear programming relaxation of the problem SPP, from here on referred to as RSPP.

The initial tests of this method are performed on the data SET33. Remembering that SET33 consists of two locomotive types, type 1 and type 2, there are five possible type sets: 'TYPE 1 ONLY'; 'TYPE 2 ONLY'; 'TYPE 1'; 'TYPE 2'; and 'TYPE 1 AND 2'. The costs c_{ij} for SET33 used in the heuristic procedure HAP are the same as those used in the exact method described in section 4.3. By varying the initial value of INTERVAL and the number of values of INTERVAL used in the heuristic HAP it is possible to produce different sets of sequences Ω. For SET33 seven different sets of sequences are created. The results of solving the problem RSPP for each of these type sets is given in table 4.6.3.i. The key to the table is:

'SET'	-	is a name given to the set of sequences being considered.
'INTERVAL'	-	gives the number of different values of the parameter INTERVAL used to construct the sequences.
'SEQ'	-	shows how many sequences there are in the set of sequences being considered.
'OBJ'	-	gives the value of the objective function for the optimal solution to the problem RSPP.

'Tk' - indicates how many locomotives of type k are used.

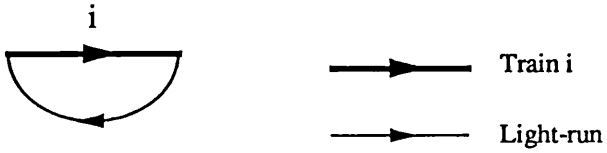
'INT' - indicates whether or not the solution of RSPP is integer.

Recall that there are 12 type 1 locomotives and 6 type 2 locomotives.

SET	INTERVAL	SEQ	OBJ	T1	T2	INT
SEQ1	2	29	800	12	4	YES
SEQ2	3	66	576.67	11.67	6	NO
SEQ3	4	74	576.67	11.67	6	NO
SEQ4	5	94	570	12	6	NO
SEQ5	5	80	510	12	5	YES
SEQ6	7	135	383.33	12	6	NO
SEQ7	7	186	372.86	12	6	NO

Table 4.6.3.i

For the pairs of results SEQ4/SEQ5 and SEQ6/SEQ7, although in each case the heuristic HAP is executed for the same number of values of INTERVAL, different values of the parameter are used in each instance. Also, in SEQ7 all singletons are included, i.e. all sequences of the type:



where i is a particular train. The singleton sequence is replicated for each locomotive type compatible with train i.

The best result found for data set SET33 was 372.86 light-run minutes using 18 locomotives. It is known from the results found in section 4.4.1 that the optimal integer solution for 18 locomotives is 140 light-run minutes. Nevertheless, these results are encouraging given that only 186 variables are considered in test problem SEQ7. Also, a consequence of the size of problem SEQ7 is that the time to solve SEQ7 to optimality using LAMPS was only 6 seconds of CPU time on an IBM PS/2 Model 55SX.

The important question now is : 'How will this method cope with the larger data sets SET178 and SET189?'

SET178

For the purpose of the exercise of generating sequences, only locomotives types 1,2,3 and 4 are considered in an attempt to reduce the number of type sets considered at this experimental stage. This is possible for SET178 because type 5 is dominated by all other types, i.e. every train compatible with locomotive type 5 is also compatible with at least one of the locomotive types 1,2,3 or 4. There are 29 possible type sets arising from these four locomotive types. For SET178, once the base value of INTERVAL is found for a particular type set, the value of INTERVAL is increased by 60 minutes and heuristic HAP is repeated. The number of repetitions of the heuristic is controlled by specifying a stopping criterion. As INTERVAL is increased with a view to admitting more light-run variables into the problem AP and thereby reducing the value of the optimal solution, the optimal solution of the AP should be reduced at each iteration. However, there is an upper limit on the value of INTERVAL, as beyond a certain threshold value (characteristic of the type set considered) all possible connections are included in the problem AP. Therefore, for each of the type sets INTERVAL is only increased until there is no change in the value of the optimal solution to AP for four repetitions of HAP. Obviously, the value

of INTERVAL at the time when the iterations are suspended does not necessarily coincide with the threshold value for the type set as further reductions in the value of the optimal solution may still be possible. As with the set of sequences SEQ7 for SET33 all singletons are included in the set of sequences Ω and the resulting set contained 9,285 sequences.

The results of solving the problem RSPP for the set of sequences Ω are presented in table 4.6.3.ii. The problem RSPP is solved four times and each time the limit on the availability of the locomotive types varied. The variations are listed as test problems T178/1, T178/2, T178/3 and T178/4. The availabilities of the locomotive types for test T178/1 are approximated from the data for SET189. The key to the table is:

'TEST'	-	gives the name of the test problem.
'OBJ'	-	gives the value of the objective function for the optimal solution to the problem RSPP.
'Tk'	-	indicates how many locomotives of type k are used. Two figures a/b are given, where a is the number of locomotives of type k used and b is the number of locomotives of type k available.
'INT'	-	indicates whether or not the solution of RSPP is integer.
'SUM OF INF'	-	gives the sum of the infeasibilities if the solution to RSPP is infeasible.

TEST	OBJ	T1	T2	T3	T4	INT	SUM OF INF.
T178/1	INF	-/14	-/22	-/20	-/32	-	12.5435
T178/2	5000.05	27/27	22/22	20/20	32/32	NO	-
T178/3	980	100/100	10/100	2/100	6/100	NO	-
T178/4	980	119/1000	6/1000	0/1000	1/1000	YES	-

Table 4.6.3.ii

LAMPS reported the problem T178/1 was infeasible due to a deficit of 12.5435 in the number of type 1 locomotives available. So, in T178/2 the number of locomotives of type 1 available is increased by 13 (=12.5345 rounded up) and a feasible solution found which used a total of 101 locomotives. For T178/3 the number of locomotives available is 100 for each locomotive type. An optimal feasible solution was found and only the constraint on the availability of locomotive type 1 was effective. Now compare the result of test T178/4 with the results obtained from the reduction of variables exercise discussed earlier. For the reduction of variables exercise there is an unlimited number of locomotives available, but for strategies 1 and 2 the solution found was infeasible. For T178/4 the level of locomotive availability for each of the locomotive types effectively removes the constraint on locomotive availability, and an optimal feasible solution of 980 was found.

It was felt that, compared with the reduction of variables results, these results suggested a way forward for the problem. So, it was decided to apply this method to the real data set SET189.

SET189

The data set SET189 has 10 locomotive types which implies a maximum possible 1,275 type sets, though some of these may be empty sets. Given the number

of type sets the policy this time is to run the heuristic HAP for only two values of INTERVAL, namely, 360 minutes and 720 minutes. The set of sequences so produced contains 16,051 sequences including all singleton sequences. Based on these sequences, the problem RSPP is solved for two test problems which have different limits on the number of locomotives available. In test T189/1 the number of locomotives available for each type is:

TYPE	No. AVAILABLE
1	14
2	6
3	16
4	6
5	14
6	12
7	20
8	5
9	2
10	4

These are the values as given for the data set SET189. In test T189/2 an upper limit 100 is imposed on each locomotive type.

The results for these test sets are given in table 4.6.3.iii. The key to the table is as follows:

'TEST' - gives the name of the test problem.

'OBJ' - gives the value of the objective function for the optimal solution to the problem RSPP.

'TOTAL LOCO' - gives the total number of locomotives used.

'INT' - indicates whether or not the solution of RSPP is integer.

'SUM OF INF' - gives the sum of the infeasibilities if the solution to RSPP is infeasible.

TEST	OBJ	TOTAL LOCO.	INT	SUM OF INF.
T189/1	INF	-	-	13.3367
T189/2	1260.00	118	YES	-

Table 4.6.3.iii

In the case of test T189/2, none of the constraints on the availability of a locomotive type were effective and the optimal integer solution for this limited set of sequences was found. For test T189/1 the solution found was infeasible. It may be possible to run HAP for additional values of INTERVAL in the hope of finding additional sequences which could work the timetable for the given locomotive availability. However, such a strategy could give rise to an undesirable explosion in the number of sequences generated with no guarantee of reaching feasibility.

It was for this reason that the direction taken at this stage is to use a column generation technique to achieve feasibility as a first goal and optimality as a second goal. However, before discussion of the details and application of the column generation technique there are a few more words to be said about the generation of sequences heuristic.

4.6.4 Further Development of the Heuristic Procedure

Heuristic ModHAP

The starting set of sequences for the column generation technique are provided by the heuristic HAP and the dual values obtained from the solution of the problem RSPP are then used to generate new sequences. It is therefore a worthwhile exercise to find a good starting set of sequences. To this end, the heuristic HAP is modified. It was the inclusion of singletons in the set of sequences which suggested a way of improving the heuristic HAP. The modified version of HAP is referred to as ModHAP.

The heuristic HAP increases the value of INTERVAL until a base value is found which corresponds to a value of INTERVAL for which all trains in the type set can be scheduled. That is, if the value of INTERVAL for the type set χ is greater than or equal to the base value for type set χ, for each train $i \in \chi$ there exists at least one train $j \in \chi$ such that $c_{ij} < \infty$ and a finite solution to the problem AP for type set χ exists. This ensures that, as all trains in the data set being considered are included in at least one type set, every train will be represented in at least one of the schedules of the set Ω. Unfortunately, this can also mean that INTERVAL takes a high value and this is not desirable as the connection times entering into the sequences created may be large.

In the modified version ModHAP all possible singletons of each train i, one for each compatible locomotive type, are included in the set of sequences Ω at the outset to ensure that each train is included in at least one sequence. Then in the cost matrix c_{ij} of the problem AP c_{ii} equals light-run time from end location i to start location i and, if there does not exist a train j ($j \neq i$) such that $t_{ij} \leq$ INTERVAL then as before $c_{ij} = \infty$. This construction means that there now exists a finite solution to AP for all values of INTERVAL as no matter how low the value of INTERVAL is the

assignment $i \rightarrow i$ will always be possible. The solutions to AP can include both schedules of length ≥ 2 and singleton schedules. However, when using the heuristic ModHAP as all singletons are automatically included in the set Ω only those schedules of length ≥ 2 are added to the set Ω.

The new heuristic ModHAP is tested on the data set SET189. The values of INTERVAL used to construct the set of sequences are the same as those used for the previous tests on SET189 using the heuristic HAP. The resulting number of sequences was 16,336 including the singletons. The restrictions on the availability of the locomotives in tests T189/1M and T189/2M are the same as those in T189/1 and T189/2, respectively. The results of solving the problem RSPP for the two test problems are given at table 4.6.4.i.

TEST	OBJ	TOTAL LOCO.	INTEGER	SUM OF INF.
T189/1M	INF	-	-	12.1053
T189/2M	770.00	124	YES	-

Table 4.6.4.i

Comparing the test problem T189/1M with T189/1, the results show that the use of ModHAP has led to an improvement in the sum of infeasibilities and this translates to a reduction in the minimum number of locomotives required to achieve feasibility. As with the solution for T189/2, for the solution to T189/2M the upper limit on the availability of each of the locomotive types is not constraining the problem. The optimal solution to T189/2M requires a greater number of locomotives than the corresponding solution T189/2, but this is coupled with a significant reduction in the objective value of the optimum solution. This last result suggests that by accepting smaller values of INTERVAL, the restriction which the value of INTERVAL imposes on the light-run times means that the heuristic ModHAP

generates sequences with smaller light-run times.

Heuristic LOCOST

As already shown by British Rail's method, another means of generating good sequences is to solve the assignment problem and control the number of locomotives used by placing a cost on the use of a locomotive. This is done using a heuristic LOCOST and the results compared with ModHAP.

The formulation of the assignment problem for LOCOST is similar to AP, but the objective function has been changed to account for the cost of using a locomotive. The formulation for a type set χ is as follows:

APL $\quad$ MIN $\Sigma_{i\in\chi}\Sigma_{j\in\chi} (c_{ij} + \mu m_i + \mu M_{ij})a_{ij}$

s.t.

$$\Sigma_{j\in\chi} a_{ij} = 1 \qquad \forall\, i\in\chi \quad \textbf{APL1}$$

$$\Sigma_{j\in\chi} a_{ji} = 1 \qquad \forall\, i\in\chi \quad \textbf{APL2}$$

$$a_{ij} \in \{0,1\} \qquad \forall\, i,j\in\chi \quad \textbf{APL3}$$

The zero-one variables are:

a_{ij} - 1 if the locomotive performs a light-run from the end location of train i to the start location of train j,

\- 0 otherwise.

The data are:

- c_{ij} - the light-run time from the end location of train i to the start location of train j;
- m_i - the number of times a locomotive crosses midnight while working train i;
- M_{ij} - the number of time a locomotive crosses midnight while performing the light-run from the end location of train i to the start location of train j;
- μ - the marginal cost of a locomotive.

χ is the type set being considered, where χ is a subset of the trains in the data set considered.

The matrix structure of the problem APL is unimodular, therefore the condition that a_{ij} is a zero-one variable can be relaxed.

For heuristic LOCOST all possible connections are allowed, i.e. the value of c_{ij} is not altered as is the case with ModHAP.

The problem APL is solved for each type set associated with the data set SET189. Two values of the parameter μ are used, $\mu=150$ and $\mu=300$. These values of μ were not chosen arbitrarily. The value of 150 coincides with the marginal cost of a locomotive used in British Rail's heuristic to solve the stock diagramming problem for SET189. As before, all singletons are added to the set of sequences Ω prior to the use of heuristic LOCOST. The final number of sequences in the set Ω is 8,978. This is significantly fewer than the number created when using ModHAP. Using these sequences two test problems are solved. For tests T189/1L and T189/2L the limits on the number of locomotives used are the same as those used for T189/1

and T189/2, respectively. The key to the columns of table 4.6.4.ii is the same as for table 4.6.4.i.

TEST	OBJ	TOTAL LOCO.	INTEGER	SUM OF INF.
T189/1L	INF	-	-	9.2942
T189/2L	1200.00	106	YES	-

Table 4.6.4.ii

Comparing these results with those found using ModHAP. It is clear that heuristic LOCOST generates schedules which use fewer locomotives to work the trains. The sum of infeasibilities has fallen from 12.1053 for test T189/1M to 9.2942 in test T189/1L, a drop which represents a decreased requirement of almost 3 locomotives. Similarly, for the second sets of tests the difference in the number of locomotives required is notable. However, the restrictions imposed on connection time mean that ModHAP out-performs LOCOST when comparisons are made of light-run times incurred for T189/2M and T189/2L.

Clearly, it might be possible to improve the results obtained for both methods by using alternative values of INTERVAL or μ. These alternative values could be selected at random or a method could be developed to test the most promising values, values which could be dependent or independent of the type sets. Also it is possible to generate more sequences by using additional values of the parameters INTERVAL and μ as only two values are used in the above tests. A further possibility would be to combine the two procedures ModHAP and LOCOST by adding the sequences generated by both of the heuristics to the set Ω. These extensions to the procedure for generating sequences are not considered at this stage. For now, given that the next step is to use a column generation technique to find optimal solutions to the problem RSPP, a reasonably good set of sequences will suffice. It is stated at the time of

implementing the column generation procedure which heuristic procedure is used to produce the initial set of sequences Ω.

CHAPTER FIVE

COLUMN GENERATION

5.1 Introduction

This chapter introduces the column generation technique. Section 5.3 reviews the use of this method in a theoretical and a practical context. To conclude, a description of how the technique is applied to the stock diagramming problem dealt with in this thesis is given.

5.2 Column Generation

The seminal paper on the column generation technique is by Dantzig and Wolfe [20] and dates back to 1959. The technique provides a means for the efficient computation of large scale mathematical programming problems. The method involves decomposing a linear problem into independent linear subproblems representing its several parts and a coupling problem, the master problem, which links the subproblems by means of coupling (common) equations. The master problem is obtained from the subproblems by means of linear transformations. In general, the master problem contains only a few more rows than there are coupling equations in the original problem, but a large number of columns corresponding to the extreme points and extreme rays of the polytopes associated with the subproblems. In practice though the master problem need only contain a subset of the set of possible columns. This is because the original problem is solved to optimality by generating columns and introducing them to the master problem as required, rather than tabulating all possible columns at the outset. This technique is referred to as "column generation". It is this attribute of the column generation technique which makes it an attractive procedure for the solution of large scale problems where the number of variables in the original problem is prohibitively large.

The column generation procedure involves alternating between the subproblems and the master problem. The subproblems receive a set of dual values associated with the coupling equations. The subproblems are resolved using the new dual values and the solutions are sent back to the master problem. The master problem then evaluates an optimal combination of the new solutions along with the old ones, and sends a revised set of dual values back to the subproblems. This iterative procedure continues until an optimality test is passed, or in economic terms until an 'agreement' is achieved between the master problem and the subproblems. This iterative process is finite.

Mathematically, the general principle is as follows. Consider a general linear programming problem.

LP

$$\text{MIN } cx + dy \qquad \textbf{LP1}$$

s.t.

$$\bar{A}x + \bar{B}y = b \qquad \textbf{LP2}$$

$$Ax = b_1 \qquad \textbf{LP3}$$

$$By = b_2 \qquad \textbf{LP4}$$

$$x \geq 0 \qquad \textbf{LP5}$$

$$y \geq 0 \qquad \textbf{LP6}$$

Where $\bar{A}$, $\bar{B}$, A and B are matrices. b, b_1 and b_2 are vectors with m_0, m_1 and m_2 components, respectively. x and y are vectors for which values have to be found so as to minimize expression LP1.

Now consider the following theorems. Consider a system of linear equations $Ex=\beta$. Initially, suppose that the polyhedron $\wp=\{x|Ex=\beta,x\geq 0)$ is non-empty and bounded.

Theorem 5.2.1 (Lasdon [42])

Let $\wp=\{x|Ex=\beta,\ x\geq 0\}$ be non-empty and bounded, and let x_i, $i=1,\ldots,r$ be its extreme points. Then any element $x\in\wp$ may be written as

$$x=\Sigma_i\, x_i\lambda_i, \qquad i=1,\ldots,r,$$
$$\Sigma_i\, \lambda_i=1, \qquad \lambda_i\geq 0$$

The importance of this theorem in the theory of the simplex method lies in the following theorem and its implication.

Theorem 5.2.2 (Dantzig [18])

A basic feasible solution corresponds to an extreme point in the convex set of feasible solutions.

Conversely, the set of extreme points corresponding to feasible solutions of $Ex=\beta$ constitutes a convex set. Therefore, any feasible point of the simplex can be represented by extreme points by the above two theorems. In the above theorems, it is assumed that $\wp$ was bounded. Is this assumption necessary, or is it possible to represent the simplex of an unbounded problem in a similar way? To see that it is, first consider a theorem from Cohn [15].

Theorem 5.2.3

Let

$$Ex=\beta \qquad C1$$

be a system of m equations in n unknowns. Then the following assertions are equivalent:

i) *the system C1 has a solution,*

ii) *the augmented matrix (E,β) has the same rank as E,*

iii) *for any vector u of length m, $uE=0 \Rightarrow u\beta=0$.*

When a solution exists, the general solution of C1 has the form $x_0 + x'$, where x' is a particular solution of C1 and x_0 is the general solution of the associated homogeneous system

$$Ex=0 \qquad C2$$

Therefore, by theorem 5.2.3 a complete solution of the system $Ex=\beta$ incorporates the homogeneous solutions. The specification of a general solution is of import if $\wp=\{x \mid Ex=\beta,\ x\geq 0\}$ is unbounded. In this case the simplex method not only locates the extreme points of the polyhedron given by $\wp$, but also the homogeneous solutions of $\wp$ and these correspond to extreme rays of the unbounded polyhedron $\wp$. Theorem 5.2.1 can therefore be extended to encompass the situation where $\wp$ is unbounded.

Theorem 5.2.4 (Lasdon [42])

Let $\wp=\{x|Ex=\beta, x\geq 0\}$ be non-empty. Then a point, x, is in $\wp$ if and only if it can be written as a convex combination of extreme points of $\wp$ plus a nonnegative linear combination of extreme rays (homogeneous solutions) of $\wp$; i.e.,

$$x=\Sigma_i x_i\lambda_i$$

where

$$\Sigma_i \delta_i\lambda_i=1, \qquad \lambda_i\geq 0$$

and

$\delta_i =$ *1 if x_i is an extreme point of $\wp$, or*
0 if x_i is an extreme ray of $\wp$.

So, returning to the problem LP. For the sets of equations given by LP3 and LP4 theorem 5.2.4 says that if

$$\wp_1 = \{ x \mid Ax = b_1, x \geq 0\}$$
$$\wp_2 = \{ y \mid By = b_2, y \geq 0\}$$

then any element of $\wp_1$ can be written as

$$x=\Sigma_i x_i\lambda_i \qquad (5.2.1)$$

where

$$\Sigma_i \delta_i\lambda_i=1 \qquad (5.2.2)$$

and x_i is an extreme point/extreme ray of $\wp_1$, with

$\delta_i =$ 1 if x_i is an extreme point of $\wp_1$, or
0 if x_i is an extreme ray of $\wp_1$.

Similarly,

$$y=\Sigma_j\ y_j\mu_j \tag{5.2.3}$$

where

$$\Sigma_j\ \gamma_j\mu_j=1 \tag{5.2.4}$$

and y_j is an extreme point/extreme ray of $\wp_2$, with

$\gamma_j =$ 1 if y_j is an extreme point of $\wp_2$, or
0 if y_j is an extreme ray of $\wp_2$.

Viewing problem LP as: choose from all solutions of LP3 to LP6 those which satisfy LP2 and minimize LP1. All solutions of LP3 to LP6 can be represented by appropriate values of λ_i and μ_j in expressions (5.2.1) and (5.2.3), and constraint LP2 can then be enforced by substituting (5.2.1) and (5.2.3) into LP2 to obtain:

$$\Sigma_i\ (\bar{A}x_i)\lambda_i + \Sigma_j\ (\bar{B}y_j)\mu_j = b \tag{5.2.5}$$

Performing the same linear transformation for LP1 gives:

$$\Sigma_i\ (cx_i)\lambda_i + \Sigma_j\ (dy_j)\mu_j \tag{5.2.6}$$

Now, defining

$$P_i = \bar{A}x_i,$$
$$R_j = \bar{B}y_j,$$
$$cx_i = f_i,$$
$$dy_j = g_j,$$

and substituting in expressions (5.2.2), (5.2.4), (5.2.5) and (5.2.6) gives a linear problem, which is equivalent to LP, expressed in terms of the variables λ_i and μ_j.

MP

$$\text{MIN } \Sigma_i f_i\lambda_i + \Sigma_j g_j\mu_j \qquad \textbf{MP1}$$

s.t.

$$\Sigma_i P_i\lambda_i + \Sigma_j R_j\mu_j = b \qquad \textbf{MP2}$$

$$\Sigma_i \delta_i\lambda_i = 1 \qquad \textbf{MP3}$$

$$\Sigma_j \gamma_j\mu_j = 1 \qquad \textbf{MP4}$$

$$\lambda_i \geq 0 \qquad \textbf{MP5}$$

$$\mu_j \geq 0 \qquad \textbf{MP6}$$

The problem MP is the master problem. The matrix associated with the problem MP has only m+2 rows and as many columns as the polyhedra $\wp_1$ and $\wp_2$ have extreme points and extreme rays. Rather than tabulating all such columns, initially the master problem only contains a subset of the set of possible columns. To see how columns are subsequently added to the master problem, consider the dual vector $\pi=(\pi_0,\pi_1,\pi_2)$ obtained by solving the master problem based on a subset of the possible columns. π_0 corresponds to constraint set MP2 and scalars π_1 and π_2 correspond to constraint sets MP3 and MP4. Pricing out variables λ_i and μ_j gives,

$$\bar{f}_i = f_i - \pi_0 P_i - \pi_1\delta_i \qquad (5.2.7)$$

$$\bar{g}_j = g_j - \pi_0 R_j - \pi_2\gamma_j \qquad (5.2.8)$$

The usual simplex criterion will seek to find a variable λ_s or μ_t which represents the greatest violation in the optimality conditions $\bar{f}_i \geq 0$ and $\bar{g}_j \geq 0$. Such a variable is then chosen to enter the basis. The question is: how is the entering variable found?

Back substitution for P_i and R_j in (5.2.7) and (5.2.8) yields expressions in terms of the original variables, i.e.:

$$\bar{f}_i = (c - \pi_0\bar{A})x_i - \pi_1\delta_i \qquad (5.2.9)$$

$$\bar{g}_j = (d - \pi_0\bar{B})y_j - \pi_2\delta_j \qquad (5.2.10)$$

Without loss of generality, suppose that:

$$\min(\bar{f}_i,\bar{g}_j) = \bar{f}_s = (c - \pi_0\bar{A})x_s - \pi_1\delta_i \qquad (5.2.11)$$

Recalling that x_i is an extreme point/extreme ray of $\wp_1$. If $\wp_1$ is bounded an optimal solution of $Ax = b_1$ will occur at an extreme point. If $\wp_1$ is unbounded, then the simplex method will locate an extreme ray corresponding to a homogeneous solution $Ax = 0$, $x \geq 0$. Therefore (5.2.11) is equivalent to solving the subproblem

SP

$$\text{MIN } (c - \pi_0\bar{A})x \qquad \textbf{SP1}$$

s.t.

$$Ax = b_1 \qquad \textbf{SP2}$$

$$x \geq 0 \qquad \textbf{SP3}$$

The solution x_s of the problem SP gives the next column P_s which enters the basis, and hence the next column which is added to the subset of columns in the master problem. The column P_s is given by:

$$P_s = \begin{Bmatrix} \bar{A}x_s \\ \delta_s \end{Bmatrix}$$

where $\delta_s = 1/0$ depending on whether or not x_s is an extreme point/extreme ray of the polytope $\wp_1$.

Note that in the master problem any point on an extreme ray can be expressed as a nonnegative multiple of the point on the extreme ray located by the simplex method, hence the stipulation that $\delta_i = 0$ if x_i is a homogeneous solution of $Ax = b_1$.

If $\bar{f}_s < 0$ then, barring degeneracy, the inclusion of P_s in the basis of the master

problem will result in a decrease in the value of the objective. If $\bar{f}_s=0$ then there are no columns which will reduce the value of the objective function further and the procedure terminates.

It is clear from the above discussion, that although the column generated by the subproblem achieves the maximum violation of the optimality conditions, any column with a negative relative cost could be added to the basis with a promise of a decrease in the value of the objective function. Therefore, an extension is to locate two columns corresponding to $\min(\bar{f}_i)<0$ and $\min(\bar{g}_j)<0$, or even all columns with $\bar{f}_i<0$ and $\bar{g}_j<0$.

The assertion made at the beginning of this chapter that this procedure is finite is based on the following theorem.

Theorem 5.2.5 (Dantzig [18])

Only a finite number of iterations of the simplex algorithm is required if each basic feasible solution is improved by introducing into the basis either an extreme point P^*_t, *where* $P^*_t=\{\bar{A}x_t\}$ *with* x_t *a solution of* $Ax=b_1$, *chosen so that*

$\{1\}$

$$\pi P^*_t = \min(\pi P_j) < 0$$

where π *are the simplex multipliers of the basis, and* P_t *is defined in a similar way to* P^*_t, *or by introducing into the basis* $\bar{P}^*_t$, *where* $\bar{P}^*_t=\{\bar{A}x_t\}$, *and* x_t *is a*

$\{0\}$

homogeneous solution of $Ax_t=0$, *from a finite set such that* $\pi\bar{P}^*_t<0$.

A similar theorem can be expressed for the set $\wp_2$.

A simple outline of the proof is given. It is clear that in the bounded case there is only a finite number of distinct extreme points of the subproblem polytopes, and therefore if the algorithm were to continue indefinitely it could only do so by repeating the set of variables defining the extreme point. In the unbounded case, when the simplex method obtains a homogeneous solution, the vector x_i found corresponds to a point on the extreme ray located by the method. The number of such vectors, ignoring multiples, is finite corresponding to the finite number of extreme rays of the subproblem polytopes. Hence the set of P_i is finite.

5.3 Applications of the Column Generation Technique

Dantzig and Wolfe [20] acknowledge a paper presented by Ford and Fulkerson [33] as the inspiration behind the development of the column generation technique. Ford and Fulkerson propose a method for solving the maximal multicommodity network flow problem. For this problem there are K subproblems, one for each of the K commodities in the network. At the time of writing Ford and Fulkerson stated that: "Straightforward application of the simplex method to such problems is usually not feasible, since even small networks may generate linear programs which are too large for present machine capacity." Since this work the column generation technique has been applied to a number of multicommodity flow problems see: Bellmore, Bennington and Lubore [5]; Riberio and Soumis [63]; Appelgren [1]; and, Ferland and Michelon [29].

Although the strength of the column generation technique lies in the potential for a large reduction in the size of the problems which have to be worked with, the situation can be further simplified by means of an intelligent decomposition which constructs subproblems that are easily solved. For examples of this see: Desrosiers, Soumis and Desrochers [24]; Riberio and Soumis [63]; and, Ford and Fulkerson [33].

For these applications of the column generation technique the subproblems are shortest path problems. Also see Bellmore, Bennington and Lubore [5] where, by means of a simple consideration of the optimality conditions, the subproblems could be solved directly as minimal cost network flow problems.

5.4 Review

Before discussing the particular decomposition applied in this thesis, some of the aforementioned work warrants further discussion.

Ford and Fulkerson [33] formulate the maximal multicommodity network flow problem in such a way that the matrix of the linear program is the incidence matrix of arcs vs. chains joining sources to sinks for the various commodities. So, if $A_1,\ldots,A_m$ are the set of arcs in the network and $C_1,\ldots,C_n$ the set of chains, including sources and sinks for each commodity, then $A=(a_{rs})$ is the incidence matrix where:

a_{rs} - 1 if C_s contains arc A_r,
- 0 otherwise.

The linear programming formulation which follows is:

MMNF

$$\text{MAX } \Sigma_{s=1}^{n} x_s \qquad \text{MMNF1}$$

s.t.

$$\Sigma_{s=1}^{n} a_{rs}x_s + x_{n+r} = b_r \qquad r=1,\ldots,m \qquad \text{MMNF2}$$

$$x_s \geq 0 \qquad s=1,\ldots,n+r \qquad \text{MMNF3}$$

where b_r is the flow capacity of A_r. x_s, s=1,...n, is the amount of flow of commodity along C_s and x_s, s=n+1,...,n+r, is the spare capacity on arc A_r. The problem MMNF corresponds to the master problem in the theory of column generation. Not all possible columns of A, i.e. the chains, are enumerated at the outset. Instead, they are generated from the subproblems using an $O(n^2)$ polynomial time shortest path algorithm, made acyclic by introducing source and sink nodes for the commodities. The subproblems ensure conservation of flow of commodities and are unbounded. The solutions to the subproblems are extreme rays and therefore do not have to satisfy a convex combination constraint in the master problem. In the application of the procedure a bounded subproblem is obtained by only admitting nonnegative dual values, thereby eliminating the possibility of producing negative cycles and ensuring that the solution of the subproblem only admits 0-1 elements to the matrix A.

Ford and Fulkerson could not report any computational experience with the method. However, they assert that: "It would certainly be more practicable...than straightforward application of the simplex method to a node-arc formulation of the problem..."

A more recent multicommodity flow problem solved using the column generation technique is the multiple-depot vehicle scheduling problem presented by Riberio and Soumis [63]. Simply stated, the problem considers a set of n trips $T_1,...,T_n$ with pre-defined starting and ending times. There also exists a set of m depots $D_1,...,D_m$ each containing $r_1,...,r_m$ vehicles, respectively. If τ_{ij} is the travel time from the end location of trip i to the start location of trip j, the problem is to find a feasible assignment of trips to vehicles so that: each trip is completed; no more than the available vehicles are used; and, the total cost is minimized. A cost is incurred when an intertrip is made, i.e. the cost of moving a vehicle when it is not performing a trip.

Mathematically, the problem can be expressed as:

MDVSP

$$\text{MIN } \Sigma_k \Sigma_i \Sigma_j c_{ij} x_{ij}^k$$

s.t.

$\Sigma_k \Sigma_i x_{ij}^k = 1$	$j=1,..,n+m$	**MDVSP1**
$\Sigma_i x_{ij}^k - \Sigma_i x_{ji}^k = 0$	$k=1,...,K,$	
	$j=1,...n+m$	**MDVSP2**
$\Sigma_j x_{n+k,j}^k \leq r_k$	$k=1,...,K$	**MDVSP3**
$x_{ij}^k \geq 0$	$k=1,...,K,$	
	$j=1,...n+m$	**MDVSP4**
x_{ij}^k integer	$k=1,...,K,$	
	$j=1,...n+m$	**MDVSP5**

where x_{ij}^k is the flow of vehicle type k from end location of trip T_i to start location of trip Tj. c_{ij} is the cost of the intertrip and is independent of the vehicle type. Note that $i,j=n+1,...,n+m$ represent the depots.

The decomposition then runs as follows:

D_MDVSP

$$\text{MIN } \Sigma_k \Sigma_{p\in\Omega(k)} c_p y_p$$

s.t.

$\Sigma_k \Sigma_{p\in\Omega(k)} a_{jp} y_p = 1$	$j=1,...,n$	**D_MDVSP1**
$\Sigma_{p\in\Omega(k)} y_p \leq r_k$	$k=1,...,K$	**D_MDVSP2**
$y_p \in \{0,1\}$		**D_MDVSP3**

where $\Omega(k)$ is the set of paths leaving and returning to the depot D_k. If p is a path in $\Omega(k)$, then c_p is the cost of traversing the path p and $a_{jp}=1$ if p covers trip T_j. Associating a variable y_p with each path p, the above formulation states that the master problem is to choose among the available paths so as to satisfy the conditions of the original problem.

There are K subproblems, one for each depot. The subproblems seek to satisfy the conservation of flow constraints, constraint set MDVSP2, and the solutions to the subproblems are either zero or unbounded as a result of the homogeneity of the constraint set. As a result, there is no convexity constraint in D_MDVSP. An unbounded solution to the kth subproblem is a circuit through depot D_k. In practice, extreme ray solutions for the subproblems are found by solving unconstrained shortest path problems based on the network underpinning the subproblems. The shortest path problems are made acyclic by the addition of source and sink nodes for the depot.

Riberio and Soumis report the solution of problems with up to 300 trips and 6 depots. They state that the "...final set of test problems is formed by instances four to five times larger than the largest problems exactly solved so far."

Among the papers which seek to solve multicommodity flow problems using column generation is that of Appelgren [1]. This paper is worth mentioning as it opens the discussion on what is known as a 'restricted' master problem. Essentially, the master problem can be represented at two extremes. One extreme is the 'full' master problem for which all possible solutions of the subproblems are represented. The other extreme is the restricted master problem, obtained from the full master problem by dropping all columns except those in the basis and the new columns about to be introduced. This restriction on the number of columns allowed to exist in the master problem is of use when the problem could expand by an indefinite number of columns. Even though a column is dropped from the active set of columns, it still exists in the subproblems and may be generated again if required.

In practice, Dantzig [18] suggests three variants for controlling the number of columns in the restricted master problem:

i) The restricted master problem is augmented by each new column, but each column that drops out of the basis is dropped from the current restricted master.

ii) The restricted master problem is augmented by more and more columns and those dropping out of the basis are retained as supplementary columns.

iii) The restricted master problem is augmented by more and more columns, and those dropping out of the basis are retained up to the available memory capacity within the electronic computer; at this point, columns that price out most positive are dropped.

In Appelgren's restricted form of the master problem, all columns generated by the subproblems are added to the restricted master but each column that subsequently drops out of the basis is dropped from the restricted master. So, Appelgren allows room for the columns in the basis plus as many columns as there are subproblems. This scheme effectively reduces the size of the master problem. The drawback, however, is that previously discarded columns may have to be regenerated by the subproblems as the simplex method requires them.

5.5 Decomposition for the Stock Diagramming Problem

The decomposition applied to the problem covered in this thesis places the conservation of flow constraints in the subproblems. The requirement that each train is worked exactly once is held in the master problem along with the constraint limiting the number of locomotives of each type used. The conservation of flow

constraints can be represented by K independent subsystems of linear equations, one for each vehicle type k. There are, therefore, K independent subproblems. Recalling the original problem SD first, the formulations which follow describe this decomposition.

Let the set of trains be $\tau=\{i \mid i \text{ is the } i^{th} \text{ train in the timetable}\}$. Let the set of locomotive types be $\Lambda=\{k \mid k \text{ is the number of the locomotive type}\}$. Then define the set P(k) as $P(k)=\{i \mid \text{train } i \text{ is compatible with locomotive type } k\}$.

SD

$$\text{MIN } \Sigma_{i\in\tau}\Sigma_{j\in\tau}\Sigma_{k\in\Lambda}\, c_{ij}\, x_{ij}^{k}$$

s.t.

$$\Sigma_{j\in P(k)}\, x_{ij}^{k} = y_i^k \qquad \forall\, i\in P(k),\ k\in\Lambda \qquad \textbf{SD1}$$

$$\Sigma_{j\in P(k)}\, x_{ji}^{k} = y_i^k \qquad \forall\, i\in P(k),\ k\in\Lambda \qquad \textbf{SD2}$$

$$\Sigma_{k\in\Lambda}\, y_i^k = 1 \qquad \forall\, i\in\tau \qquad \textbf{SD3}$$

$$\Sigma_{i\in\tau}\Sigma_{j\in\tau}\,(m_i y_i^k + M_{ij}x_{ij}^k) \le U_k \qquad \forall\, k\in\Lambda \qquad \textbf{SD4}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, i,j\in P(k),\ k\in\Lambda \qquad \textbf{SD5}$$

$$y_i^k \in \{0,1\} \qquad \forall\, i\in P(k),\ k\in\Lambda \qquad \textbf{SD6}$$

where the zero-one variables are:

x_{ij}^k - 1 if train j follows train i on locomotive type k,
- 0 otherwise;

y_i^k - 1 if train i is worked by locomotive type k,
0 otherwise.

The data elements are:

- c_{ij} - cost of performing the light-run from the end location of train i to the start location of train j;
- m_i - number of times a locomotive crosses midnight while working train i;
- M_{ij} - number of times a locomotive crosses midnight whilst moving from end location i to start location j;
- U_k - the number of locomotives of type k available.

If Ω is the set of schedules satisfying constraint sets SD3 and SD4 in SD, then the master problem can be written as a set-partitioning problem with the added constraints on locomotive availability. The formulation is given as:

SPP

$$\text{MIN } \Sigma_{r\in\Omega} c_r s_r$$

s.t.

$$\Sigma_{r\in\Omega} \delta_{ri} s_r = 1 \qquad \forall\, i\in\tau \qquad \textbf{SPP1}$$

$$\Sigma_{r\in\Omega} L_{rk} s_r \le U_k \qquad \forall\, k\in\Lambda \qquad \textbf{SPP2}$$

$$s_r \in \{0,1\} \qquad \forall\, r\in\Omega \qquad \textbf{SPP3}$$

where the zero-one variables are:

- s_r - 1 if sequence r is used,
- 0 otherwise.

The data elements are:

c_r - total light-run cost of working sequence r;

δ_{ri} - 1 if train i is included in sequence r,

- 0 otherwise;

L_{rk} - number of times locomotive type k crosses midnight whilst working sequence r;

U_k - number of locomotives of type k available.

The dual problem associated with SPP is given by:

DSPP

$$\text{MAX } \Sigma_{i\in\tau} z_i + \Sigma_{k\in\Lambda} U_k w_k$$

s.t

$$\delta_{ri} z_i + L_{rk} w_k \leq c_r \qquad \forall\, r\in\Omega \text{ DSPP1}$$

$$z_i \text{ free variable} \qquad \forall\, i\in\tau \text{ DSPP2}$$

$$w_k \leq 0 \qquad \forall\, k\in\Lambda \text{ DSPP3}$$

The variables are:

z_i - the dual values associated with constraint set SPP1. z_i can take positive or negative values as SPP1 is an equality constraint.

w_k - the dual values associated with constraint set SPP2. w_k takes non-positive values as SPP2 is a less than or equal to constraint.

The data are as defined for the problem SPP.

Therefore a sequence violating the optimality conditions has a reduced cost given by:

$$c_r - \delta_{ri} z_i - L_{rk} w_k < 0. \qquad (5.5.1)$$

So the aim is to find a sequence which maximally violates the optimality condition and add this sequence to the master problem. Rewriting (5.5.1) in terms of the variables x_{ij}^k and y_i^k gives the expression

$$c_{ij}x_{ij}^k - z_i y_i^k - w_k(m_i y_i^k + M_{ij}x_{ij}^k) \qquad (5.5.2)$$

Therefore as the maximum violation in the optimality conditions is the minimum value of the expression given by (5.5.2), this minimum value can be found by solving the subproblems given by:

S_k

$$\text{MIN } \Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\ c_{ij}x_{ij}^k - z_i y_i^k - w_k(m_i y_i^k + M_{ij}x_{ij}^k)$$

s.t.

$$\Sigma_{j\in P(k)}\ x_{ij}^k = y_i^k \qquad \forall\ i\in P(k) \quad S_k1$$

$$\Sigma_{j\in P(k)}\ x_{ji}^k = y_i^k \qquad \forall\ i\in P(k) \quad S_k2$$

$$x_{ij}^k \geq 0 \qquad \forall\ i,j\in P(k) \quad S_k3$$

$$y_i^k \geq 0 \qquad \forall\ i\in P(k) \quad S_k4$$

The variables x_{ij}^k and y_i^k are defined in the same manner as in SD except that the restriction that the variables are binary is relaxed. The data elements m_i, M_{ij} and c_{ij} are as defined for SD. The data elements z_i and w_k are the dual values associated with constraint sets SPP1 and SPP2, respectively, and are retrieved from the solution to SPP.

The solution to the master problem does not have to satisfy the convex combination constraint, as the formulation S_k is a system of homogeneous equations. Note also that as the problem S_k is unimodular, therefore, the variables x_{ij}^k and y_i^k only take integer values in an optimal solution to S_k.

The solutions to the above subproblem consist of one or more sequences which when priced out have a zero or negative cost. The solution value is the sum of the reduced costs of the sequences in the solution. As the variables are not restricted to take binary values, the solutions are either zero or unbounded.

The next chapter presents a detailed discussion of the exact nature of the above subproblems. The methods employed to solve the subproblems are compared and criticized. Also, in response to the discussion in this chapter, there is a description of a scheme used to restrict the size of the master problem in the situation where a large number of columns are being passed to the master problem from the subproblems.

CHAPTER SIX

SUBPROBLEMS

6.1 Nature of Subproblem

In chapter five it is stated that the solutions to the subproblems are zero or unbounded. How do these solutions arise and how are they interpreted in terms of a graphical representation? Remembering that a schedule is a cycle with periodicity one day, then the generation of a column with a negative reduced cost consists of finding a negative cost circuit in the network underpinning the subproblem. A minimum cost solution with value zero indicates that all circuits have a non-negative cost. An unbounded solution is analogous to repeated traversals of circuits with negative costs. A feasible solution need not include every train considered by the subproblem, but a non-trivial solution should include at least one such train. Examples of an unbounded solution and of a zero (non-trivial) solution are given below.

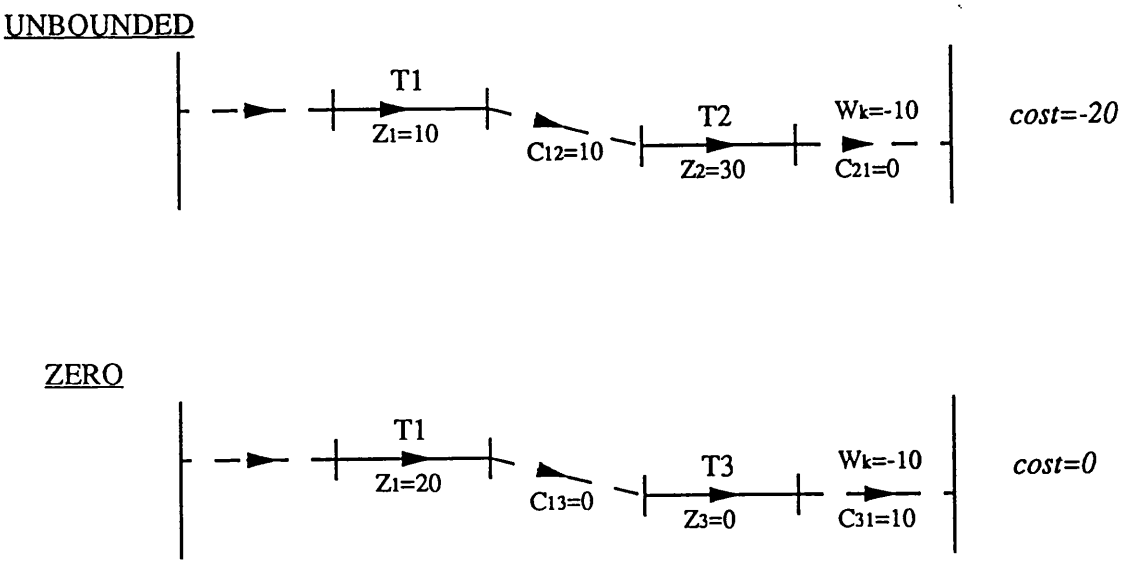


Figure 6.1.i

In the above figure the solid lines represent the train arcs and the dashed lines represent the light-run arcs. 'Tn' indicates train n.

The example of an unbounded solution corresponds to a schedule which includes trains 1 and 2. The zero example corresponds to a schedule which includes trains 1 and 3. The first solution has cost=-20 and the second solution has cost=0.

An unbounded solution corresponds to an extreme ray of the polytope of the subproblem. Consider the formulation S_k introduced in chapter five. By adding the constraint that each train should be worked no more than once, a bounded subproblem is obtained with each extreme point corresponding to an extreme ray of the unbounded subproblem. The new formulation is given below as BS_k. A non-trivial solution now corresponds to a circuit (circuits) in the network which is (are) traversed exactly once.

BS_k

$$\text{MIN } \Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\, c_{ij}x_{ij}^k - z_i y_i^k - w_k(m_i y_i^k + M_{ij}x_{ij}^k)$$

s.t.

$$\Sigma_{j\in P(k)}\, x_{ij}^k = y_i^k \qquad \forall\, i\in P(k)\ \ BS_k1$$

$$\Sigma_{j\in P(k)}\, x_{ji}^k = y_i^k \qquad \forall\, i\in P(k)\ \ BS_k2$$

$$y_i^k \le 1 \qquad \forall\, i\in P(k)\ \ BS_k3$$

$$x_{ij}^k \ge 0 \qquad \forall\, i,j\in P(k)\ \ BS_k4$$

$$y_i^k \ge 0 \qquad \forall\, i\in P(k)\ \ BS_k5$$

In chapter five it is stated that optimal solutions to BS_k are integer as the problem is unimodular. Now that the variables are bounded by a value 1, x_{ij}^k and y_i^k take zero-one values in any optimal solution to BS_k.

6.2 Alternative Formulation

As the subproblem BS_k is a network problem it lends itself to a graphical representation. As before P(k) is the set of trains compatible with locomotive type k. For this set of trains the network used by the locomotives can be described by a graph $G_k=(V_k,A_k)$ where $V_k=P(k)UP(k)$ is a set of nodes representing the start and end points of each of the trains in P(k), and $A_k=P(k)UP(k)xP(k)$ is the set of oriented arcs comprising of $|P(k)|$ train arcs and $|P(k)|^2$ light-run arcs. An example of such a graph for the case where $P(k)=\{1,2,3\}$ is given below.

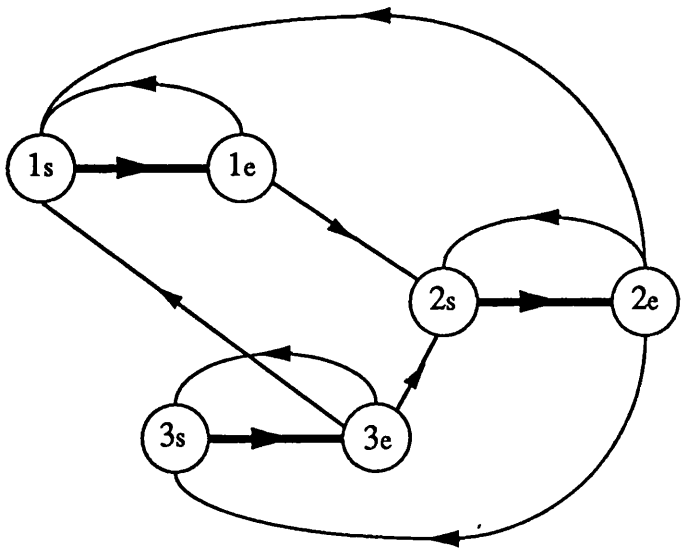


Figure 6.2.i

The labels on the nodes are defined as follows:

i_s = start node for train i;

i_e = end node for train i.

The arcs described by ⟶ are the train arcs and, referring to the formulation BS_k, illustrate the variables y_i^k. Similarly, the arcs described by →⟵ are the light-run arcs which illustrate the variables x_{ij}^k. As the only outward arc for each node i_s is the train arc and the only inward arc for the node i_e is the train arc, it is clear from the argument that follows that nodes i_s and i_e can be merged to i and a simplified graphical representation obtained.

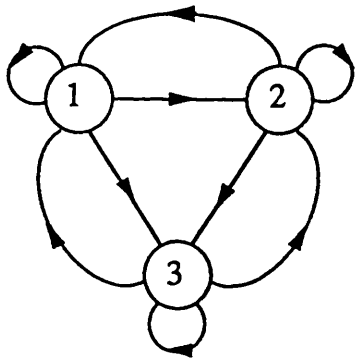


Figure 6.2.ii

The above graph is a complete graph $CG_k=(CV_k,CA_k)$ where $CV_k=P(k)$ is the set of nodes and $CA_k=P(k)xP(k)$ the set of arcs which now excludes the train arcs.

Is this new graphical representation a valid description of the subproblem and how does it translate to the formulation of the subproblem?

For the problem BS_k consider the objective function for a particular train i. If $x_{ij}^k=1$ for some j, by constraint set BS_k1 this implies that $y_i^k=1$. Conversely, if

$y_i^k=1$ this implies that there exists a j such that $x_{ij}^k=1$. So, $x_{ij}^k=1 \leftrightarrow y_i^k=1$. By a similar argument $x_{ij}^k=0 \leftrightarrow y_i^k=0$. Hence, the objective function can be rewritten as:

$$\Sigma_{i\in P(k)}\Sigma_{j\in P(k)} (c_{ij}-z_i-m_iw_k-M_{ij}w_k)x_{ij}^k \qquad (6.2.1)$$

Now consider an alternative definition of the train variables y_i^k and the light-run variables x_{ij}^k.

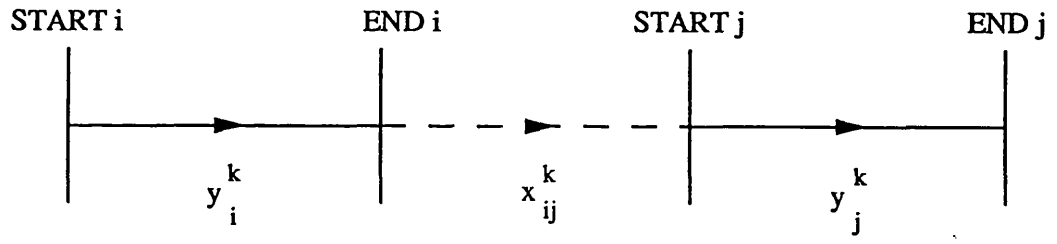


Figure 6.2.iii

As illustrated in the above diagram, y_i^k represents a locomotive type k running from the **start** location of train i to the **end** location of train i, and x_{ij}^k represents a locomotive type k running from the **end** location of train i to the **start** location of train j. Let v_{ij}^k represent a locomotive of type k running from the **start** location of i to the **end** location of i and then to the **start** location of train j. i.e.

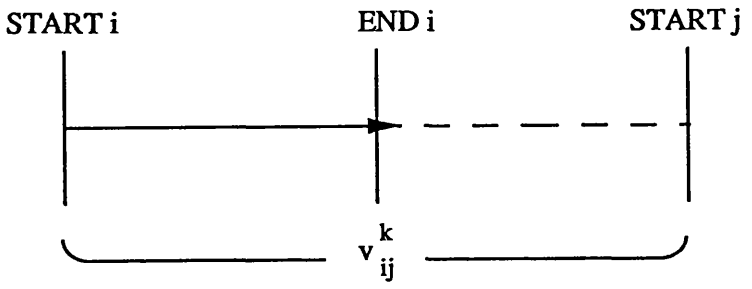


Figure 6.2.iv

As already stated, for a particular train i worked by a locomotive type k then there is a train j such that $x_{ij}^k=1 \leftrightarrow y_i^k=1$, or for every train j $x_{ij}^k=0 \leftrightarrow y_i^k=0$. Therefore, for a train i worked by a locomotive type k, if $x_{ij}^k=1$ for a train j (if $x_{ij}^k=0$ for every train j) then $y_i^k=1$ ($y_i^k=0$) and, by definition, $v_{ij}^k=1$ for train j ($v_{ij}^k=0$ for every train j). Conversely, for a train i worked by a locomotive type k, if $v_{ij}^k=1$ for a train j (if $v_{ij}^k=0$ for every train j) this implies that $x_{ij}^k=1$ for train j ($x_{ij}^k=0$ for every train j) and $y_i^k=1$ ($y_i^k=0$). So $x_{ij}^k=1 \leftrightarrow v_{ij}^k=1$, $x_{ij}^k=0 \leftrightarrow v_{ij}^k=0$, and (6.2.1) can be rewritten as:

$$\Sigma_{i\in P(k)}\Sigma_{j\in P(k)} (c_{ij}-z_i-m_iw_k-M_{ij}w_k)v_{ij}^k \qquad (6.2.2)$$

What about constraint sets BS_k1, BS_k2 and BS_k3? Consider constraint set BS_k1. If $\Sigma_{j\in P(k)}x_{ij}^k=0$ this implies that $y_i^k=0$ and this in turn implies that $\Sigma_{j\in P(k)}v_{ij}^k=0$. Similarly, if $\Sigma_{j\in P(k)}x_{ij}^k=1$ this implies that $y_i^k=1$ and this gives $\Sigma_{j\in P(k)}v_{ij}^k=1$. By symmetry, BS_k2 also holds. Thus:

$$\Sigma_{j\in P(k)}\, x_{ij}^{k} = y_i^{k} = \Sigma_{j\in P(k)}\, v_{ij}^{k} \qquad (6.2.3)$$

and

$$\Sigma_{j\in P(k)}\, x_{ji}^{k} = y_i^{k} = \Sigma_{j\in P(k)}\, v_{ji}^{k} \qquad (6.2.4)$$

Now from BS_k1 and BS_k2:

$$\Sigma_{j\in P(k)}\, x_{ij}^{k} = \Sigma_{j\in P(k)}\, x_{ji}^{k} = y_i^{k} \leq 1 \qquad (6.2.5)$$

(6.2.3), (6.2.4) and (6.2.5) together give:

$$\Sigma_{j\in P(k)}\, v_{ij}^{k} = \Sigma_{j\in P(k)}\, v_{ji}^{k} \qquad (6.2.6)$$

$$\Sigma_{j\in P(k)}\, v_{ij}^{k} \leq 1 \qquad (6.2.7)$$

Constraint set (6.2.6) ensures conservation of flow and constraint set (6.2.7) ensures that the subproblems are bounded. So bringing expressions (6.2.2), (6.2.6) and (6.2.7) together gives a formulation of the subproblem BS_k in terms of the newly defined variables v_{ij}^{k}, i.e.

$\overline{BS_k}$

$$\Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\, (c_{ij} - z_i - m_i w_k - M_{ij} w_k) v_{ij}^{k}$$

s.t.

$$\Sigma_{j\in P(k)}\, v_{ij}^{k} = \Sigma_{j\in P(k)}\, v_{ji}^{k} \qquad \forall\, i\in P(k)\ \overline{BS_k}1$$

$$\Sigma_{j\in P(k)}\, v_{ij}^{k} \leq 1 \qquad \forall\, i\in P(k)\ \overline{BS_k}2$$

$$v_{ij}^{k} \geq 0 \qquad \forall\, i\in P(k)\ \overline{BS_k}3$$

The variables v_{ij}^{k} take zero-one values as the matrix structure of this problem is totally unimodular.

(Clearly the argument given above could equally be applied to formulation SD and a new formulation obtained which replaces the variables x_{ij}^k and y_i^k with the variables v_{ij}^k.)

The next section discusses the methods used to solve the subproblems given by $\overline{BS}_k$.

6.3 Solution of Subproblem Methods

6.3.1 Shortest Path

In a paper on "Routing with Time Windows by Column Generation", Desrosiers et al. [24] solve the subproblems using a specialized version of the Ford-Bellman algorithm which they refer to as the "Shortest Path with Time Windows" (SPTW). The time-dependency of the problem allows for the construction of an acyclic network, in most instances, within which a shortest path must be located. It is stated in chapter five, one column or many columns can be added to the master problem during an iteration of the column generation technique. The SPTW algorithm is a polynomial time algorithm and Desrosiers *et al.* use the speed of the algorithm to advantage. Their strategy is to generate a number of different routes at each iteration of the column generation technique by repetition of the SPTW algorithm. The aim of this scheme is to accelerate progression to an optimal solution.

A shortest path algorithm is used to solve the subproblems generated in the Multiple-depot Vehicle Scheduling Problem described by Riberio and Soumis [63]. The graphs which represent the networks underpinning the subproblems are made acyclic by exploiting the relation that any ordered pair of trips (T_i,T_j) is compatible if and only if $e_i + t_{ij} \leq s_j$, where e_i is the end time of trip T_i, t_{ij} is the travel time between trips T_i and T_j and s_j is the start time of trip T_j. To complete the acyclic graph a copy of the depot node is taken to produce source and sink nodes. A fuller

description of the method of construction of such a graph is given by Levin [47] and the resulting graph referred to as a "schedule map".

It would be desirable to represent the subproblems dealt with in this thesis using acyclic graphs and thereby solve the subproblems in polynomial time using a shortest path problem. Unfortunately, such a simple transformation is not possible for two reasons. The first problem is that there are no time-dependent precedence relationships limiting the order in which trains can appear in a schedule as is true in the above cases. The second complication arises because there are no depots in the stock diagramming problem and therefore no natural starting point for a schedule. Nevertheless, these considerations do not preclude the use of a shortest path algorithm to locate circuits with negative costs in the subproblem networks, as shortest path algorithms can be used to detect negative cycles. The negative cycles correspond to schedules which violate the optimality conditions and hence, new columns to be added to the master problem SPP.

The shortest path algorithm used to detect negative cycles in the graphs associated with the subproblems $\overline{BS}_k$ is the matrix multiplication method. (See Appendix A for a description of the algorithm.) The matrix multiplication method finds shortest paths between all pairs of nodes and so it is not necessary to specify a starting node. If the method is terminated as soon as a negative cycle is found the complexity of the algorithm is $O(n^4)$ (see Appendix A for proof). This gives a method for generating columns in polynomial time. However, there are two drawbacks to this method. The first concerns the amount of computer memory needed to keep a record of the order in which nodes are visited in a path. This information must be kept in order to reconstruct a path which forms a negative cost cycle. The second drawback is the most important disadvantage of this method. As the procedure terminates as soon as a first negative cost cycle is found, the maximum possible violation in the optimality conditions is not necessarily found. This may mean that the resulting drop in the value of the objective function of the master problem is small at each iteration

of the column generation technique. This affects the time taken to find an optimal solution to the original problem. This tendency is illustrated and discussed further in chapter seven, but for now a strategy for dealing with the first drawback is suggested.

In an attempt to overcome storage problems which may be encountered when using the matrix multiplication method for large data sets, an alternative scheme is proposed. As a first step the matrix multiplication method is used to find a node through which a negative cycle passes. However, instead of keeping a record of all the nodes visited along each path, only the node I at the start of the path is recorded. If a negative cycle is found to pass through node I, the matrix multiplication method stops and the Bellman-Ford Algorithm (see Appendix B) is used to find a negative cycle using node I as a starting point. For comparison, for a problem with n nodes the matrix multiplication method stores the information on the nodes in the negative cycles in an nxnxn matrix, whereas given a starting node the Bellman-Ford Algorithm keeps this record in an nxn matrix. The trade-off for a reduction in storage is an increase in computing time as both the matrix multiplication and the Bellman-Ford algorithms are being solved. For the matrix multiplication method, now that only the details of the starting node of a negative cycle are required, the algorithm has complexity $O(n^3 \log n)$ (see Appendix A). The Bellman-Ford Algorithm has complexity $O(n^3)$ (see Appendix B).

An extension to the above method is to make multiple passes through the matrix multiplication method at each major iteration, with the intention of producing a number of columns. One way of doing this is to ban a particular connection in a negative cycle just located by the Bellman-Ford algorithm for the current subproblem. So, if a negative cycle through starting node I has the connection $I \rightarrow J$, introduce a large positive cost on the arc $I \rightarrow J$ before the next execution of the matrix multiplication method. This added loop can be executed until no further negative cycles exists, but it is advisable to control the number of new columns produced by introducing a limit on the number of times the matrix multiplication method is

invoked.

6.3.2 Simplex Method

The second method proposed is to solve the subproblems using the simplex method. This method was not adopted initially due to complexity considerations. In a discussion of complexity of algorithms Papadimitriou and Steiglitz [60] present an argument due to Klee and Minty [40] which asserts that there exists a class of linear programming problems in n variables which require 2^n-1 iterations of simplex to find an optimum. Nemhauser and Wolsey [55] confirm that the simplex algorithm is not a polynomial time algorithm, but also present the simplex algorithm as outstanding evidence against the practice of worst-case analysis of algorithms as the simplex method performs well in many real-world applications. Moreover, recent probabilistic analysis suggests that the *expected* running time of the simplex method is bounded by a polynomial in m, the number of constraints, and n, the number of variables, in the standard linear programming problem.

Given that the running time may prove reasonable and that the simplex method locates a column which corresponds to a maximum violation of the optimality conditions, this method is tested. The hope is that the trade-off between the time taken to solve the subproblems and the quality of the columns located would give rise to a positive payoff in terms of the reduction in the objective function value of the master problem. It is shown in chapter seven that the results are outstanding compared to those achieved using the shortest path method and this led to the simplex method being adopted as the method for solving the subproblems.

Having opted for a method which finds optimal solutions to the subproblems, the only remaining doubts concern the use of a non-polynomial time algorithm. These doubts were finally allayed when it was found that there is a startlingly simple way to achieve the same quality of solutions using a polynomial time procedure.

6.3.3 Assignment Method

For the subproblem given by $\overline{BS_k}$, a non-trivial solution is a set of schedules which includes at least one but not necessarily all trains in the set P(k). Suppose that for a locomotive type k the set P(k)={1,2,3,4,5,6}. Further suppose that the optimal solution to the subproblem $\overline{BS_k}$ is found using the simplex method, and that the set of schedules corresponding to this solution do not include train 6. This solution is illustrated below using the complete graph CG_k associated with this subproblem.

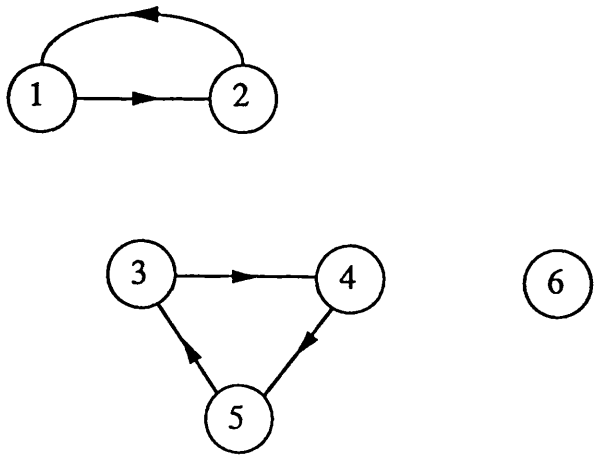


Figure 6.3.3.i

Now, consider a two stage modification of the formulation $\overline{BS_k}$. The modification converts the inequalities of constraint set $\overline{BS_k}2$ to equalities to give constraint set $\underline{BS_k}1$ of $\underline{BS_k}$. Then, adds constraint set $\overline{BS_k}1$ to constraint set $\underline{BS_k}1$ to give constraint set $\underline{BS_k}2$ of $\underline{BS_k}$. The conservation of flow condition becomes implicit.

$\underline{BS_k}$

$$\Sigma_{i\in P(k)}\Sigma_{j\in P(k)} (c_{ij}-z_i-m_iw_k-M_{ij}w_k)v_{ij}^{\ k}$$

s.t.

$$\Sigma_{j\in P(k)} v_{ij}^{\ k} = 1 \qquad \forall\, i\in P(k)\ \underline{BS_k1}$$

$$\Sigma_{j\in P(k)} v_{ji}^{\ k} = 1 \qquad \forall\, i\in P(k)\ \underline{BS_k2}$$

$$v_{ij}^{\ k} \geq 0 \qquad \forall\, i,j\in P(k)\ \underline{BS_k3}$$

This is the same as problem $\overline{BS_k}$, except that this time the solution must correspond to a set of schedules which include every train in the set P(k). The formulation $\underline{BS_k}$ is recognizable as an assignment problem, also known as a bipartite weighted matching problem. Papadimitriou and Steiglitz give the following theorem:

Theorem 6.3.3.1

The Hungarian Method correctly solves the assignment problem for a complete bipartite graph with 2n nodes in $O(n^3)$ arithmetic operations.

For a description of the Hungarian Method see Appendix C.

So, if it is possible to construct a complete bipartite graph associated with the problem $\underline{BS_k}$, then it is possible to solve the problem given by $\underline{BS_k}$ in polynomial time using the Hungarian Method.

By duplication of the trains in the set P(k) it is possible to create a complete bipartite graph $BG_k=(SV_k,TV_k,BA_k)$, where $SV_k=P(k)$, $TV_k=P(k)$ are the sets of source and sink nodes and $BA_k=P(k)xP(k)$ is the set of arcs. The weight on the arc

given by the ordered pair (i,j), $i \in SV_k$, $j \in TV_k$, is given by $(c_{ij} - z_i - m_i w_k - M_{ij} w_k) = C_{ij}$. Note that the Hungarian Method requires that the costs on the arcs are non-negative, so if $C_{ij} < 0$ for any ordered pair (i,j) then calculate $MC = \max_{(i,j)}(-C_{ij})$ and then solve the problem with costs $\overline{C}_{ij} = C_{ij} + MC$. The cost of the minimum solution can be found by calculating the value of $(\Phi - |P(k)| x MC)$, where Φ is the solution to the assignment problem.

So for the above example with $P(k) = \{1,2,3,4,5,6\}$, the graph BG_k is:

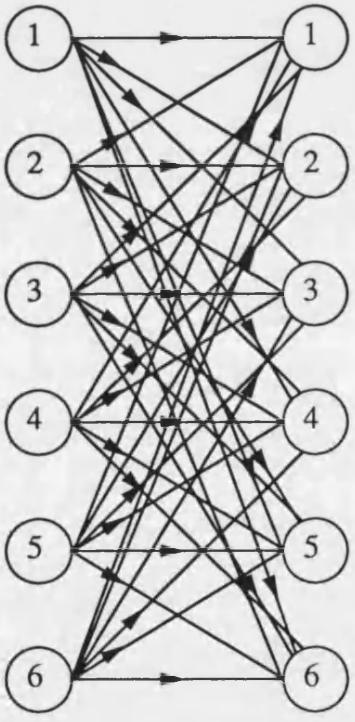


Figure 6.3.3.ii

The example solution of figure 6.3.3.i shown on this graph appears as:

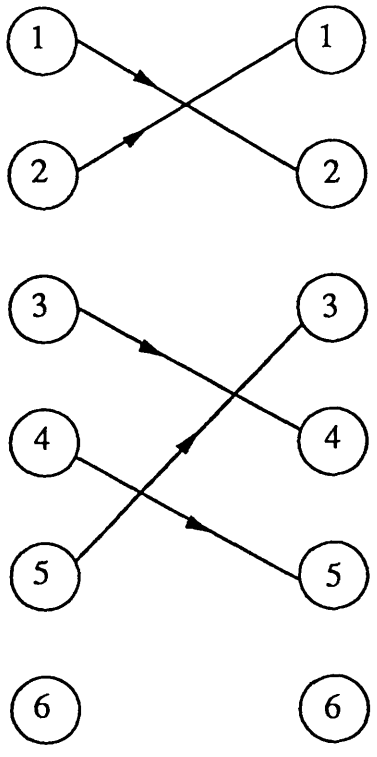


Figure 6.3.3.iii

Clearly, this is not a solution to the assignment problem as node 6 has not been covered. However, performing the matching 6→6 achieves a feasible solution to the problem $\underline{BS_k}$. Additionally, if the cost on the arc (6,6) is taken to be zero then the solution is not only a feasible but also an optimal solution to the assignment problem as the objective function value agrees with that found using the simplex method. Moreover, if all arcs (i,i) have a cost $C_{ii}=0$, then any optimal solution to $\overline{BS_k}$ can be converted to an optimal solution to $\underline{BS_k}$ by assigning any unassigned train to itself. The question is: Is it possible to construct the subproblems so that it becomes reasonable to set $C_{ii}=0$, solve the assignment problem and from the solution to $\underline{BS_k}$ derive an optimal solution to the problem $\overline{BS_k}$?

In the context of the column generation technique, if all singleton sequences, i.e. all sequences of the type i→i, are included in the master problem at the outset then in the pricing out exercise all singleton sequences will have a non-negative reduced cost. This means that if a singleton sequence appears in a solution set of sequences for the problem $\overline{BS}_k$, it contributes a zero cost to the value of objective function of $\overline{BS}_k$. This is clear, as a singleton sequence with a positive reduced cost would never appear in the solution to $\overline{BS}_k$. It is the inclusion of all singletons in the master problem which makes it is possible to set $C_{ii}=0$ in the assignment matrix and find a solution to $\underline{BS}_k$ which has the same value as the solution to $\overline{BS}_k$. Moreover, the optimal solution to $\underline{BS}_k$ can be converted into an optimal solution to $\overline{BS}_k$. To see this, consider the possible states of a train i in an optimal solution to the problem $\overline{BS}_k$:

i) train i is included in a schedule length ≥ 2;

ii) train i is included in a schedule length $= 1$, i.e. i→i;

or, iii) train i is not included in any schedule.

If a particular train i *must* be present in a schedule of length ≥ 2 in any optimal solution of $\overline{BS}_k$, then the Hungarian Method will locate such a schedule. If train i *must not* appear in any schedule of length ≥ 2 in an optimal solution of $\overline{BS}_k$ but may or may not appear in a singleton sequence, then in this situation the assignment i→i is made. In effect, the assignment i→i is a default setting which is used to satisfy the specification that an optimal solution to $\overline{BS}_k$ need not include all the trains in P(k). A feasible and optimal solution for the problem $\overline{BS}_k$ is then obtained from the solution to $\underline{BS}_k$ by removing all singleton sequences. The removal of all singleton sequences from the solution to $\underline{BS}_k$ is valid as the addition of the singleton sequence to the master problem represents a duplication of columns. Therefore only those sequences of length ≥ 2 need to be added to the master problem. The validity of this approach is confirmed empirically and the results of the experiments are given in chapter seven.

6.4 Constructing Subproblems

For the stock diagramming problem with K locomotive types the set of subproblems has a natural order equal to K, one subproblem for each locomotive type. If the solution to each of these subproblems is zero then the optimality condition is satisfied and an optimal solution to the original problem given by SD is found. However, it may be possible to accelerate the progression towards optimality by constructing and solving additional subproblems to generate more columns at each iteration of the column generation technique. Given that there are K subproblems based on the sets P(k)={i | train i can be worked by locomotive type k}, it is possible to construct subproblems based on the proper subsets of P(k). Such a family of subsets of the set P(k) can be constructed using the argument of chapter four, i.e. by considering type sets. Recall from the discussion of chapter four that the problem AP is solved for one locomotive type in a type set and then copies made of the sequences found. For the subproblems, it is not possible to solve the subproblem for one locomotive type in the type set and then make copies of the sequences found, as the values of w_k may differ between locomotive types. Therefore, based on the trains in the type set, a subproblem is constructed for each locomotive type which defines the type set. For example, if the type set 'TYPE k_1 AND k_2 ONLY'={4,5,6}, then two subproblems based on the set of trains {4,5,6} are constructed. The first subproblem relates to locomotive type k_1 and the second relates to locomotive type k_2. It is possible to control the number of subproblems considered by an appropriate specification of the type sets. For instance, a limiting condition may be: "Consider the type sets based on all combinations of length ≤ 3 of locomotive types." During the solution of the subproblems using the three methods described above, the actual control strategies used are stated in chapter seven.

CHAPTER SEVEN

OPTIMAL SOLUTION

7.1 Introduction

This chapter describes in detail the application of the column generation technique to the stock diagramming problem. This discussion begins with a description of how the master problem is constructed, augmented and solved. Following on, there is a comparison of the different subproblem solution methodologies employed.

7.2 The Master Problem

7.2.1 Solution of the Master Problem

In every instance in the following discussion the linear programming relaxation of the master problem is solved using the primal simplex algorithm. This algorithm is available in LAMPS, the commercial mathematical programming package introduced in chapter four.

The formulation of the linear programming relaxation of the master problem SPP presented in chapter five is given here as:

RSPP

$$\text{MIN } \Sigma_{r\in\Omega}\, c_r s_r$$

s.t.

$$\Sigma_{r\in\Omega}\, \delta_{ri} s_r = 1 \qquad \forall\, i\in\tau \quad \textbf{RSPP1}$$

$$\Sigma_{r\in\Omega}\, L_{rk} s_r \leq U_k \qquad \forall\, k\in\Lambda \quad \textbf{RSPP2}$$

$$s_r \geq 0 \qquad \forall\, r\in\Omega \quad \textbf{RSPP3}$$

The sets, variables and data are as defined for formulation SPP.

The problem as specified is to find a subset of sequences from a given set of sequences Ω. The subset chosen must represent a minimum cost mix of sequences which work each train exactly once whilst using no more than the number of locomotives available. So, in order to solve the master problem, and commence the column generation technique, a set of starting sequences Ω is required.

7.2.2 Construction of the Set Ω

The starting set of sequences is constructed using one of the heuristic procedures described in chapter four. The resulting set of sequences may not contain a subset of sequences which represent a feasible solution to the master problem. If this is the case then a standard Phase I procedure is used in conjunction with the column generation technique to locate a feasible solution to the master problem.

7.2.3 Phase I

The formulation used for the Phase I procedure is a simple adaption of the master problem RSPP.

RSPPI

$$\text{MIN } \Sigma_k \rho_k$$

s.t.

$$\Sigma_{r\in\Omega} \delta_{ri} s_r = 1 \qquad \forall\, i\in\tau \quad \textbf{RSPPI1}$$

$$\Sigma_{r\in\Omega} L_{rk} s_r \leq U_k + \rho_k \qquad \forall\, k\in\Lambda \quad \textbf{RSPPI2}$$

$$s_r \geq 0 \qquad \forall\, r\in\Omega \quad \textbf{RSPPI3}$$

$$\rho_k \geq 0 \qquad \forall\, k\in\Lambda \quad \textbf{RSPPI4}$$

where the sets, variables and data are as defined for SPP with the addition that:

ρ_k - is the artificial variable associated with locomotive type k.

Alternative Phase I formulations exist but, as all singletons are automatically included in the set Ω, it is known that the constraint set RSPPI1 is satisfied immediately and constraint set RSPPI2 can then be satisfied by the addition of artificial variables ρ_k. So, the problem RSPPI seeks to find a solution which works each train exactly once but uses as few locomotives above the limit on the stipulated number available as possible. If the optimal solution to RSPPI for the set Ω is not zero, then there does not exist a mix of sequences from the set Ω which represents a feasible solution to the problem RSPP. It is therefore necessary to use the column generation technique to generate sequences and augment the set Ω until: $\Sigma_k \rho_k=0$ and feasibility is achieved; or, the column generation procedure terminates with $\Sigma_k \rho_k>0$ and the problem is proven infeasible. If the problem RSPP is feasible, then when $\Sigma_k \rho_k=0$ all the artificial variables ρ_k are at their lower limit of zero and can be eliminated.

The subproblems associated with RSPPI are exactly the same as those of the problem RSPP barring a change in the objective function coefficients of the variables i.e.

BS_kI

$$\text{MIN } \Sigma_i\Sigma_j\ (-z_i - m_i w_k - M_{ij} w_k) v_{ij}^k$$

s.t.

$$\Sigma_j\ v_{ij}^k = \Sigma_j\ v_{ji}^k \qquad \forall\ i \in P(k)\ \ BS_kI1$$

$$\Sigma_j\ v_{ij}^k \leq 1 \qquad \forall\ i \in P(k)\ \ BS_kI2$$

$$v_{ij}^k \geq 0 \qquad \forall\ i,j \in P(k)\ \ BS_kI3$$

The iterative process of Phase I is illustrated in figure 7.2.3.i. Notice that each time the master problem is augmented it is not solved from a slack basis. Instead, the previous basis is stored and the simplex algorithm recommences from this old basis.

Having found an initial feasible solution to RSPP the Phase II procedure commences.

7.2.4. Phase II

In Phase II the column generation technique is used to solve RSPP and take the optimal solution to RSPPI from the status of a feasible solution to RSPP to that of an optimal feasible solution.

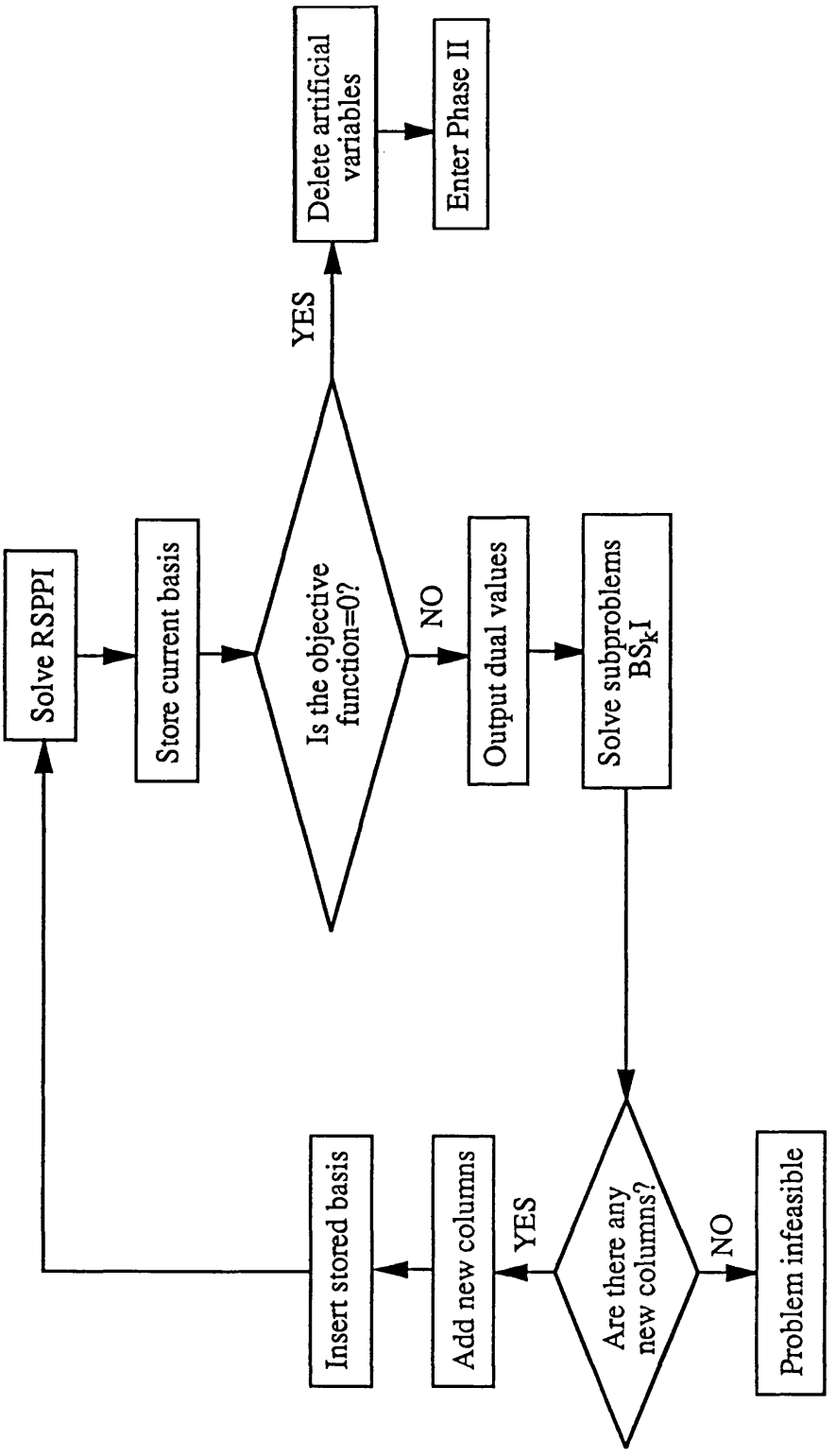

Solve RSPPI
Store current basis
Is the objective function=0?
YES
Delete artificial variables
Enter Phase II
NO
Output dual values
Solve subproblems BS_kI
Are there any new columns?
YES
Add new columns
Insert stored basis
NO
Problem infeasible

Figure 7.2.3.i

The Phase II procedure is similar to the Phase I procedure except that, having secured feasibility, there is only one stopping criterion, namely: if no negative cycles exist, then the solution is optimal and the procedure terminates. Schematically the procedure is as shown in figure 7.2.4.i.

The interaction between the master problem and the subproblems during the Phase I and Phase II procedures is now established. The solution methods used to solve the subproblems are now discussed. A data set SET189, described in chapter four, is used as a tool for explanation.

7.3 Test Set of Data

In chapter four the heuristics HAP and ModHAP are applied to the data set SET189. For each of the heuristics the solution of the master problem based on the initial set of sequences was found to be infeasible. The sum of infeasibilities was equal to 13.3367 when heuristic HAP was used and 12.1053 when ModHAP was used. Hence, having used either HAP or ModHAP to create the initial set of sequences, both Phase I and Phase II procedures are required to find an optimal solution to RSPP for SET189. As each subproblem solution methodology is discussed, it is made clear which heuristic is used to create the set Ω.

7.4 Subproblem Methods

7.4.1 Shortest Path Method

Interaction between Master and Subproblems

Having solved the Master Problem using the primal simplex algorithm of LAMPS, LAMPS produces an output file of the dual values. The FORTRAN

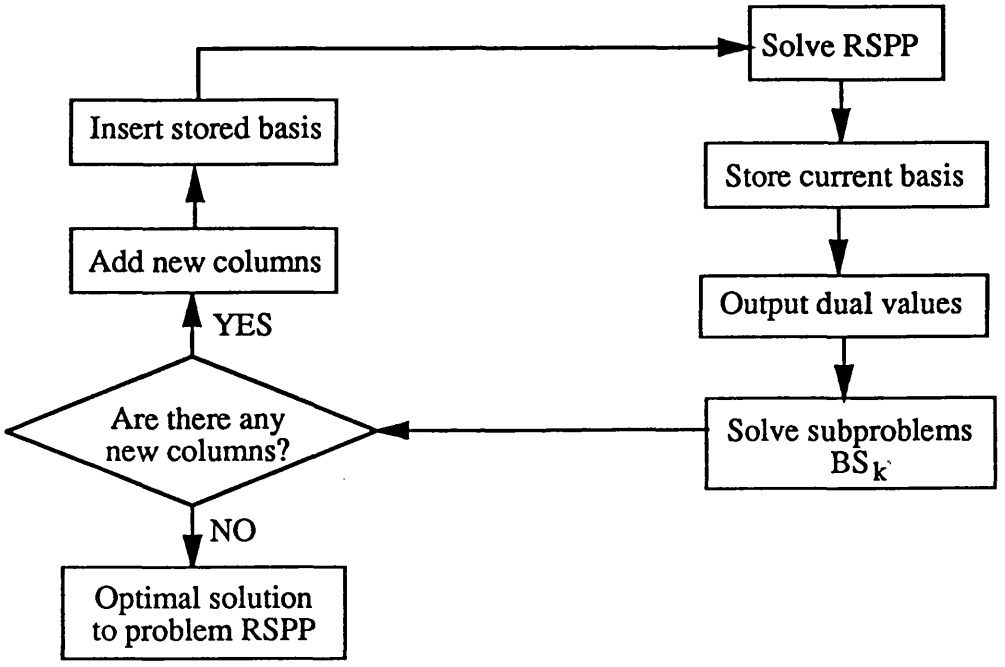

Solve RSPP
Store current basis
Output dual values
Solve subproblems
BS_k
Are there any
new columns?
YES
Add new columns
Insert stored basis
NO
Optimal solution
to problem RSPP

Figure 7.2.4.i

program AUTONEG is then used to find new columns using the matrix multiplication method in conjunction the Bellman-Ford algorithm. AUTONEG produces an output file which stores the new columns in the AMS format required by LAMPS. LAMPS is re-entered, the new columns added, and the master problem resolved. A flow-diagram of AUTONEG is given at figure 7.4.1.i. The parameter COUNT is used to control the number of repetitions of the shortest path method for each of the subproblems. Figure 7.4.1.ii describes how this method is incorporated into the column generation technique for a Phase II procedure.

Type Sets

The number of subproblems solved at each iteration of the column generation technique is dependent on:

i) the number of type sets specified and therefore the number of subproblems created (see chapter six for further details);
ii) the number of times the shortest path method is repeated for each subproblem.

Dealing with i) first. For the data set SET189 the type set specification was: "Consider all combinations of length ≤ 3 of locomotive types." An example of such a type set might be: 'TYPE 158 ONLY'={all those trains which can be worked by locomotive types 1 and 5 and 8 but no other locomotive types}. Recall that for each combination of locomotive types two type sets are specified. Therefore, for the data set SET189 which has 10 locomotive types, for this specification the maximum number of type sets possible is 350. Some of these type sets may be empty. ii) refers to the value given to the parameter COUNT.

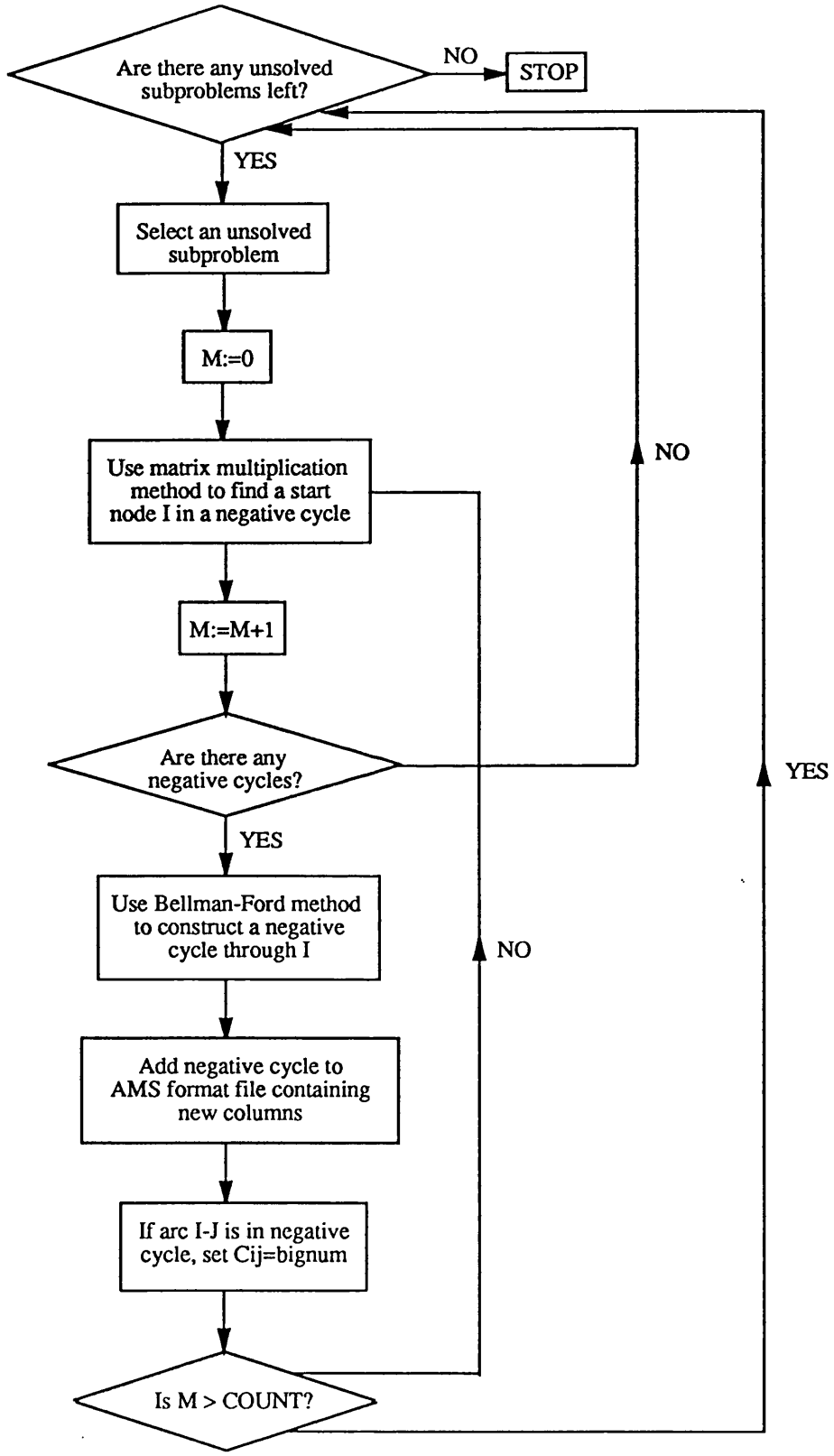
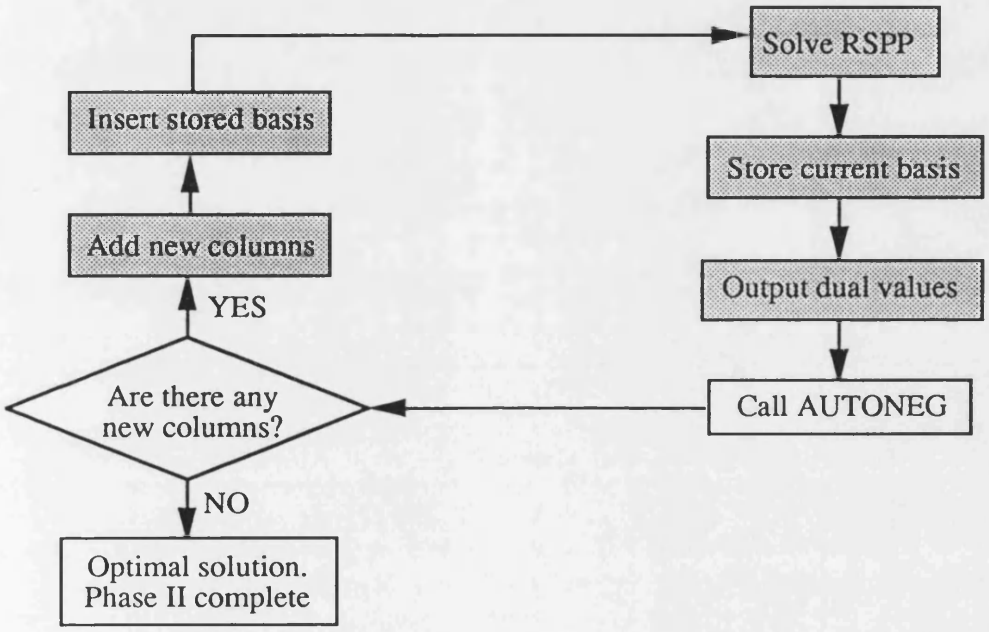

Are there any unsolved subproblems left?
NO
STOP
YES
Select an unsolved subproblem
M:=0
Use matrix multiplication method to find a start node I in a negative cycle
NO
M:=M+1
Are there any negative cycles?
YES
YES
Use Bellman-Ford method to construct a negative cycle through I
NO
Add negative cycle to AMS format file containing new columns
If arc I-J is in negative cycle, set Cij=bignum
Is M > COUNT?

Figure 7.4.1.i



Procedures carried out by LAMPS are indicated by the shaded boxes.

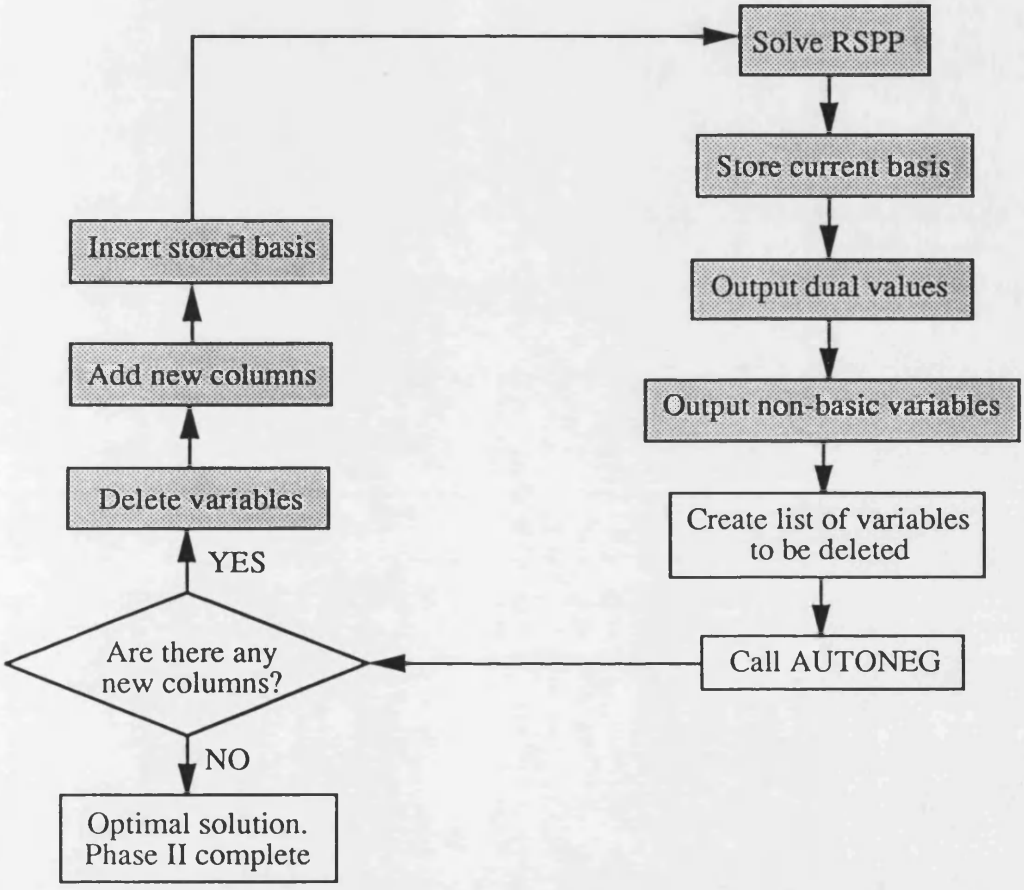
Figure 7.4.1.ii

Restricted Master

For large problems such as SET189, by solving a number of subproblems at each iteration of the column generation technique, the number of columns generated may be sufficient to result in an explosion in the number of variables in the master problem. This growth in the number of variables may give rise to storage and running time problems. Therefore, a restricted form of the master problem is constructed at each iteration. The restriction used here is to delete all current non-basic variables from the master problem before addition of the new columns. The singletons are excluded from consideration during the deletion process. This ensures that during Phase I constraint set RSPPI1 is satisfied and a feasible solution to the problem RSPPI exists. See chapter five for a more detailed discussion on the theory of the restricted master problem. Figure 7.4.1.iii is the same as figure 7.4.1.ii except that it incorporates the use of this restricted master problem.

Results of Phase I

The results shown in table 7.4.1.iii refer to the Phase I procedure applied to the data set SET189. The starting set of sequences were created by the heuristic HAP. The value of COUNT was set at 1 up to iteration 12 when it was increased to min(n_s,10), where n_s is the number of trains in the current subproblem. As expected the number of columns added increased once COUNT was increased. In the table, the value of the objective function after each iteration of the column generation technique is given, along with the percentage decrease in the objective function and the number of columns added. The key to the columns of the table is:



Procedures carried out by LAMPS are indicated by the shaded boxes.

Figure 7.4.1.iii

'ITERATION' - number of the iteration.

'COL. ADD.' - the number of columns added .

'OBJECTIVE' - the value of the objective function.

'% DECREASE' - the percentage decrease in the objective function.

ITERATION	COL. ADD	OBJECTIVE	% DECREASE
0	-	13.3367	-
1	480	12.7951	4.06
2	1376	12.7889	0.05
3	800	12.7888	0.00
4	1280	12.3685	3.29
5	448	12.1646	1.65
6	1088	12.0142	1.24
7	193	11.8157	1.65
8	480	11.8071	0.07
9	449	11.7376	0.59
10	256	11.7059	0.27
11	321	11.6758	0.26
12	852	11.6478	0.24
13	1816	11.2649	3.29
14	1900	10.6873	5.13
15	2202	9.6947	9.29
16	1600	9.4192	2.84
17	1474	9.2410	1.89
18	1835	9.0160	2.43
19	2173	8.4577	6.19

Table 7.4.1.iii

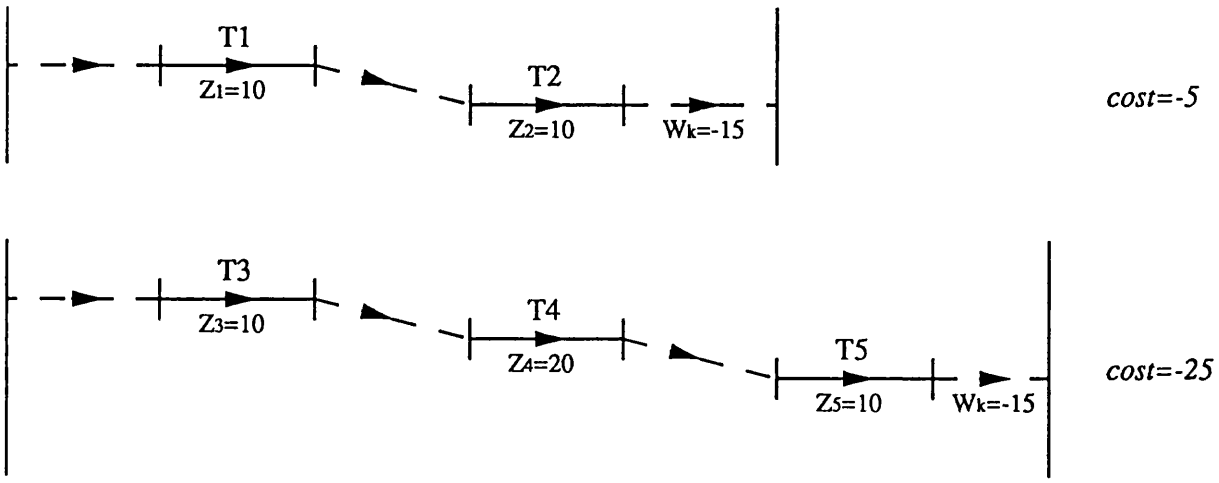
Discussion of Results

Clearly progression towards zero is very gradual, even though a marked improvement occurred after the value of COUNT was increased. This method proved to be, not only inefficient in producing a decrease in the objective function, but also very cumbersome and slow. As shown by the results, a large number of columns were generated at each iteration. This justifies the use of the word 'cumbersome' as a label for the method as a number of variables had to be handled at each iteration. This translates into the use of a large amount of computing time due to: i) the generation of the columns; ii) the retrieval and deletion of the columns within LAMPS; and, iii) the addition of the columns to the master problem. Although LAMPS is versatile and incorporates many useful features, it remains a generalized mathematical programming package and therefore does not lend itself to the large amount of data manipulation required by this shortest path method. Additionally, as an example of i), the time to generate the 2173 columns at iteration 19 in table 7.4.1.c using the FORTRAN program AUTONEG on a VAX 6330 was 39'03" of CPU time. When AUTONEG was translated to Salford FORTRAN and run on an Epsom PC AX3, after 2 hours only 884 columns had been generated and so the procedure was interrupted.

It is for these reasons that the shortest path method was dismissed as a viable means of solving the stock diagramming problem using a PC-based method. This method was abandoned at this stage in favour of the simplex method. However, before proceeding to discuss the simplex method, some further investigations carried out using the shortest path method are described. The experiments are based on the interpretation of the dual values.

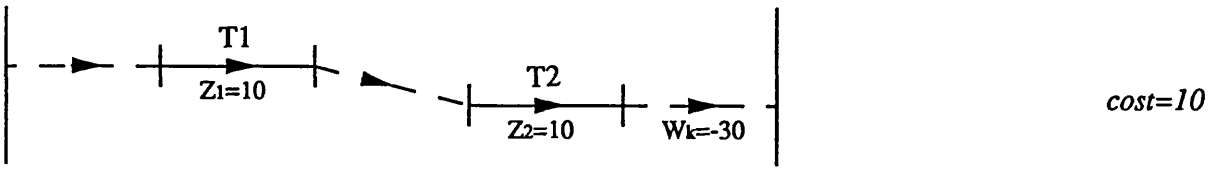
Interpretation of Dual

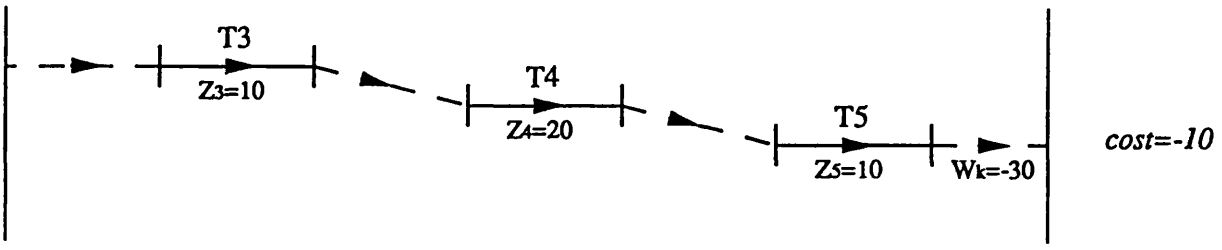
In an attempt to accelerate the progression of the objective function of the master problem towards zero, before passing the information on the dual values to the subproblems, the dual value for the locomotives is doubled; i.e. w_k becomes $2w_k$ for each k. The rationale behind this transformation is based on a twofold argument. Firstly, by increasing the marginal cost of a locomotive and thereby presenting locomotives as a scarce resource, the hope is that the solutions to the subproblems are sequences which use fewer locomotives. The aim is to satisfy constraint set RSPPI2 in as few iterations as possible. The second intention is that, the doubling of the marginal cost of a locomotive type makes sequences with a small negative cost take a non-negative cost, and those with a large negative cost remain negative. Hence, when the pricing out exercise is performed using the true dual values, the negative cycles found represent a greater violation in the optimality conditions. To illustrate this hypothesis consider two pairs of negative cycles which represent the situation before and after the transformation of the dual values. In the following diagrams the solid lines represent the train arcs and the dashed lines represent the light-run arcs. 'Tn' indicates train n.



BEFORE

'BEFORE' the dual value w_k is doubled, the first sequence is a negative cycle and would be located by the matrix multiplication method before the negative cycle shown by the second sequence. This is because the first sequence contains only two trains.





AFTER

'AFTER' the value of w_k is doubled the first sequence has a positive cost, but the second sequence still has a negative cost after the transformation.

If after a number of iterations no negative cycles are located, this does not mean that the stopping criterion for the column generation technique has been satisfied, as the doubling of w_k may have made a negative cost cycle appear to be non-negative. The next step is to reduce the factor by which the dual value w_k is multiplied by a step size less than or equal to 1. The process can be repeated, reducing the multiplication factor once all negative cycles are found, until the multiplication factor equals 1 and the true dual values are being used. The results of using this scheme, hereafter referred to as M2, are given in table 7.4.1.iv. The dual values used to start M2 were the same as those used for iteration 8 of the original scheme. For each iteration the multiplication factor used was 2. The value of COUNT

was fixed at 1 up to iteration 5, then increased to min(n_a,10) for iteration 6 onwards.

ITERATION	COL. ADD	OBJECTIVE	% DECREASE
0	-	11.8157	-
1	474	11.7382	0.67
2	193	11.7153	0.20
3	449	11.7153	0.00
4	464	11.7142	0.01
5	464	11.7072	0.06
6	1504	11.2649	3.78
7	1676	10.5704	6.17
8	1622	9.7750	7.52
9	1035	8.3713	14.36
10	2265	7.6880	8.16
11	1779	7.0132	8.78
12	1775	6.5621	6.43
13	1033	6.0136	8.39
14	1806	5.4381	9.57
15	1104	4.7505	12.64
16	883	4.7470	0.07
17	949	4.3728	7.88
18	1712	4.1840	4.32
19	949	3.7157	11.19
20	1453	3.5969	3.20
21	1521	3.2874	8.60
22	1753	2.9657	9.79
23	1179	2.8583	3.62

Table 7.4.1.iv

In the main, the decrease in the value 'OBJECTIVE' appears to be more promising than that shown in table 7.4.1.iii. Nevertheless, it is still not sufficient to justify pursuing with this method.

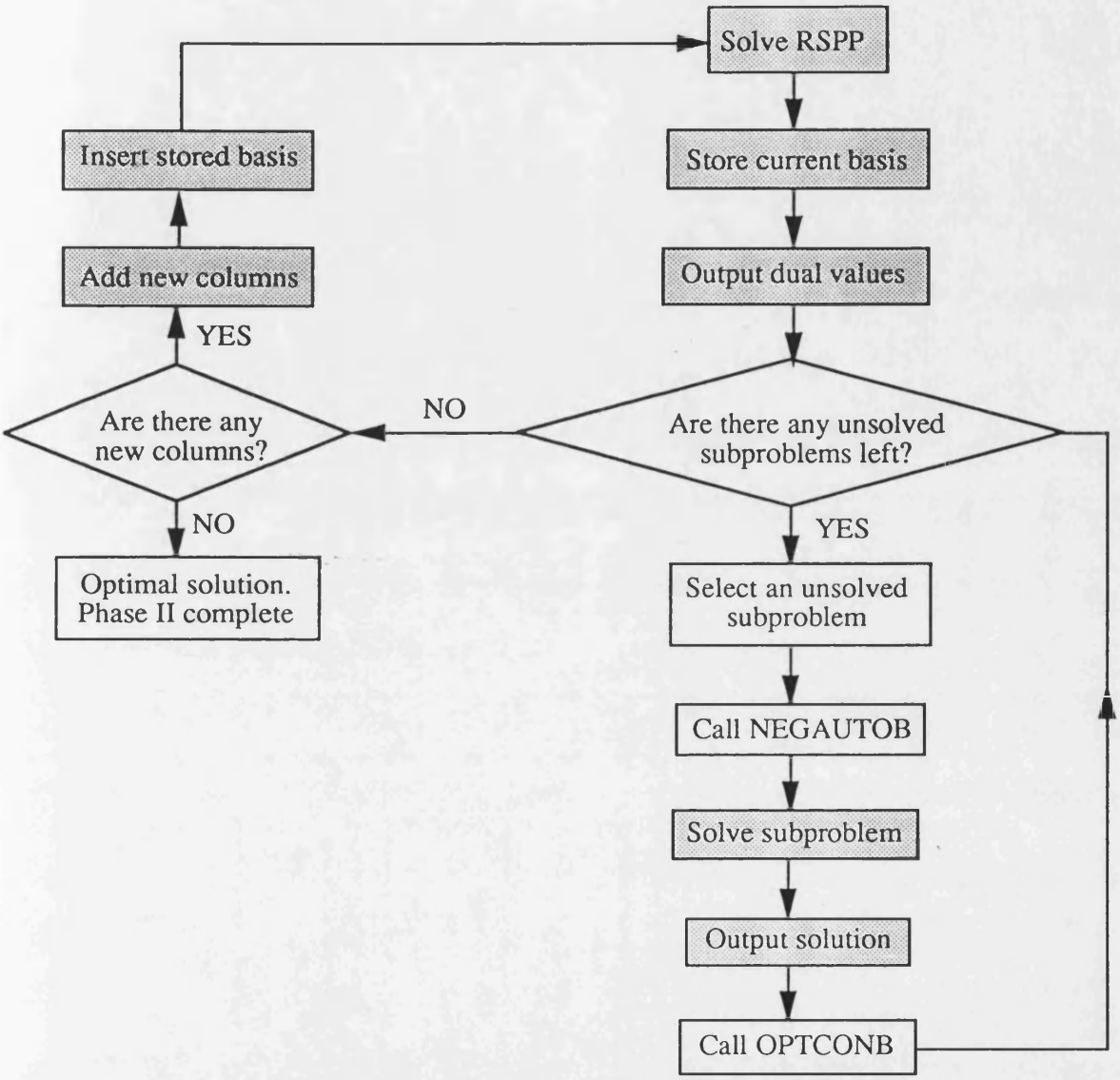
7.4.2 Simplex Method

Interaction between Master and Subproblems

Using the dual values retrieved from LAMPS, the FORTRAN program NEGAUTOB constructs a subproblem from the list of subproblems in the AMS format required by LAMPS. Each subproblem is then called by LAMPS and solved using the primal simplex algorithm. If a negative cost solution to a subproblem is found, the solution is output by LAMPS. The FORTRAN program OPTCONB is then used to convert the solution from LAMPS and produce an AMS format file which describes the new columns to be added to the master problem RSPP. LAMPS is re-entered, the new columns added and the master problem resolved. A flow-diagram of the procedure for Phase II is given at figure 7.4.2.i.

Type Sets

As the simplex algorithm is non-polynomial, it was decided that the best policy is to use fewer type sets than the number used for the shortest path method. The type set specification is: "Consider all combinations of length = 1 of locomotive types". For the SET189 data set this gives rise to 20 possible subproblems, and these are all of the form:



Procedures carried out by LAMPS are indicated by the shaded boxes.

Figure 7.4.2.i

'TYPE k ONLY' = {all those trains which can be worked by a locomotive type k, but no other locomotive type};

and,

'TYPE k' = {all those trains which can be worked by a locomotive type k}.

Some of these sets are empty, and therefore not considered. For example, 'TYPE 7 ONLY'={ }.

Iterations

The 20 subproblems are each solved once. Again, this is done in recognition of the fact that the simplex method is non-polynomial. It is possible to find additional columns to add to the master problem by banning a connection in a negative cost solution just located by the simplex method, then resolving the modified subproblem using the simplex method. This is done for the shortest path method. If such a repetition policy is pursued, the first solution found for each subproblem is the minimum cost solution. Any subsequent non-zero solutions are multiple solutions, or solutions which violate the optimality conditions but have an increased solution value.

Restricted Master

Only a few columns are produced at each iteration, see results table 7.4.2.i, therefore all columns added to the master problem are kept along with the original set Ω.

Results of Phase I

The results given in table 7.4.2.i refer to the Phase I procedure. The starting set of sequences Ω was created by the heuristic ModHAP. The results are presented for each iteration of the column generation technique for Phase I. The column headings are the same as those given in the tables of section 7.4.1.

ITERATION	COL. ADD	OBJECTIVE	% DECREASE
0	-	12.1053	-
1	29	11.3090	6.58
2	25	9.0374	20.09
3	34	6.1336	32.13
4	36	3.7472	38.91
5	31	2.4037	35.85
6	25	1.1889	50.54
7	21	0.4613	61.20
8	23	0.0000	100.00

Table 7.4.2.i

The Phase I procedure was completed successfully in only 8 iterations of the column generation technique.

As an indication of the time required to a solution to a subproblem using this method, an example is given here for a single type set with 100 trains. For SET189

the orders of the type sets range from a maximum of 161 to a minimum of 3. The times were recorded on an EPSON PC AX3.

	CREATE:	57 seconds
Modules of LAMPS:	CONVERT:	97 seconds
	SETUP:	24 seconds
	PRIMAL:	100 seconds

CREATE refers to the time to specify the subproblem in an AMS format file using the Salford FORTRAN program NEGAUTOB. Once specified, the subproblem can be called from LAMPS. There are three modules of LAMPS which have to be entered to solve the problem. CONVERT takes the AMS format file and converts it into a format which is easily acceptable to all the tasks of LAMPS. SETUP further converts the file created by CONVERT into a more compact and efficient format suitable for the primal simplex algorithm. PRIMAL invokes the primal simplex method to find an optimal solution to the problem. The total time represented by the above parts is 278 seconds.

Discussion of Results

Compared to the shortest path method, this method yields a significant decrease in the objective function of the master problem at each iteration. Also, given that a subproblem with 100 trains can be solved in under 300 seconds, the time to complete an iteration of the column generation technique on a PC is more acceptable. The number of columns retained in the master problem when using the simplex method is greater than in the shortest path method, i.e. c.17,000 compared with c.2000. This is as a result of the different policies used for restricting the size of the

master problem in the two cases. However, as the basis of the current solution to the master problem is saved after each iteration of the column generation technique, the time taken to perform the task PRIMAL in LAMPS does not become prohibitively great. As an example, when a partial pricing policy is used, the time to solve a problem of 18,931 columns and 210 rows from a stored basis is 470 seconds of CPU time and takes 731 iterations of simplex. Under a total pricing policy for the same problem, the time is 794 seconds of CPU time and takes only 272 iterations of simplex. Times were recorded on an IBM PS/2 Model 55 SX computer. These results affirm the superiority of the simplex method over the shortest path method. It was therefore decided that this method should be adopted for the Phase II procedure.

Phase II

The implementation and the type sets used are exactly the same as for the Phase I procedure.

Results of Phase II

On replacement of the objective function of RSPPI with that of RSPP, the solution found by the Phase I procedure was shown to have a cost of 6438.5970 light-run minutes. The Phase II procedure terminated after 94 iterations of the column generation technique. The cost of the solution to the master problem was 710 light-run minutes. This is the value of the optimal solution to the linear programming relaxation of the original problem. However, the solution found was not integer so the value of 710 light-run minutes only represents a lower bound on the solution to the original problem given by SD. The method used to find an integer solution is described in chapter eight.

The problem of degeneracy was encountered during the execution of the Phase II procedure and the figure of 94 for the number of iterations includes some instances of cycling. In fact, the solution value of 710 was found after 54 iterations and thereafter the problem continued to cycle until the optimality conditions were satisfied. Problems with degeneracy also occurred during the execution of the primal simplex algorithm for the master problem. It was as a consequence of these experiences that methods for dealing with degeneracy are reviewed.

Degeneracy

It is not the purpose of this thesis to provide a well-constructed analysis of degeneracy resolution methods. Instead, the limited experience achieved during the solution of this problem is related. A discussion of how to deal with cycling between iterations of the column generation technique is given in chapter nine. The discussion which follows considers degeneracy resolution techniques when degeneracy occurs during the execution of the LAMPS primal simplex algorithm.

One method attempted was a right-hand side perturbation scheme. See Ryan [66] for an example of the use of this scheme. This method worked on nearly all the occasions on which it was employed. However, interaction with LAMPS and the amount of data manipulation required made this exercise a cumbersome task. A discussion with one of the authors of LAMPS revealed that LAMPS incorporates a right hand side perturbation scheme for dealing with degeneracy, and it was the use of a partial pricing scheme which was giving rise to problems with degeneracy. So, the alternative approach was to override the partial pricing facility and price out all variables at each iteration of the primal simplex algorithm. This is time consuming but not as frustrating as observing a problem continue to cycle. This method proved to be successful each time it was used.

7.4.3 Assignment Method

Interaction between the Master and Subproblems

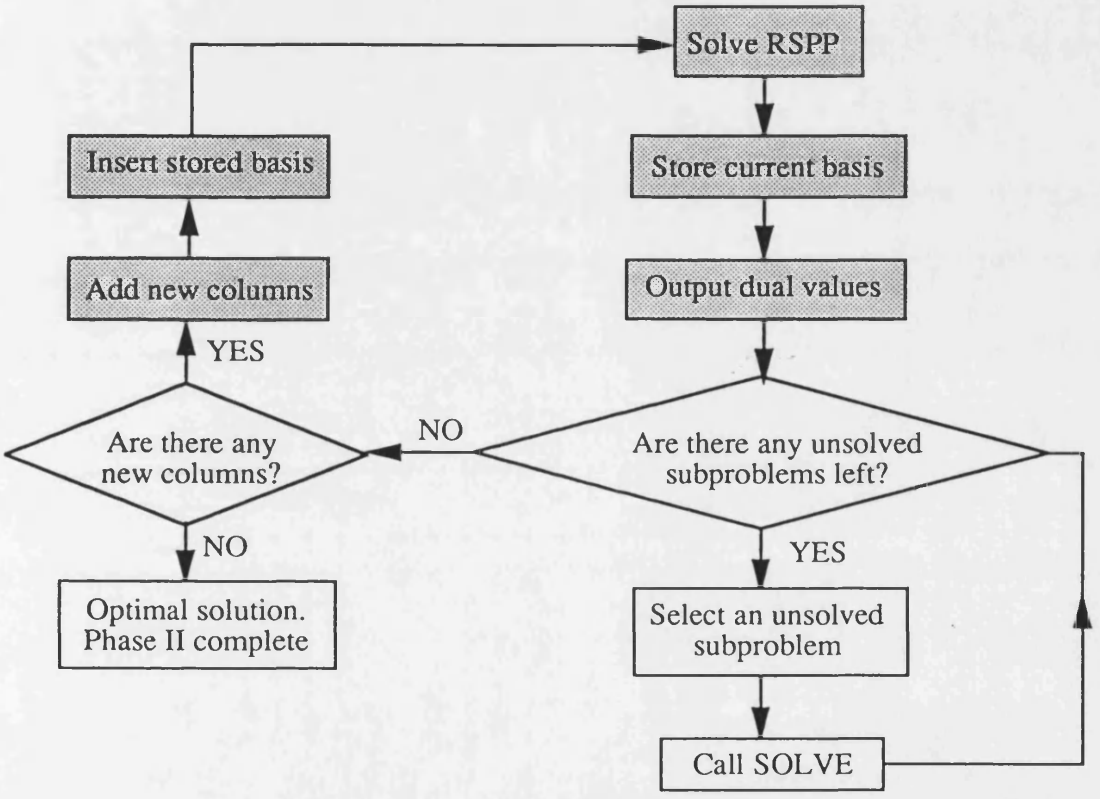
For each subproblem in turn, the file containing dual values output by LAMPS is called by the FORTRAN program SOLVE. The program SOLVE then: constructs assignment problem $\underline{BS_k}$ for the particular subproblem; solves the assignment problem using the Hungarian Method; interprets the results; and, if the solution is non-zero, augments AMS format file containing the new columns. The column generation technique for Phase I and Phase II then continues as for the simplex method. The flow-diagram of the procedure for Phase II is given at figure 7.4.3.i.

Assignment Method v. Simplex Method

This method was not used to solve RSPP for SET189 using the column generation technique. Instead, a straight comparison was made between the simplex method and the assignment method to: a) check that they gave the same results; and, b) compare solution times. To make these comparisons two sets of actual dual values, retrieved during the solution of SET189 using the simplex method, were used as the costs for the subproblems. The two sets of dual values are referred to as DV1 and DV2. For each set of dual values tests were performed on 11 subproblems based on 11 type sets of varying order.

Results

In the following tables, 7.4.3.i and 7.4.3.ii, the results of solving the subproblems are given for the simplex method and the assignment method. Tables 7.4.3.i and 7.4.3.ii refer to sets DV1 and DV2, respectively. The key to the tables is:



Procedures carried out by LAMPS are indicated by the shaded boxes.

Figure 7.4.3.i

'TYPE SET' - is the name of the type set on which the subproblem is based.

'TRAINS' - states the number of trains in the subproblem, i.e. the order of the type set.

'OBJ. SIM' - gives the cost of the solution to the subproblem found using the simplex method

'OBJ. ASS' - gives the cost of the solution to the subproblem found using the assignment method.

TYPE SET	TRAINS	OBJ. SIM.	OBJ. ASS.
T1i	34	-319.29	-319.30
T2i	8	0.00	0.00
T3i	13	0.00	0.00
T4i	23	0.00	0.00
T5i	14	-140.46	-140.47
T6i	53	-2700.27	-2700.37
T7i	31	-307.86	-307.89
T8i	51	-1325.64	-1325.65
T9i	53	-2195.53	-2195.50
T10i	3	0.00	0.00
T11i	8	0.00	0.00

Table 7.4.3.i

TYPE SET	TRAINS	OBJ. SIM.	OBJ. ASS.
T1ii	44	-942.96	-943.00
T2ii	17	-391.20	-391.20
T3ii	13	0.00	0.00
T4ii	33	-185.29	-185.24
T5ii	47	-1385.20	-1385.24
T6ii	81	-4977.64	-4977.32
T7ii	72	-2020.43	-2020.36
T8ii	99	-6683.48	-6682.91
T9ii	100	-7504.69	-7505.00
T10ii	3	0.00	0.00
T11ii	15	-43.28	-43.28

Table 7.4.3.ii

Any disparity in the values 'OBJ. SIM.' and 'OBJ. ASS.' is due to rounding errors. Solutions to the assignment problem which did not agree with those found by the simplex method were subsequently proved to be a multiple solution of the solutions found by the simplex method. This was done by first observing the reduced costs of the variables. Then, for completeness, the simplex solution was forced to be the same as the assignment solution and a check of the objective function value confirmed it to be an optimal solution.

Having established that the solution found by the assignment method agrees with that found by the simplex method, the following tables compare the CPU times to find an optimal solution to a subproblem using both methods. Table 7.4.3.iii relates to table 7.4.3.i and table 7.4.3.iv to table 7.4.3.iii. All the times in the tables are given in seconds. For the simplex method the times given are:

'CREATE' - see section 7.4.2 for definition.

'CONVERT' - see section 7.4.2 for definition.

'SETUP' - see section 7.4.2 for definition.

'PRIMAL' - see section 7.4.2 for definition.

'TOTAL' - this is the sum total of the times used by CREATE, CONVERT, SETUP and PRIMAL.

For the assignment method the times given are:

'SOLVE' - the total time to execute the program SOLVE.

'ASS.' - the time to solve the subproblem using the Hungarian Method. (This time is included in 'SOLVE'.)

TYPE SETS	SIMPLEX METHOD					ASSIGNMENT METHOD	
	CREATE	CONVERT	SETUP	PRIMAL	TOTAL	SOLVE	ASS.
T1i	10.77	12	4	8	34.77	8.79	0.50
T2i	4.29	2	2	2	12.29	4.73	0.11
T3i	7.36	6	2	6	21.36	6.27	0.22
T4i	4.78	3	2	3	12.78	6.05	0.11
T5i	4.95	3	2	2	11.95	6.15	0.16
T6i	18.95	29	7	22	76.95	14.95	1.38
T7i	9.78	11	4	7	31.78	8.95	0.49
T8i	17.97	27	6	21	71.97	15.05	2.09
T9i	17.42	26	6	19	70.42	16.04	2.14
T10i	4.50	3	2	< 1	10.50	4.34	0.11
T11i	4.23	2	2	1	9.23	4.73	0.11

Table 7.4.3.iii

TYPE SETS	SIMPLEX METHOD					ASSIGNMENT METHOD	
	CREATE	CONVERT	SETUP	PRIMAL	TOTAL	SOLVE	ASS.
T1ii	13.96	20	5	15	53.96	11.59	0.33
T2ii	4.67	4	2	3	13.67	5.05	0.16
T3ii	8.96	11	3	6	29.86	8.52	0.22
T4ii	4.56	4	2	3	13.56	5.49	0.11
T5ii	15.50	33	6	20	74.50	13.46	0.82
T6ii	38.35	66	16	53	173.35	29.34	5.11
T7ii	31.10	51	12	43	137.10	21.88	2.14
T8ii	56.27	98	21	86	261.27	39.07	5.28
T9ii	57.25	97	24	100	278.25	46.59	11.43
T10ii	3.13	2	2	1	8.13	4.07	0.11
T11ii	4.29	3	2	2	11.29	4.73	0.11

Table 7.4.3.iv

The relative speed of the methods can be seen by comparing the times to describe and solve the subproblems. For the simplex method this time is given by TOTAL and for the assignment method this time is given by SOLVE. For a comparison of the speed of the primal simplex algorithm with the assignment algorithm see the values given by PRIMAL and ASS, respectively. These comparisons indicate that the assignment method provides a means of finding sequences which violate the optimality conditions in appreciably faster times than the simplex method. Also, with the assignment method there is no need to transfer between LAMPS and specialized FORTRAN programs as is the case with the simplex method, compare figures 7.4.2.i and 7.4.3.i. Therefore, it is recommended that the assignment method is adopted in favour of the simplex method for solution of the subproblems. Also, in future applications, the improvement in solution times using

the assignment method suggests that it may be worthwhile using the idea proposed in section 6.3.1 of chapter six, and solving a greater number of subproblems at each iteration of the column generation technique.

CHAPTER EIGHT

INTEGER SOLUTION

8.1 Introduction

Chapter seven establishes a means of finding an optimal solution to the linear programming problem RSPP. If the optimal solution found is integer, then the set of sequences s_r which take the value 1 represent a workable set of schedules for the stock diagramming problem. Where this is not the case it is necessary to use a branch and bound scheme to find an integer solution. This chapter describes the scheme used in this thesis.

Section 8.2 reviews some branch and bound strategies which have proved popular in other scheduling applications. Section 8.3 describes the procedure used in this application. To conclude, the results of implementing the scheme for the data set SET189 are shown in section 8.4.

8.2 Review

For the scheduling problems discussed in [5],[21],[61],[66],[28],[68],[81] the solution method proceeds by first solving the linear programming relaxation of the problem using the technique of column generation. Their differences lie in the means by which the solution method makes the journey from a continuous solution to an integer solution. A common trait amongst recent problems formulated as set-partitioning or set-covering problems is that the conventional variable branching strategy has been abandoned in favour of branching on sets of variables.

Marsten [50] describes an enumerative method for resolving fractionality in large set-partitioning problems which does not involve the selection of a single variable on which to branch. Instead, in Marsten's procedure "...the variables are grouped together in classes and the basic choice involved is which class should be responsible for covering a particular row." The first stage of the procedure is to group the variables into a number of classes. The branching then proceeds by generating a tree in which each branch represents the association of a row of the set-partitioning problem with a class of variables. Once each row has been fixed to a class of variables the decision about an individual variable is automatically determined. This method is subsequently used by Nemhauser *et. al.* [54] and Marsten and Shepardson [51] in set-partitioning applications.

Another branch and bound scheme which has proved popular in set-partitioning and set-covering problems is proposed by Etcheberry [27] in the context of set-covering. The essential idea of this branching scheme, herewith referred to as "constraint branching", is as follows. Consider the set-covering (set-partitioning) problem

$$\text{MIN } \Sigma_{j\in J}\, c_j x_j$$

such that

$$\Sigma_{j\in J}\ a_{ij}x_j \geq 1\ (=1) \qquad \forall\, i\in I$$

$$x_j \in \{0,1\} \qquad \forall\, j\in J$$

where $I=\{1,\ldots,m\}$ is the index set of the m constraints in the problem and $J=\{1,\ldots,n\}$ is the index set corresponding to the columns x_j. The data a_{ij} is given by:

a_{ij} - 1 if i is in column j,
0 otherwise.

Suppose that s and t are the indices of two distinct constraints of the set-covering (set-partitioning) problem and define:

$$F(s)=\{j \mid a_{sj}=1, \forall j \in J\}$$

and

$$F(t)=\{j \mid a_{tj}=1, \forall j \in J\}.$$

Then, the 1-branch is imposed by replacing the constraints indexed by s and t by

$$\Sigma_{j \in F(s) \cap F(t)} x_j \geq 1 \ (=1),$$

and the 0-branch by replacing the constraint set indexed by s with

$$\Sigma_{j \in F(s) \setminus F(t)} x_j \geq 1 \ (=1),$$

and setting

$$x_j=0 \quad \forall j \in F(s) \cap F(t).$$

This scheme is used by Desrochers and Soumis [21], Ryan [66] and Wren *et al.* [82] in crew scheduling applications. Constraint branching is also used by Forbes, Holt and Watts [32] in their work on the stock diagramming problem discussed in this thesis.

The advantage of Marsten's branching scheme and the constraint branching scheme over the conventional variable branching strategy is that branching takes place on a number of variables at each node rather than a single variable. It is clear that for

the conventional variable branching case, although the 1-branch imposes significant structure on the problem, the 0-branch has little effect. On the other hand, constraint branching provides a more balanced branching structure, and this is advantageous for problems where a large number of variables are present. With particular regard to the set-partitioning problem, Ryan [66] states: "It is well known that the conventional variable branch is particularly ineffective in the resolution of fractional solutions which arise at the optimal solution of the relaxed set-partitioning problem." It is for these reasons that the constraint branching scheme is adopted as a means of resolving fractionality arising in the optimal solution of the problem RSPP.

8.3 Constraint Branching Strategy

8.3.1 Interpretation of the Constraint Branch

Define $T(k)=\{r \mid$ sequence s_r is worked by locomotive type k$\}$ and for a particular train and locomotive pair (i,k) consider the value of

$$\Sigma_{r\in T(k)} \delta_{ri} s_r.$$

This expression gives the fraction of train i which is worked by locomotive type k. Now consider an optimal fractional solution at any node of the branching tree and any compatible train and locomotive pair (i,k), three situations can occur.

i) $$\Sigma_{r\in T(k)} \delta_{ri} s_r = 0.$$

The interpretation of this is that no fraction of train i is worked by a locomotive of type k.

ii) $$0 < \Sigma_{j \in T(k)} \delta_{ri} s_r < 1.$$

This shows that train i is partially worked by a locomotive type k, the remainder being worked by other compatible locomotive types.

iii) $$\Sigma_{r \in T(k)} \delta_{ri} s_r = 1.$$

In this case train i is worked by a locomotive type k. Note that this does not necessarily mean that there exists an $r \in T(k)$ such that $s_r = 1$. The s_r with $r \in T(k)$ may still be fractional. Contrast this with Ryan's work [66] where satisfaction of conditions i) and iii) for every possible crew and trip pairing is usually sufficient to ensure an integer solution. Does this mean that for the problem considered in this thesis that even after using a constraint branching to achieve a situation where i) and iii) hold for all possible (i,k) pairs, it will still be necessary to use a variable branching technique to resolve any remaining fractionalities?

To see that this is not the case, consider what the successful implementation of the constraint branching scheme establishes for the stock diagramming problem. For every (i,k) pair which satisfy iii), the question: "Which locomotive should work train i?" is answered. The question remaining to be answered in order to specify a solution set of schedules is: "Which train should be worked after train i?". Ignoring for now the limit on the number of locomotives of each type available, the second question can be answered by solving a series of two-dimensional assignment problems. To show this, consider the following diagrams. Each diagram is a tri-axial representation of simple examples of the stock diagramming problem. In the first example problem there are four trains and a single locomotive type. A feasible solution to this problem is illustrated by the pattern of x's shown at figure 8.3.1.i below.

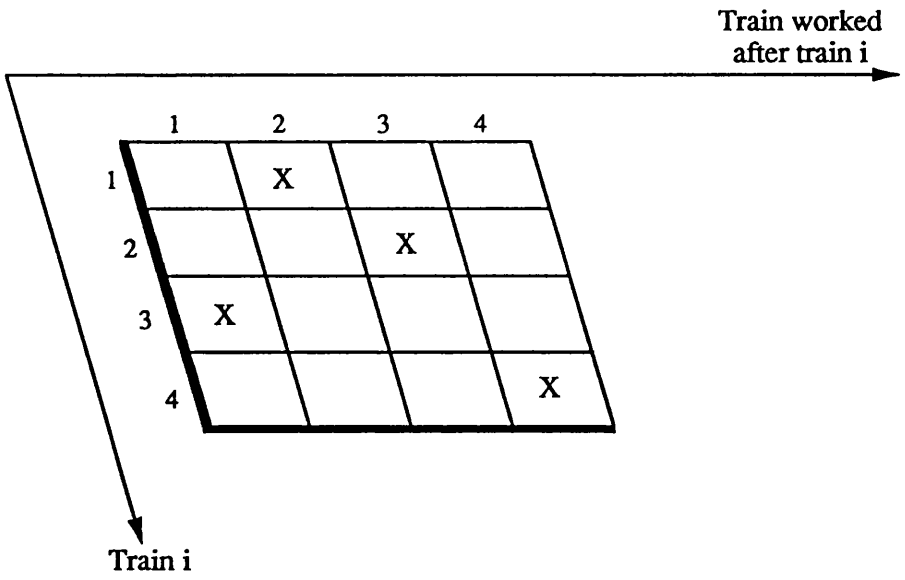


Figure 8.3.1.i

The problem becomes more complicated when trains can be worked by more than one locomotive type. In the second example there are still four trains, but now there are two locomotive types type 1 and type 2. Suppose the pattern of compatibility is:

train	allowed locomotive type
1	1
2	1 2
3	1 2
4	2

Then, the shaded areas of figure 8.3.1.ii indicate where assignments are incompatible on a particular locomotive type for this example.

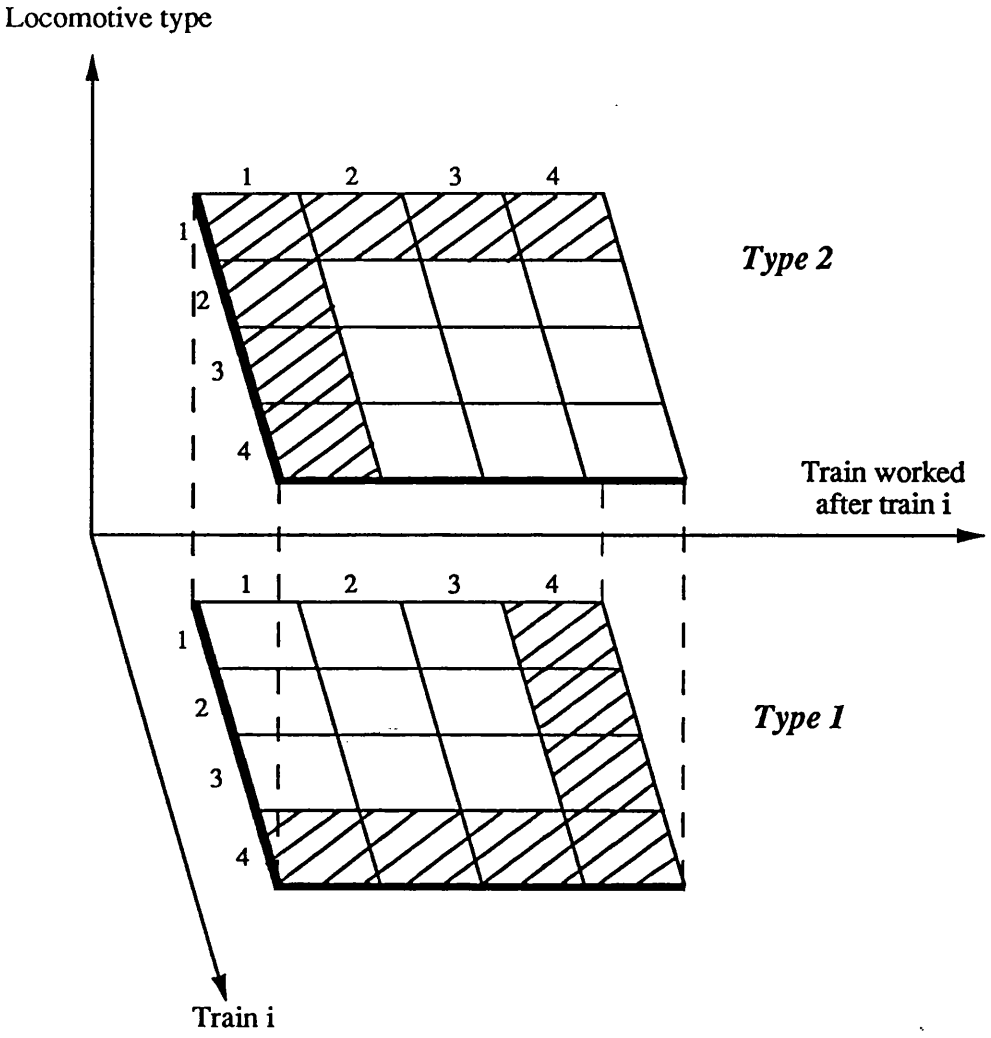


Figure 8.3.1.ii

From the diagram it can be seen that train 1 cannot be worked by locomotive type 2 and train 4 cannot be worked by locomotive type 1. Therefore, the decision about the allocation of a locomotive type to a train is automatic for trains 1 and 4 and the remaining decision problem concerns trains 2 and 3. Suppose that it is arbitrarily decided that train 2 should be worked by locomotive type 1. This automatically rules out the possibility of train 2 and train 4 being in the same schedule and may therefore

render the solution non-optimal or infeasible. In general, the cost preference or feasibility of such an arbitrary allocation of a locomotive type k to a train i is unknown. However, once an optimal solution to the linear programming relaxation of the stock diagramming problem is found, a preference indicated by the value of the expression

$$\Sigma_{r\in T(k)} \delta_{ri} s_r$$

emerges. Provided that a feasible solution exists, by implementing the constraint branching technique a node is reached where for each train i there exists a locomotive type k such that

$$\Sigma_{r\in T(k)} \delta_{ri} s_r = 1.$$

At this node, even though the s_r's may still be fractional, it is known which is the preferred locomotive type for each train i for the solution at this node. So supposing that for the second example given above the preference found is:

train	preferred locomotive type
1	1
2	2
3	1
4	2

Then figure 8.3.1.ii can be redrawn as

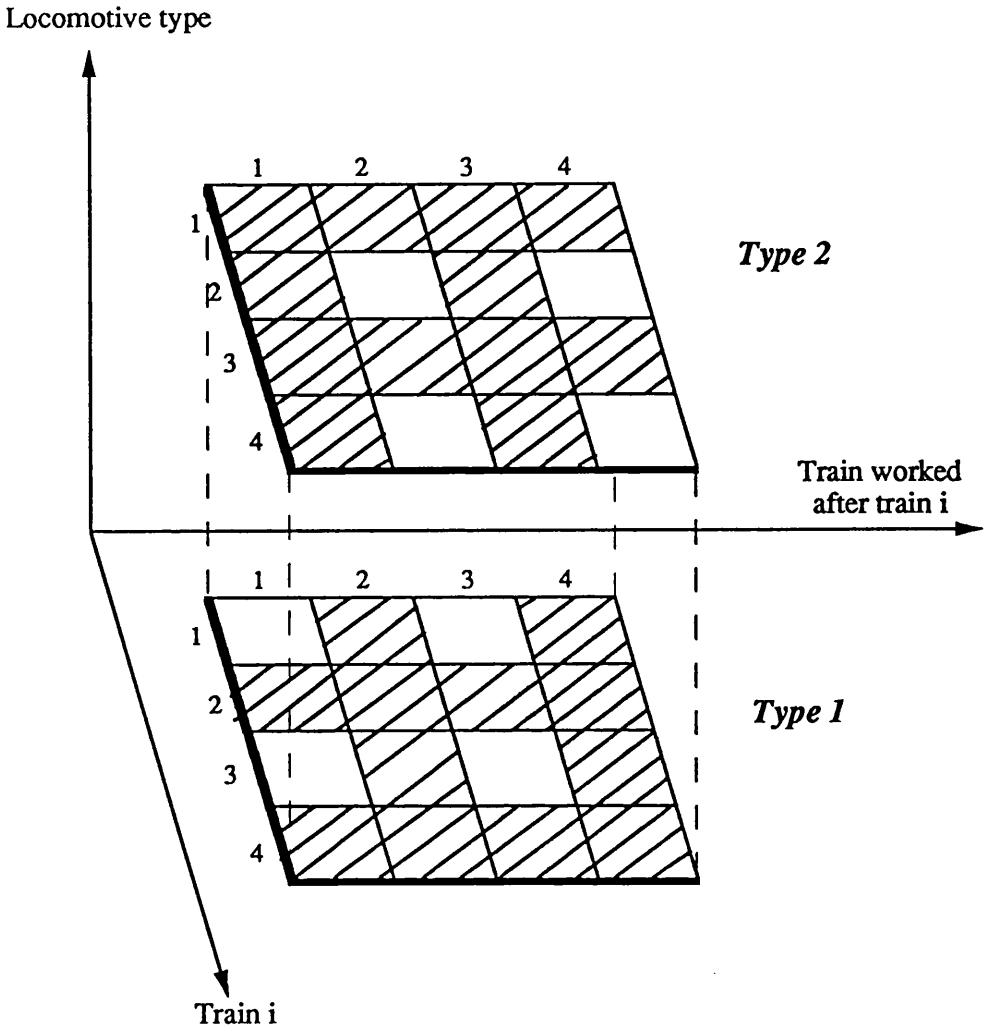


Figure 8.3.1.ii.

Now, solving the assignment problem for each locomotive type in turn gives a solution in which each train is worked exactly once. Moreover, the solution found takes account of cost preferences and feasibility.

Therefore, the primary aim of the constraint branching scheme describe here is to establish which locomotive type should work each train. Once this is done it is a matter of using this information to construct a set of feasible schedules. First consider how the constraint branching scheme is implemented.

8.3.2 Implementation of the Constraint Branching Scheme

At each node of the branch and bound tree, for each train i not already allocated a locomotive type at a node farther up the branch or as a consequence of automatic allocation (i.e. only one locomotive type is compatible), the value of

$$\Sigma_{r\in T(k)}\ \delta_{ri}s_r$$

is computed for each compatible locomotive type k. Then for those train and locomotive pairs (i,k) for which

$$0 < \Sigma_{r\in T(k)}\ \delta_{ri}s_r < 1,$$

the value of

$$f_{max} = \max_{(i,k)} \Sigma_{r\in T(k)}\ \delta_{ri}s_r$$

is calculated. For a train and locomotive pair (i,k) with

$$\Sigma_{r\in T(k)}\ \delta_{ri}s_r = f_{max},$$

the constraint

$$\Sigma_{r\in T(k)}\ \delta_{ri}s_r = 1$$

is added to the master problem to fix train i to locomotive type k. The master problem is resolved from a stored basis using the dual simplex algorithm. The formulation of the master problem RSPP with the new constraints added is as follows.

Let F be the set of pairs with $F=\{(i,k) \mid$ train i is fixed to locomotive type k$\}$. Then the formulation follows:

CSPP

$$\text{MIN } \Sigma_{r\in\Omega} c_r s_r$$

s.t.

$$\Sigma_{r\in\Omega} \delta_{ri} s_r = 1 \qquad \forall\, i\in\tau \quad \textbf{CSPP1}$$

$$\Sigma_{r\in\Omega} L_{rk} s_r \le U_k \qquad \forall\, k\in\Lambda \quad \textbf{CSPP2}$$

$$\Sigma_{r\in T(k)} \delta_{ri} s_r = 1 \qquad \forall\, (i,k)\in F \quad \textbf{CSPP3}$$

$$s_r \ge 0 \qquad \forall\, r\in\Omega \quad \textbf{CSPP4}$$

The sets τ and Λ, the variables and the data are as given for formulation RSPP. The subproblems associated with this revised form of the master problem are now described.

Let $R(k)=\{i \mid$ train i is compatible with locomotive type k and train i has not been fixed to a locomotive type $\overline{k}$, where $\overline{k}\neq k\}$. Suppose that the dual value associated with constraint set CSPP3 of the master problem CSPP is given by f_i, where f_i can take positive or negative values. Then the subproblem for a locomotive type k is:

$\underline{CBS_k}$

$$\text{MIN } \Sigma_{i\in R(k)}\Sigma_{j\in R(k)} (c_{ij}-z_i-f_i-w_k(m_i+M_{ij}))v_{ij}^k$$

s.t.

$$\Sigma_{j\in R(k)} v_{ij}^k=1 \qquad \forall\, i\in R(k) \quad \underline{CBS_k}1$$

$$\Sigma_{j\in R(k)} v_{ji}^k=1 \qquad \forall\, i\in R(k) \quad \underline{CBS_k}2$$

$$v_{ij}^k \geq 0 \qquad \forall\, i,j\in R(k) \quad \underline{CBS_k}3$$

where the variables and the data is as defined for the subproblems given by $\underline{BS_k}$.

The formulation of the subproblems given by $\underline{CBS_k}$ ensures that whenever column generation is used at a node of the branching tree, the subproblems considered comply with the way in which trains have been fixed to locomotives at the current node of the search tree. Therefore, any columns which are added to the master problem CSPP satisfy the constraints CSPP3. It may be necessary to use the column generation technique in the following situations.

To ensure that the value of f_{max} reflects a strong preference of a train for a particular locomotive type, a parameter MINPREF is specified to indicate an acceptable threshold value for f_{max}. For example, a suggested value is MINPREF=0.75. If it happens that the value of f_{max}<MINPREF, then at the current node the column generation procedure is repeated and f_{max} recalculated at each iteration until $f_{max}\geq$MINPREF. Observe that the number of iterations of the column generation procedure at each node can be reduced or bypassed altogether by reducing the value of MINPREF.

The addition of a new constraint may make the problem infeasible, indicating that the limits on locomotive availability have been breached. In this instance, remove the new constraint, i.e. backtrack to the parent node, and implement the column generation procedure. After addition of the columns to the master problem CSPP,

solve the updated problem CSPP and calculate f_{max}. If $f_{max} \geq$ MINPREF, add the constraint which corresponds to this value of f_{max}. Continue in this way until the solution to CSPP, after addition of the new constraint, is feasible.

The pairing of a train i with a locomotive type k is chosen so that the expression

$$\Sigma_{r \in T(k)} \delta_{ri} s_r$$

is maximized and reflects the preference of train i for a locomotive type k in an optimal fractional solution. It is for this reason that the depth first 1-branch is pursued until all trains are allocated a locomotive type and the 0-branch is left unexplored. As the 1-branch becomes deeper the value of f_{max} may fall to 0.5 or less or the solution may exhibit an increased tendency to become infeasible after the addition of the new constraint. As already described above, in such cases further columns are added in the hope of resolving the problem. However, it may happen that no further column generation can take place as there are no new columns which violate the optimality conditions. This indicates that, the linear programming relaxation of the original problem is restored to optimality at the current node. If this occurs the 0-branch at the current node is evaluated and, if necessary, 0-branches farther up the tree are explored.

Observe from the above description, that where column generation takes place it only continues until a solution is found which is feasible, or yields a value of f_{max} which is deemed to be acceptable within the prescribed limits. This implementation contrasts with that of Desrosiers et al. [24] where at each node of the branch and bound tree "...the covering problem is reoptimized generating new routes as needed."

Once a feasible solution is found such that for each train i there exists a locomotive type k with

$$\Sigma_{r\in T(k)}\ \delta_{ri}s_r=1$$

the branching terminates. At this stage the s_r's may be fractional or integer. In the latter case, a solution to the stock diagramming problem is found and the next phase of the procedure can be bypassed. Otherwise, using the partition of trains according to locomotive type which now exists, the next step is to construct a feasible set of schedules.

8.3.3 Construction of Schedules

As described in section 8.3.1 this stage can be viewed as a problem which involves solving a series of assignment problems, one for each locomotive type. Unfortunately, the limit on the number of locomotives available means that the problem is not so straightforward. Suppose that for each locomotive type in turn an assignment problem is constructed with cost matrix $LR_k=(c_{ij})$, where c_{ij} is the light-run time from end location i to start location j and $i,j\in P(k)=\{i \mid$ train i is compatible with locomotive type k$\}$. Then the solution found reflects a situation where light-run cost is minimized without regard for the number of locomotives of type k available. If the constraint on locomotive availability is added the problem becomes NP-hard, and is referred to in Nemhauser and Wolsey [55] as 'a flow problem with budget constraint'. The formulation of the problem for a particular locomotive type k is:

PAP_k

$$\text{MIN } \Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\ c_{ij}v_{ij}^{\ k}$$

s.t.

$$\Sigma_{j\in P(k)}\ v_{ij}^{\ k}=1 \qquad \forall\ i\in P(k)\ \ \mathbf{PAP_k1}$$

$$\Sigma_{j\in P(k)}\ v_{ji}^{\ k}=1 \qquad \forall\ i\in P(k)\ \ \mathbf{PAP_k2}$$

$$\Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\ (m_i + M_{ij})v_{ijk} \le U_k \qquad \mathbf{PAP_k3}$$

$$v_{ij}^{\ k} \in \{0,1\} \qquad \forall\ i,j\in P(k)\ \ \mathbf{PAP_k4}$$

Where the set P(k) and the variables and data are as defined before.

The solutions of the linear programming relaxation of PAP_k are not necessarily integer. However, it is possible to find integer feasible solutions to this problem by solving an assignment problem. As shown in chapter three, the Lagrangean relaxation problem which arises from relaxing constraint set PAP_k3 is an assignment problem solvable in polynomial time. Constraint set PAP_k4 becomes redundant in this problem as the solutions to this assignment problem only take zero-one values. If μ is the Lagrangean multiplier associated with constraint set PAP_k3, then the cost matrix of the assignment problem is $LR_k=(c_{ij}+\mu m_i+\mu M_{ij})$. Recall that m_i is the number of times a locomotive crosses midnight whilst working train i and M_{ij} is the number of times a locomotive crosses midnight whilst performing the light-run from train i to train j. Using a large value for μ and running the assignment problem for each locomotive type in turn a set of schedules are found and a lower bound on the number of locomotives required to work the trains, according to the current partition, is found. If for any of the locomotive types the limit on the number of locomotives of that type available is violated, then this indicates that for the current partition there is no feasible integer solution. If this is the case it is necessary to return to the branch and bound tree and investigate branches as yet unexplored. In the results for the data set SET189 which follow, it is shown that for the schedules constructed in accordance with the first partition found, the constraints on locomotive availability hold.

Once a feasible integer solution is found using the above method, it is then possible to associate a parameter μ_k with each locomotive type and by using different values of μ_k find alternative schedules. This investigation was carried out for the data set SET189 and the results are presented in the following section.

The value chosen for μ_k for each run of the assignment problem was fairly arbitrary, so it was decided that an improved method would be to perform a parametric investigation using a linear programming formulation of the problem.

The mathematical programming package LAMPS allows the user to specify lower and upper bounds for the parametric investigation of the right-hand side of constraint PAP_k3. Continuously varying the value of U_k between these specified limits investigates the trade-off, in terms of light-run time, between the total cost of performing light-runs and using fewer or more locomotives type k. The linear programming relaxation of the problem PAP_k is solved during the parametric investigation. Only the integer solutions are of interest, therefore if a continuous solution is found two steps are taken to locate an integer solution. If the value of U_k is non-integer in the continuous solution, the first step is to round down the value of U_k to the nearest integer value. If the solution remains continuous after reoptimization, the next step is to use the branch and bound facility of LAMPS to find an optimal integer solution for the current value of U_k.

The parametric investigation using the linear programming relaxation was performed for the partition found for SET189 for each locomotive type in turn. It was found that during this investigation none of the solutions found were continuous and so it was never necessary to add cuts or use branch and bound to find integer solutions. The results of this exercise are presented in the next section.

The linear programming relaxation method proved more effective and efficient than the Lagrangean relaxation method at finding alternative integer solutions to the problem PAP_k. The solutions found by continuously varying the value of U_k for SET189 not only included those found when the assignment problem was solved for different values of μ_k, but also included additional integer solutions. Moreover, it is shown in the following section that for SET189 an optimal integer solution was found for every integer value of U_k between the upper and lower limits specified. A more detailed discussion of the relationship between the Lagrangean relaxation and the linear programming relaxation of the problem PAP_k is given in Appendix D.

8.4 Results

8.4.1 Constraint Branching

Discussion of Results

The optimal solution to the linear programming relaxation of the master problem RSPP for data set SET189 was 710 light-run minutes using 99 locomotives. This solution was non-integer and so the branching scheme described in section 8.3.2 was implemented. For this exercise the value of MINPREF was set at 0.75. A partition was found after the 1-branch was explored to a depth of 123 nodes of the branching tree. The solution of the problem CSPP was non-integer with an objective function value of 1181.6257 light-run minutes and a total of 97 of the available 99 locomotives were used. In total it was necessary to perform 25 iterations of the column generation technique. 18 of these iterations were required to satisfy the stipulation that $f_{max} \geq$ MINPREF and 7 iterations were performed when the problem became infeasible.

It was observed that near the root of the tree the value of f_{max} was close to 1 and as a consequence the value of the objective function to the solution of CSPP remained close to 710 as the new constraints were added. Not until node 65 was it necessary to use the column generation technique to restore the value of f_{max}, and feasibility held until node 76.

Clearly, for a problem with 189 trains and given that 27 of these trains are compatible with only a single locomotive type, a branch depth of 123 nodes indicates that a high proportion of trains had to be allocated a locomotive type by means of a constraint branch. However, for the data set SET189 consider the following: each of the 189 trains is compatible with an average of approximately 4 different locomotive types; no distinction, in terms of a train's preference, is made between locomotive

types; and, the costs in the problem are independent of locomotive type. All these factors encourage a breadth of choice which in turn means that the linear programming relaxation of the set-partitioning problem does not exhibit strong integer properties.

When the constraint branching procedure was being developed for this problem one strategy attempted was to fix all those trains, compatible with more than one locomotive type, currently being worked by only one locomotive type. That is, suppose that at any node of the branching tree a train and locomotive pairing (i,k) had not been fixed but

$$\Sigma_{r\in T(k)}\ \delta_{ri}s_r=1,$$

then fix train i to locomotive type k. This has the effect of reducing the number of trains which have to be considered when calculating f_{max}. Also, a number of variables s_r are automatically forced to take the value zero. Unfortunately, it was found that this strategy resulted in the problem becoming infeasible early on in the constraint branching procedure, and it became necessary to backtrack and evaluate a number of 0-branches. It was for this reason that this strategy was abandoned before a partition had been found.

8.4.2 Use of the Partition to Generate Integer Solutions

Assignment Method

The partition found for SET189 was used to construct a feasible set of schedules for the locomotives. The assignment problem was solved for each locomotive type in turn using a large value of μ equal to 6000 light-run minutes. (This figure was chosen arbitrarily.) The results are given at table 8.4.2.i. The key to the column headings is:

'LOCO. TYPE' - the locomotive type being investigated.

'AVAIL' - the number of locomotives of the specified type available.

'USED' - the number of locomotives of the specified type used.

'LR COST' - the total light-run cost of the schedules created for the specified locomotive type.

In the last two rows of the table, totals are given for the number of locomotives available, the number used and the light-run cost incurred by the schedules.

LOCO. TYPE	AVAIL	USED	LR COST
1	14	14	120
2	6	5	0
3	16	16	30
4	6	6	10
5	14	13	300
6	12	12	90
7	20	19	250
8	5	5	60
9	2	2	20
10	4	4	20
TOTAL LOCO.	99	96	-
TOTAL LR	-	-	900

Table 8.4.2.i

These results show that the partition is feasible as AVAIL $\geq$ USED for all locomotive types. The set of schedules found for this value of μ uses 96 locomotives

and 900 minutes of light-run time.

The value of μ was then varied for each locomotive type and alternative solutions found. A solution which used more locomotives than the solution shown in table 8.4.2.i but had a reduced light-run time total is given at table 8.4.2.ii. The columns are the same as those of 8.4.2.a with one addition:

μ_k - the value of the parameter μ chosen for locomotive type k.

LOCO. TYPE	μ_k	AVAIL	USED	LR COST
1	60	14	14	120
2	60	6	5	0
3	60	16	16	30
4	60	6	6	10
5	50	14	13	300
6	60	12	12	90
7	30	20	20	200
8	60	5	5	60
9	60	2	2	20
10	60	4	4	20
TOTAL LOCO.		99	97	-
TOTAL LR		-	-	850

Table 8.4.2.ii

Comparing these results with those of table 8.4.2.i, an extra locomotive type 7 has been used to reduce the light-run cost by 50 minutes to 850 minutes.

Parametrics

The next set of tables summarizes the results of performing the parametric investigation procedure, described in 8.3.3, on the partition found for data set SET189. In the headings preceding each table the locomotive type k is named followed by three parameters indicating: the number of locomotives available, this is the original value of U_k; the lower bound for the parametric variation of U_k; and, the upper bound for the parametric variation of U_k. The key to the columns in the table is:

'USED' - the number of locomotives used by the solution.

'LR COST' - the total light-run cost of the solution.

Only feasible solutions are given. Also, a solution is not given for every value of U_k between its upper and lower bounds, only those solutions for which the total light-run cost changes are reported.

Locomotive type 10, U_{10}=4, Lower bound=1, Upper bound=8

USED	LR COST
4	20

Table 8.4.2.iii

Locomotive type 9, U_9=2, Lower bound=1, Upper bound=4

USED	LR COST
2	20

Table 8.4.2.iv

Locomotive type 8, U_8=5, Lower bound=1, Upper bound=8

USED	LR COST
5	60
6	40

Table 8.4.2.v

Locomotive type 7, U_7=20, Lower bound=1, Upper bound=30

USED	LR COST
19	250
20	200

Table 8.4.2.vi

Locomotive type 6, U_6=12, Lower bound=1, Upper bound=12

USED	LR COST
12	90
13	30

Table 8.4.2.vii

Locomotive type 5, $U_5=14$, Lower bound=1, Upper bound=21

USED	LR COST
13	300
14	260
15	220
16	200

Table 8.4.2.viii

Locomotive type 4, $U_4=6$, Lower bound=1, Upper bound=9

USED	LR COST
6	10

Table 8.4.2.ix

Locomotive type 3, $U_3=16$, Lower bound=1, Upper bound=24

USED	LR COST
16	30

Table 8.4.2.x

Locomotive type 2, U_2=6, Lower bound=1, Upper bound=9

USED	LR COST
5	0

Table 8.4.2.xi

Locomotive type 1, U_1=14, Lower bound=1, Upper bound=21

USED	LR COST
14	120
15	90
16	70

Table 8.4.2.xii

According to this investigation there is a feasible set of schedules which has a total light-run cost of 810 light-run minutes and uses a total of 98 locomotives. This is the minimum cost solution which uses no more than the available locomotives for this partition. This investigation also indicates where savings can be achieved by increasing the number of locomotives available. For example, increasing the availability of each of the locomotive types 1,5,6 and 8 by 1 would give a solution which uses 102 locomotives at a cost of 660 light-run minutes.

8.5 Comments and Future Work

It is felt that the constraint branching phase of this solution method is an area which warrants further study. One possibility may be to use the data on preferred locomotive type which is provided for each train, but no longer used in practice by

British Rail. It would then be possible to bias the choice of locomotive type for each train by including a preference weighting as a component of the costs in the objective function. A further suggestion concerns the possibility of reducing the time taken to perform the branching by starting at the root with a restricted master problem. The argument for this is that the greatest proportion of the columns in the master problem are created by the heuristic procedure carried out prior to column generation, and many of these columns may remain inactive throughout the implementation of column generation and constraint branching procedure. Neither of these ideas have been tested here.

The results presented in section 8.4.2 are only valid for the partition being considered, and for an alternative partition the feasibility and potential savings would differ. This suggests that a further area of research may be to develop a swap heuristic to investigate the savings which might be made by allocation of trains to alternative locomotive types and thereby develop an alternative partition. At the start of a swap procedure, it is suggested that the information provided by the dual values associated with the problem CSPP could be used to decide where a swap might take place. For example, suppose a train i has been fixed to a locomotive type k and that the dual value f_i associated with constraint set CSPP3 in CSPP has the most negative value. This suggests that fixing train i to locomotive type k is expensive in the current solution. Therefore when deciding which train-locomotive pairings to swap, unfix train i from locomotive type k and fix train i to an alternative compatible locomotive type $\overline{k}$. Again it is suggested that the information on the marginal cost of the locomotive types in the current solution to CSPP is used to decide which locomotive type $\overline{k}$ should be chosen. If $\overline{k}$ is compatible with train i and $w_{\overline{k}}$ has a least negative value then choose locomotive type $\overline{k}$. Whether or not such a procedure is worth pursuing for a particular partition depends on how good the partition is believed to be. Unless the solution found at the end of Phase II is integer, as the optimal integer solution is not known, the only yardstick for assessing the quality of a partition is the value of the continuous solution found at the end of Phase II. To make the

comparison for the data set SET189, the value of the continuous solution at the end of Phase II was 710 light-run minutes for a solution which used 99 locomotives. For the partition found for SET189, the best feasible solution cost 810 light-run minutes and used 98 locomotives. It is not possible to compare this result with the result that British Rail found for the SET189 problem for two reasons. Firstly, British Rail used more locomotives of type 6 than there are available. Secondly, during the solution of the problem, the British Rail analyst changed the values of the performance times for some of the trains in the timetable, and this effectively alters the number of days required to make connections between trains. Given that an aim of this thesis is to override the need for such manual intervention it is not reasonable to make such changes when using the method described here. Also it is not possible to make such changes without having a detailed knowledge of what is and what is not permitted when altering a given timetable.

CHAPTER NINE

TESTING THE METHOD

9.1 Introduction

Chapters six, seven and eight describe a method for solving the stock diagramming problem. During the development of the method the data set SET189 is used as a test set. In this chapter the method used to solve SET189 undergoes refinement and a number of additional data sets are solved.

In section 9.2 the data sets are introduced and some necessary assumptions are discussed. Sections 9.3, 9.4, 9.5 and 9.6 cover the four stages of the method: the generation of an initial set of sequences; the column generation technique for the Phase I and Phase II procedures; the constraint branching scheme; and the parametric investigation. Section 9.7 comments on the results found. The chapter concludes with section 9.8 which summarizes the final method proposed for the solution of the stock diagramming problem.

9.2 Data Sets

9.2.1 Description of Data Sets

The data used in this chapter have been provided by Dr. M. Wright and are used by Wright [83] and Forbes *et. al.* [32] in their work on the stock diagramming problem. Hereafter, in this thesis, these data sets are referred to as Wright's data sets. The data include all the required information on the trains to be scheduled, i.e. the start and end times of each train; the start and end location of each train; and, the compatible locomotive types for each train. Unfortunately, for each of Wright's data sets no information is provided on the number of locomotives of each type available

as Wright does not use this information (Wright [84]). Clearly, for a real world problem there is a limit on the number of locomotives available, and so the problem becomes unrealistic and simplified if the constraint on the number of locomotives is omitted. In order to make Wright's data sets realistic, a tight limit on the number of locomotives of each type available for a data set has been approximated.

9.2.2 Calculation of Locomotive Availability

For any stock diagramming problem it is possible to find a lower bound on the number of locomotives required to work the trains in the data set. To find this minimum a 'connection' matrix, which incorporates the condition on locomotive compatibility, is created for the set of trains in the data set. The connection matrix indicates for all connections between each pair of trains whether or not the connection is permissible. So, if train i is only compatible with a locomotive type 1 and train j is only compatible with locomotive type 2 the elements (i,j) and (j,i) in the connection matrix will take the value ∞, indicating that the connection is not allowed. If a connection between the train i and train j is permissible, i.e. they have a locomotive type in common, then the value of element (i,j) is equal to the number of days a locomotive would take to make the connection from the start of train i to the start of train j. Following the notation of previous chapters, element (i,j) equals m_i+M_{ij}. Where m_i is the number of times a locomotive crosses midnight whilst working train i and M_{ij} is the number of times a locomotive crosses midnight whilst performing the light-run from the end of train i to train j. Having created the connection matrix, an assignment problem is solved over the matrix to give the required lower bound LB_{TOT}. Clearly, the figure found is a lower bound and not necessarily the minimum number of locomotives required to work the set of trains. This can be shown by use of an example. Suppose that in the solution to the assignment problem the following assignment is made

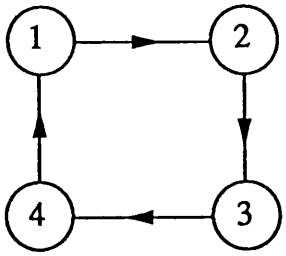


Figure 9.2.2.i

and that train 1 is compatible with locomotive type 1 only, trains 2 and 4 are compatible both with locomotive types 1 and 2, and train 3 is compatible with locomotive type 2 only. According to the connection matrix the connections shown in the diagram are permissible. However, such a schedule will never be allowed in a solution to the stock diagramming problem as this subset of trains has no locomotive type in common.

For data set SET189 the total number of locomotives available is known and equal to 99. Using the above procedure it was found that 92 is a lower bound on the number of locomotives required for SET189. In preceding chapters, SET189 was solved based on an availability of 99 locomotives and, although a solution using only 92 locomotives was not found during the parametric investigation, solutions which used fewer than 99 locomotives were found, indicating a surplus in the number of locomotives required. For Wright's data sets, as the upper bound on locomotive availability is not known, it was decided that the lower bound should be used instead and increased only if it appears that there is no feasible solution for this number of locomotives. (The way in which the number of locomotives is increased is discussed

later in this chapter.) As before, the objective is to minimize the total light-run cost.

9.2.3 Distribution of Locomotives

For the stock diagramming problem it is not sufficient simply to specify the total number of locomotives available; information on the number of locomotives of each type available is also required. For each of Wright's data sets a formula was used to specify the number of locomotives of each type, namely:

$$\frac{\text{(no. of trains compatible with locomotive type k)}}{\text{(total no. of trains)}} \times LB_{TOT}.$$

The decision about the way in which locomotive types are distributed may automatically make the problem infeasible. For instance, if the number of locomotives of type k is insufficient to work those trains which can only be worked by a locomotive type k, then it is clear that the problem is infeasible. Therefore, having used the above formula to devise a distribution, further checks need to be made to ensure the integrity of this distribution. For each locomotive type k_1 a proper subset of trains is defined by the condition 'trains which can be worked by a type k_1 locomotive only', and a connection matrix for this subset created. The assignment problem is then solved for this matrix to find a lower bound on the number of type k_1 locomotives required to work those trains which are only compatible with type k_1. It is then verified that the number of locomotives of type k_1 available is greater than this lower bound. The calculation is also made for proper subsets based on two locomotive types, i.e. the connection matrix is created for all proper subsets based on the condition 'trains which can be worked by type k_1 and type k_2 locomotives but no other type'. A check is then made to ensure that the total number of type k_1 and type k_2 locomotives is sufficient. Finally, this process is repeated for combinations of three locomotive types. These checks can be made for all combinations of up to K-1 locomotive types, where K is the number of locomotive types in the data set, but

it is believed that a check of up to three types is adequate. If it happens that the values of the lower bounds found during these checks are greater than the number of locomotives available, then the problem is infeasible and it is necessary to change the way in which the locomotives are distributed among the types before proceeding.

9.2.4 Summary of the Data Sets

Table 9.2.4.i below gives a brief description of each of Wright's data sets. The key to the table is:

'NAME' - gives the name by which the data set is referred to in this thesis.

'TRAIN' - gives the number of trains in the data set.

'LOCO' - gives the number of different types of locomotive in the data set.

'LB_{TOT}' - gives the value of LB_{TOT} for the data set.

'TYPE k' - gives the number of locomotives of type k available.

NAME	TRAIN	LOCO	LB_{TOT}	TYPE 1	TYPE 2	TYPE 3	TYPE 4	TYPE 5
WR50_3	50	3	34	4	17	13	-	-
WR50_5	50	5	33	4	1	10	8	10
WR75_5	75	5	52	5	1	17	14	15
WR100_5	100	5	64	7	2	20	17	18
WR150_5	150	5	97	12	4	28	26	27

Table 9.2.4.i

In each of the regions covered by Wright's data sets there are 32 stations. None of Wright's data sets include any gap trains. Note that for the data sets solved

in this chapter, none of the light-run times have been rounded down to the nearest ten nor have any of the stations been grouped to form an area (this includes single stations).

9.3 Generation of an Initial Set of Columns

Recall from chapter four that before starting the column generation procedure it is necessary to create an initial set of sequences Ω. For the solution of data set SET189 the heuristic procedures HAP and ModHAP described in chapter four were used to create initial sets of sequences. However, chapter four also shows that the heuristic procedure ModHAP performs better than HAP for SET189 and that LOCOST also performs well for SET189. Based upon the results found for SET189, a combination of the heuristics ModHAP and LOCOST is used to create the sets of sequences Ω for the data sets considered in this chapter. For each data set all singletons are included in Ω at the outset then additional sequences are created using ModHAP and LOCOST for a number of values of the parameters INTERVAL and μ, respectively.

Table 9.3.i gives the number of sequences created using the above method and the solution found when the linear programming problem RSPP is solved using LAMPS. RSPP is the linear programming relaxation of the set-partitioning problem SPP. The formulation of the problem SPP is given in chapter four and repeated here.

SPP

$$\text{MIN } \Sigma_{r\in\Omega}\, c_r s_r$$

s.t.

$$\Sigma_{r\in\Omega}\, \delta_{ri} s_r = 1 \qquad \forall\, i\in\tau \quad \textbf{SPP1}$$

$$\Sigma_{r\in\Omega}\, L_{rk} s_r \le U_k \qquad \forall\, k\in\Lambda \quad \textbf{SPP2}$$

$$s_r \in \{0,1\} \qquad \forall\, r\in\Omega \quad \textbf{SPP3}$$

Where the set $\tau=\{i \mid$ train i is the i^{th} train in the timetable$\}$, the set $\Lambda=\{k \mid k$ is the number of the locomotive type$\}$ and r is the index of the r^{th} sequence in the set Ω.

The variables are:

s_r - 1 if sequence r is used,
- 0 otherwise.

The data elements are:

δ_{ri} - is a binary constant; 1 if sequence r includes train i,
- 0 otherwise;

c_r - cost of sequence r, i.e. the sum of the light-run costs which make up the sequence r;

L_{rk} - the number of locomotives of type k required to work sequence r;

U_k - the total number of locomotives of type k available to work the timetable.

The key to table 9.3.i is as follows:

'NAME' - gives the name of the data set.

'SEQ' - gives the total number of sequences generated.

'INT' - gives the number of values of INTERVAL used.

'μ' - gives the number of values of μ used.

'OBJ' - indicates the value of the optimal solution to RSPP, and is equal to INF if the problem is infeasible.

'SUM INF' - if the problem is infeasible this shows the sum of the infeasibilities.

NAME	SEQ	INT	μ	OBJ	SUM INF
WR50_3	528	7	7	INF	1.1010
WR50_5	1693	7	7	INF	3.1837
WR75_5	2505	7	7	INF	5.0989
WR100_5	3210	7	7	INF	7.6216
WR150_5	2772	5	5	INF	13.8392

Table 9.3.i

Notice that the number of values of INTERVAL and μ used for the last data set was reduced. This was done to limit the number of columns created so as to reduce the amount of computer memory used.

9.4 Column Generation - Phase I and Phase II

For each of Wright's data sets, the subproblems used during the column generation technique were based on type sets for which the type set specification was: "Consider all combinations of length=1 of the locomotive types". The effect of this specification is that, if a data set has K different locomotive types, there are 2K type sets and consequently 2K subproblems are created. Some of the type sets may be empty, in which case they can be ignored. (For a detailed discussion on how the type set specification affects the number of subproblems created for the column generation technique see chapter six.)

9.4.1 Phase I

The solution to the problem RSPP, for the initial set of sequences, is infeasible for each of the data sets. Therefore, the next stage is to find a feasible solution using the column generation technique described in chapter seven to solve the Phase I

problem. The results are given in table 9.4.1.i and the key to the table is as follows:

'NAME' - gives the name of the data set.

'SEQ' - gives the number of sequences in the set Ω on completion of Phase I.

'OBJ' - gives the value of the objective function of RSPP when a feasible solution is found.

NAME	SEQ	OBJ
WR50_3	555	6312.00
WR50_5	1753	4121.75
WR75_5	2663	6568.62
WR100_5	3361	6727.26
WR150_5	3192	9662.00

Table 9.4.1.i

Phase I was completed successfully for each of the data sets. However, if a feasible solution is not found this indicates that the availability of a least one of the locomotive types has to be increased.

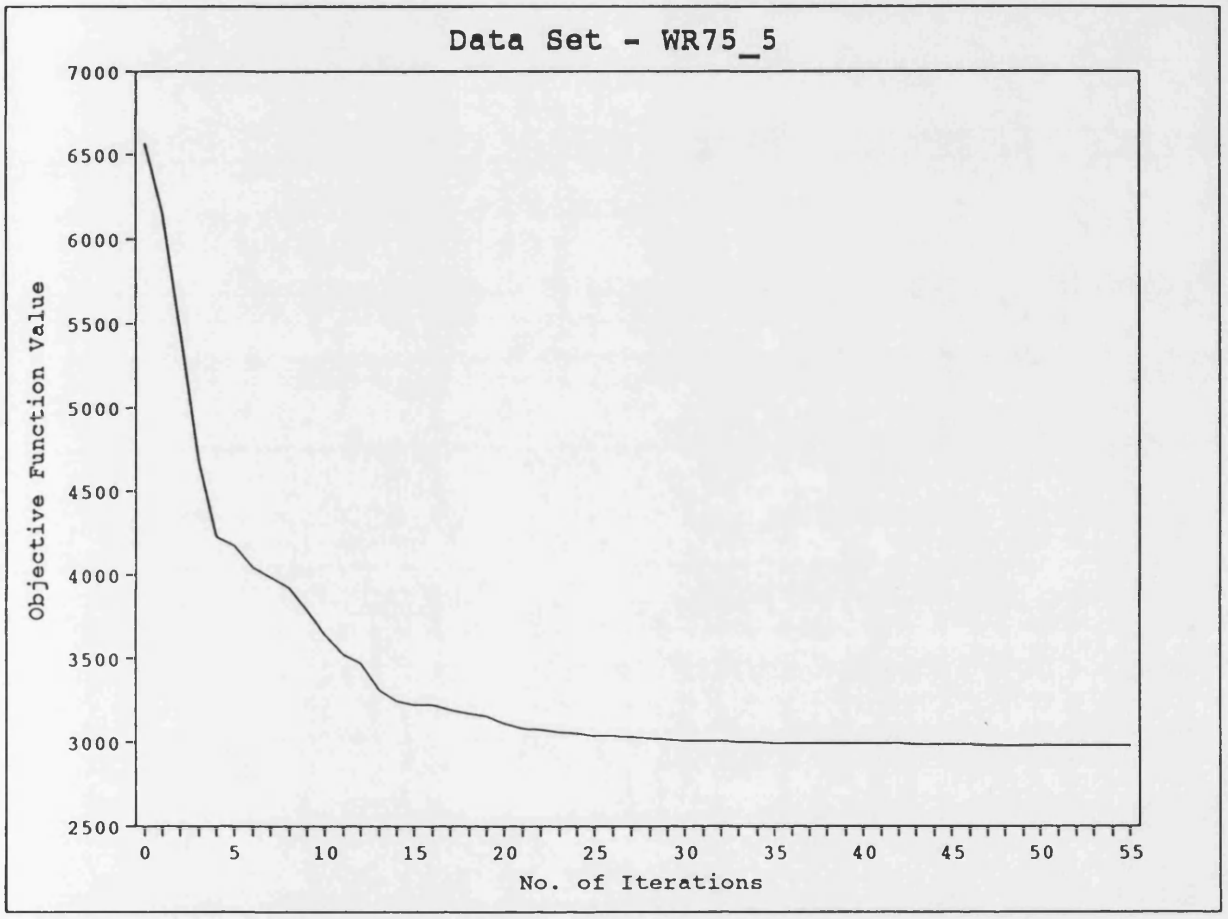
9.4.2 Phase II

In Phase II an optimal solution is found for each of the data sets. The column generation procedure used for Phase II is the same as that used for Phase I, and is described in chapter seven.

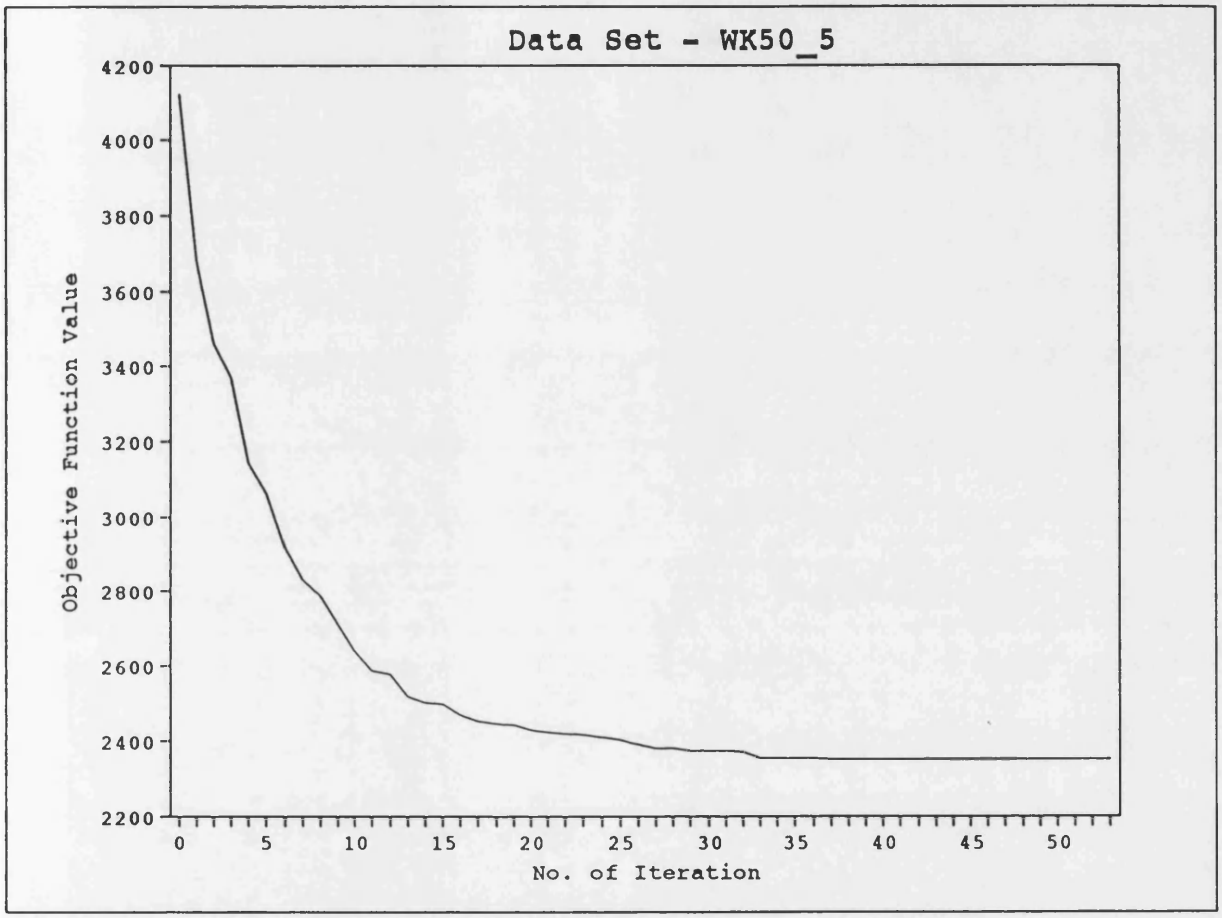
It is stated in chapter seven that during the solution of SET189 it was observed that the problem was cycling for a number of iterations of the column generation technique. This tendency is discussed here and a means of dealing with the problem of cycling is proposed.

The column generation procedure alternates between a master problem and a set of subproblems. (See chapter five for a description of the procedure.) One iteration of the column generation procedure involves: solving the current set of subproblems; adding a new set of variables to the master problem; resolving the master problem. In this thesis, the solutions to the subproblems are valid sequences which may be added to the master problem given by RSPP. A non-trivial (i.e. non-zero) solution to a single subproblem may produce a single sequence or many sequences. If the solution to at least one of the subproblems has a value less than zero, then this indicates that there exists a sequence which if introduced to the problem RSPP has an associated reduced cost which violates the condition for optimality. Suppose that such a sequence s_n has a reduced cost with an absolute value of η ($\eta > 0$). If s_n is introduced into the basis at level θ ($\theta > 0$) and the current value of the solution to the problem RSPP is Δ then the new value will be $\Delta - \eta\theta$. However, when the current solution is degenerate, i.e. at least one of the basic variables is 0, it is possible that s_n may enter the basis at value 0. This is a degenerate iteration and the effect is that the value of RSPP remains at Δ. So, barring degeneracy, the value of the problem RSPP decreases after one iteration of the column generation procedure. However, in the presence of degeneracy, it is possible for the solution value to remain the same for a number of iterations of the column generation procedure. If the solution value remains the same and in addition the solution is repeated after a number of iterations of the column generation technique, then the problem is said to 'cycle'. It was observed during the execution of the Phase II procedure for the data sets that degeneracy was causing the column generation procedure to cycle in this way. This tendency to cycle was not observed initially but

occurred more frequently as the drop in the objective function between iterations of the column generation procedure became less significant. Cycling was often resolved within less than five iterations, but for some data sets it continued for more than ten iterations. It was noted that when such prolonged cycling occurred the magnitude of the violation in the optimality conditions for each of the subproblems was less than 0.05% of the current value of the objective function for at least one iteration of the column generation procedure. Therefore, weighing up the potential decrease in the objective function against the possibility of continuing to cycle, it was decided that in such instances the Phase II procedure should be stopped. So, it is proposed that the second criterion (the first being that optimality has been proven) for stopping the Phase II procedure is: if the solution value has remained the same for ten iterations of the column generation procedure and, for at least one of these iterations, the magnitude of the violation in the optimality conditions for each of the subproblems is less than 0.1% of this solution value, then the Phase II procedure should be stopped. The solution value when the Phase II procedure is interrupted cannot be guaranteed optimal. However, it is believed that it is highly probable that the solution found is optimal or near-optimal. This belief is supported by considering the following graphs. Graphs 9.4.2.i and 9.4.2.ii plot the number of an iteration against the objective function value at that iteration. Graph 9.4.2.i shows the decrease in the objective function for during the Phase II procedure for data set WR75_5. For this data set Phase II was halted as the first stopping criterion was met, i.e. the solution satisfied the optimality conditions. Graph 9.4.2.ii refers to the Phase II procedure for data set WR50_5. For this data set Phase II was halted as the second stopping criterion was met. The pattern of decrease in graph 9.4.2.ii mirrors that of graph 9.4.2.i, and the flattening off of the graphs suggests that the solution is near-optimal in the second case.


Data Set - WR75_5
Objective Function Value
7000
6500
6000
5500
5000
4500
4000
3500
3000
2500
0
5
10
15
20
25
30
35
40
45
50
55
No. of Iterations

Graph 9.4.2.i


Data Set - WK50_5
Objective Function Value
4200
4000
3800
3600
3400
3200
3000
2800
2600
2400
2200
0
5
10
15
20
25
30
35
40
45
50
No. of Iteration

Graph 9.4.2.ii

The results of the Phase II procedure for the data sets considered in this chapter are given in table 9.4.2.i. The key to the table is as follows:

'NAME' - gives the name of the data set.

'SEQ' - gives the number of sequences in the set Ω at the termination of Phase II.

'OBJ' - gives the value of the objective function on completion of Phase II.

'OPTIMAL' - a 'NO' in the column indicates that Phase II was stopped because the second stopping criterion was satisfied, and a 'YES' indicates that optimality was proven.

'INTEGER' - indicates whether or not the solution found was integer.

NAME	SEQ	OBJ	OPTIMAL	INTEGER
WR50_3	1089	2947.00	NO	YES
WR50_5	2085	2349.00	NO	NO
WR75_5	3140	2976.40	YES	NO
WR100_5	3992	3277.67	YES	NO
WR150_5	5044	5149.00	NO	NO

Table 9.4.2.i

Although the solution found for data set WR50_3 has not been proven optimal, it is integer and so a partition of trains according to locomotive type has been found. The next stage in the solution method for data set WR50_3 is to perform the parametric investigation. For the remaining data sets the constraint branching strategy needs to be invoked to find the partition of trains by locomotive type. The implementation of the branching scheme for these data sets is described in the next

section.

9.5 Constraint Branching

Two main changes to the constraint branching scheme discussed in chapter eight have been investigated using Wright's data sets as test sets.

The first change is aimed at investigating an alternative way of imposing the constraint branch. Recall that the set T(k) is defined to be T(k)={r | sequence s_r is worked by locomotive type k}. Then for a train i and a locomotive type k pairing, train i is forced to be worked by locomotive type k by adding the constraint

$$\Sigma_{r\in T(k)} \delta_{ri} s_r = 1 \qquad (9.5.1)$$

to the problem RSPP, and train i is forced to be worked by a locomotive type other than type k by adding the constraint

$$\Sigma_{r\in T(k)} \delta_{ri} s_r = 0 \qquad (9.5.2).$$

Under the alternative strategy, there is an upper bound B_r associated with each variable s_r in the problem RSPP. During the solution of RSPP for the initial set of sequences and for Phases I and II the value of B_r remains fixed at 1, but once the constraint branching stage is reached the above system of adding constraints to the problem is emulated by simply adjusting the bounds on the variables. To see how this is done, let T(k) be as defined above and define $\overline{T}(k)$={r | sequence s_r is not worked by locomotive type k}. Then, for a train i and a locomotive type k pairing, if train i is to be worked by locomotive type k set $B_r=0$ if $\delta_{ri}=1$ and $r\in\overline{T}(k)$ and this has the same effect as adding constraint (9.5.1). Similarly, if train i is not to be worked by locomotive type k set $B_r=0$ if $\delta_{ri}=1$ and $r\in T(k)$ and this has the same effect as adding constraint (9.5.2). In a sense the variables are being switched on and off as

required, and this is similar to the way in which Ryan [66] imposes the constraint branches.

The advantage of using the upper bounds method is that no additional constraints need to be added to the master problem RSPP. As with the constraint branching procedure described in chapter eight, new formulations are given for the master problem and the subproblem associated with this constraint branching scheme. The formulation for the master problem is the same as RSPP except that upper bounds have been introduced for the variables s_r. These upper bounds are stated explicitly in constraint set USPP4 of the new formulation USPP.

USPP

$$\text{MIN } \Sigma_{r\in\Omega} c_r s_r$$

s.t.

$\Sigma_{r\in\Omega} \delta_{ri} s_r = 1$	$\forall\, i\in\tau$ **USPP1**
$\Sigma_{r\in\Omega} L_{rk} s_r \le U_k$	$\forall\, k\in\Lambda$ **USPP2**
$s_r \ge 0$	$\forall\, r\in\Omega$ **USPP3**
$s_r \le B_r$	$\forall\, r\in\Omega$ **USPP4**

Where the sets, variables and data are as given for formulation RSPP and B_r is the value of the upper bound placed on variable s_r. The value of B_r is given by:

B_r -
- 0 if δ_{ri} and $r\in T(\overline{k})$, where $\overline{k}\neq k$ and $(i,k)\in F$;
- 1 otherwise.

Where $T(k)=\{r \mid s_r$ is worked by locomotive type k$\}$ and $F=\{(i,k) \mid$ train i is fixed to locomotive type k$\}$.

The subproblem for locomotive type k is given by:

$\underline{UBS_k}$

$$\text{MIN } \Sigma_{r\in R(k)}\Sigma_{r\in R(k)} (c_{ij}-z_i-w_k(m_i+M_{ij}))v_{ij}^k$$

s.t.

$$\Sigma_{j\in R(k)} v_{ij}^k=1 \qquad \forall\, i\in R(k) \quad \underline{UBS_k}1$$

$$\Sigma_{j\in R(k)} v_{ji}^k=1 \qquad \forall\, i\in R(k) \quad \underline{UBS_k}2$$

$$v_{ij}^k \geq 0 \qquad \forall\, i,j\in R(k) \quad \underline{UBS_k}3$$

Where the variables and data are as defined for $\underline{BSP_k}$ and the set of trains $R(k)=\{i \mid$ train i is compatible with locomotive type k and train i is not fixed to locomotive type $\overline{k}\neq k\}$.

There are two disadvantages with the upper bounds method. Firstly, with the constraint method each new constraint has an associated dual value and this provides an insight into where fixing a train to particular locomotive type is proving costly. As stated at the end of chapter eight this information may prove useful if in future work a swap heuristic were developed to analyze alternative partitions. Secondly, unfixing a train-locomotive pairing when using the constraint method is done by removing a constraint from the problem CSPP, and this does not affect the way in which other trains have been fixed to locomotive types. However, when the upper bounds method is used this reversal is not so straightforward. To see this, consider two sets A and B and let $A=\{s_r \mid \delta_{ri}=1$ and $r\in T(k)\}$ and $B=\{s_r \mid \delta_{rj}=1$ and $r\in T(k)\}$. If train i and train j must not be worked by locomotive type k all the variables in the set $A\bigcup B$ have upper bounds equal to 0. If $A\bigcap B\neq 0$, simply resetting to 1 the upper bounds on the variables $s_r\in A$ also changes the status of the variables in $A\bigcap B$. Therefore it must be ensured that only the variables in $A\backslash B$ have their upper bounds reset to 1. This makes backtracking more difficult than in the constraint method. Weighing up the advantages and disadvantages of the two methods it is felt that the constraint method is preferable

and should be used in any future work on this problem.

The second change concerns the way in which infeasibility arising during branching is dealt with. The strategy described in chapter eight is to remove the most recently added constraint, then use the Phase II column generation procedure to add additional sequences in the hope that the constraint added after an iteration of the column generation method would leave the master problem CSPP feasible. This process is cumbersome in that at each iteration a new constraint has to be constructed and added with no guarantee of an improvement in feasibility. The new strategy proposed here is to use the Phase I column generation procedure to restore the problem to feasibility at the current node without ever changing the way in which trains are fixed to locomotive types. As well as being easier to implement, this alternative method reveals immediately whether or not it is feasible to fix a particular train to a locomotive type. With the old strategy it is possible to travel farther down the tree before discovering that this is the case. It is therefore proposed that for future work this new method of dealing with infeasibility should be adopted.

The above conclusions regarding the viability of the upper bounds method and the use of a Phase I procedure to resolve fractionality were made after the two new strategies were tested on Wright's data set. A constraint branching scheme, which uses the upper bounds method and resolves feasibility using the Phase I procedure, was implemented for each of the Wright's data sets with fractional solutions at the end of Phase II. As with SET189 the value of MINPREF was set at 0.75 for each of the data sets. For Wright's data sets it was decided that if $f_{max} < \text{MINPREF}$ at the current node of the tree, the (i,k) pairing for which $f_{max} = \max_{(i,k)} f(i,k)$ is accepted if after four iterations of the column generation technique the value of f_{max} remains less than MINPREF. This was done to limit the total number of iterations of the column generation technique carried out during the constraint branching procedure, and is also an effective means of dealing with instances of cycling which may occur as a result of degeneracy. It is proposed that this practice is adopted in any future work.

Before presenting the results of the constraint branching scheme, it is necessary to make a further note which applies only to Wright's data and arises as a result of the incompleteness of the data sets. In the case of data sets for which the limit on the availability of locomotives is known, if an infeasibility at a node can not be resolved, it is necessary to evaluate the 0-branch at that node and 0-branches farther up the tree if necessary. However, for the data sets discussed in this chapter, where an infeasibility can not be resolved the first step taken is to adjust the number of locomotives of each type available without increasing the total locomotive availability. Exactly where the adjustments are made is decided by considering how the train being fixed is currently being worked. For instance, if train i is to be fixed to locomotive type k_1 but a proportion of the train is worked by locomotive type k_2, then the number of locomotives type k_2 is decreased and the number of locomotives type k_1 increased. If it is not possible to restore feasibility using this scheme, then the next step is to increase the total number of locomotives available by increasing by one the number of locomotives type k_1. The master problem USPP, with the new limits on locomotive availability, is then resolved and the procedure repeated if the solution remains infeasible.

The results of implementing the branching strategy are presented in table 9.5.i. The key to the table is:

'NAME'	-	gives the name of the data set.
'OBJ'	-	gives the value of the objective function value of the problem USPP when branching stops.
'TOTAL'	-	gives the total number of locomotives available.
'TYPE k'	-	gives the number of locomotives type k available.

NAME	OBJ	TOTAL	TYPE 1	TYPE 2	TYPE 3	TYPE 4	TYPE 5
WR50_5	2395.00	33	3	1	9	8	12
WR75_5	3129.00	53	6	1	16	15	15
WR100_5	3998.80	64	9	2	18	17	18
WR150_5	5149.00	97	12	4	28	26	27

Table 9.5.i

In all cases, except data set WR100_5, the solution when a partition was established was integer. A comparison of table 9.5.i with table 9.2.4.i shows that, during branching, it was only necessary to increase the total number of locomotives available for data set WR75_5. The distribution of the locomotive types has been altered for data sets WR50_5, WR75_5 and WR100_5.

The following table 9.5.ii provides information on the number of nodes visited, and the number of iterations of the column generation procedure performed during branching. The key to the table is:

'NAME' - gives the name of the data set.

'SEQ' - gives the number of sequences in the set Ω when branching stops.

'NODES' - gives the number of nodes investigated.

'PHASE I' - gives the number of iterations of the column generation technique when the Phase I procedure is used.

'PHASE II' - gives the number of iterations of the column generation technique when the Phase II procedure is used.

NAME	SEQ	NODES	PHASE I	PHASE II
WR50_5	2299	27	17	25
WR75_5	3328	55	20	6
WR100_5	4304	62	30	6
WR150_5	5044	1	0	0

Table 9.5.ii

9.6 Parametric Investigation

Having found a partition for each of the data sets the next step is to perform a parametric investigation. With the exception of data set WR100_5, the integer solutions found indicate that a solution exists which uses no more than the number of locomotives available. (See table 9.5.i for the current status of locomotive availability.) The parametric investigation is carried out in the same way as described in chapter eight. For each of the data sets the results and comments are presented in the following tables. For the sake of brevity, the variation in the objective function as the availability of each locomotive type is changed is not shown. Instead, the minimum cost solution found as the total number of locomotives changes is given. Remember that these results relate to the way in which the trains in the data sets have been partitioned and are not global. Each table consists of three columns:

'TOTAL' - shows the total number of locomotives used.

'OBJECTIVE' - gives the minimum cost solution for the number of available locomotives.

'DECREASE' - shows the decrease in the cost incurred by increasing the value of TOTAL by one.

WR50_3

TOTAL	OBJECTIVE	DECREASE
< 34	INF	-
34	2947.00	-
35	2632.00	315
36	2546.00	86
37	2480.00	66
38	2437.00	43
39	2430.00	7
40	2425.00	5

Table 9.6.i

If fewer than 34 locomotives are available the solution is infeasible. This supports the earlier finding that 34 represents a lower bound on the number of locomotives required to schedule the trains in this data set. Notice that the value of OBJECTIVE when 34 locomotives are available agrees with the value found on completion of Phase II. This is especially encouraging as Phase II was stopped when the second stopping criterion was satisfied, and therefore the solution was not proven optimal. Of course, this is still not a guaranteed optimal solution as an alternative partition could give a better result.

WR50_5

TOTAL	OBJECTIVE	DECREASE
< 33	INF	-
33	2375.00	-
34	2290.00	85
35	2177.00	73
36	2086.00	91
37	2033.00	53
38	2013.00	20
39	1993.00	20
40	1978.00	15

Table 9.6.ii

The value of LB_{TOT} found for this data set has been verified. The value of OBJECTIVE found for 33 locomotives uses 2375.00 light-run minutes compared with the value of 2395.00 light-run minutes found when branching stopped, a saving of 20.00 light-run minutes.

WR75_5

TOTAL	OBJECTIVE	DECREASE
< 53	INF	-
53	2909.00	-
54	2548.00	361
55	2382.00	166
56	2229.00	153
57	2209.00	20
58	2189.00	20

Table 9.6.iii

The total number of locomotives available was increased during the branching procedure from the lower bound value of 52 to 53. However, the above results show that, for this partition, it is not possible to schedule the trains using fewer than 53 locomotives. At the end of the branching procedure the cost of the solution found was 3129.00, but after the parametric investigation it was found that there was a cheaper solution which used only 2909.00 light-run minutes and 53 locomotives, a decrease of 220.00 light-run minutes.

WR100_5

TOTAL	OBJECTIVE	DECREASE
< 64	INF	-
64	3645.00	-
65	3270.00	375
66	2918.00	352
67	2672.00	246
68	2449.00	223
69	2371.00	78
70	2304.00	67
71	2253.00	51
72	2206.00	47
73	2186.00	20
74	2166.00	20
75	2146.00	20
76	2126.00	20
77	2111.00	15
78	2101.00	10
79	2091.00	10
80	2088.00	3

Table 9.6.iv

The value $LB_{TOT}=64$ has been verified for this partition, but more importantly a solution which uses no more than 64 locomotives has been proved to exist. Unlike the other data sets, this result was not automatic as the solution found when branching stopped was non-integer. As was the case for data set WR75_5 the table shows a reduction in the light-run cost from 3998.80, at the end of branching procedure, to a value of 3645.00, a reduction of 353.80 light-run minutes.

WR150_5

TOTAL	OBJECTIVE	DECREASE
< 97	INF	-
97	5149.00	-
98	4733.00	416
99	4459.00	274
100	4312.00	147
101	4193.00	119
102	4079.00	114
103	4014.00	65
104	3956.00	58
105	3931.00	25
106	3911.00	20
107	3891.00	20
108	3871.00	20
109	3851.00	20
110	3836.00	15
111	3826.00	10
112	3816.00	10
113	3806.00	10
114	3800.00	6
115	3795.00	5

Table 9.6.v

The value of LB_{TOT} found has been verified. The value of OBJECTIVE for 97 locomotives agrees, not only with that found at the end of the branching procedure, but also with that found at the end of Phase II.

9.7 Discussion

It is interesting at this stage to compare these results with those found by Forbes *et. al.* [32]. Overlooking the assumptions made by Forbes *et. al.* (discussed in chapter three), and comparing solutions which use the same number of locomotives it is clear that for each data set they have found a cheaper solution. This shows that for the above data sets there is an alternative partition or distribution of locomotives among the locomotive types which gives a better solution. This is bound to be the case, as the solutions found at the end of Phase II have light-run costs greater than those found by Forbes *et. al.* This highlights the way in which the distribution of the locomotive types affects the achievable solution values. It is also worth remembering that when the information on the locomotive availability is given for each locomotive type the Forbes *et. al.* method cannot be guaranteed to find a feasible solution. In contrast, the method described in this thesis will find a feasible solution or show that a feasible solution does not exist. It is not possible to make a comparison with the solutions found by Wright [83], as Wright attaches a cost to the use of a locomotive and gives the solution values as the sum of the light-run costs and the locomotive costs without indicating how many locomotives are used.

9.8 A Method for Solving the Stock Diagramming Problem

The development and refinement of the solution method for the stock diagramming problem is now complete, and in this section a summary of the final method is given. As already stated the method can be considered as a four stage procedure. The stages are:

i) the generation of an initial set of sequences;

ii) column generation using Phase I and Phase II procedures;

iii) constraint branching;

and, iv) parametric investigation of the partition of trains according to locomotive type.

Stage i)

In the initial set of sequences all singleton sequences are first included, and additional sequences are then generated using a combination of the heuristic procedures ModHAP and LOCOST. Refer to chapter four for the description of these heuristics, and this chapter for an example of how this scheme is implemented in practice. Based on this set of sequences the linear programming relaxation of the set-partitioning problem, referred to as RSPP, is solved.

Stage ii)

An optimal (or near-optimal) solution to RSPP is found using the column generation technique described in chapter seven. RSPP is the master problem for the column generation technique and the associated subproblems are solved using the assignment method described in chapter six. A Phase I procedure is used if the solution to RSPP found at stage i) is infeasible. Once a feasible solution is found, Phase II is then used to find an optimal or near-optimal solution.

Stage iii)

The constraint branching scheme is used to find a partition of trains according to locomotive type. It is not required that an integer solution is found. Branches are imposed by adding constraints to the master problem. A description of how these 1-branch and 0-branch constraints are constructed is given in chapter eight.

The branching scheme is implemented with a depth-first 1-branch and the 0-branch left unfathomed as the 1-branch is constructed to reflect the preference of a particular train for a locomotive type. This preference is ensured at each node of the search tree by choosing a pairing (i,k) so that the value of

$$f(i,k)=\Sigma_{r\in T(k)}\delta_{ri}s_r,$$

where $f(i,k)<1$ and $T(k)=\{r \mid s_r$ is worked by locomotive type k$\}$, is maximized. Let f_{max} be the maximum value of f(i,k) found. To maintain a strong preference on the 1-branch a parameter MINPREF is specified at the start of branching and only a pairing for which $f_{max}\geq$MINPREF is accepted. (For the data sets solved in this thesis the value of MINPREF was chosen to be 0.75). The 1-branch at the current node then forces train i to be worked by its preferred locomotive type k. If f_{max} falls below the threshold value MINPREF then one iteration of the column generation technique for the Phase II procedure is carried out, and f_{max} recalculated. If after four iterations of the column generation technique the value of f_{max} remains less than MINPREF, the (i,k) pairing for which $f_{max}=\max_{(i,k)}f(i,k)$ is then accepted. The subproblems for the column generation technique are modified so that they reflect the way in which trains are fixed at the nodes farther up the current branch of the search tree. It may happen that the value of f_{max} indicates that the train i does not have a preferred locomotive type. For example, suppose that for a train i and for two (or more) locomotive types k_1 and k_2 compatible with train i $f_{max}=f(i,k_1)=f(i,k_2)$. If this is the case then arbitrarily select one of the locomotive types k with $f(i,k)=f_{max}$ and fix train i to locomotive type k.

If on addition of a 1-branch constraint the solution to the master problem becomes infeasible the column generation technique is implemented for the Phase I procedure to restore the problem to feasibility. If the master problem remains infeasible the 0-branch at that node is evaluated, and if necessary, 0-branches farther up the tree are also explored until a feasible solution is found. Branching then continues as before, evaluating only the 1-branch at each node.

Once the value of

$$\Sigma_{r\in T(k)}\ \delta_{ri}s_r$$

is equal to 0 or 1 for all (i,k) pairs, a partition of trains by locomotive types has been established and branching stops.

The constraint branching procedure is summarized at figure 9.8.i.

Stage iv)

A parametric investigation of the partition of trains according to locomotive type is performed. A method used to perform this analysis is described in chapter eight.

The method for the solving the stock diagramming problem is summarized at figure 9.8.ii. The four stages of the method are indicated on the diagram.

It is possible to extend the solution method and obtain alternative partitions by evaluating all nodes of the branching tree. A further possibility is to use the column generation procedure at each node to restore the linear programming relaxation of the original problem to optimality. This is done by Desrosiers *et. al* [24]. However, it is suggested that the extra work required to perform such a complete analysis is not worthwhile as an optimal or near-optimal solution is found to the problem RSPP and the branching procedure maximizes the preference exhibited at each node of a train for a particular locomotive type. Hence, the solution method proposed in this thesis is not an exact method, but it has its roots in optimality making it a powerful heuristic.

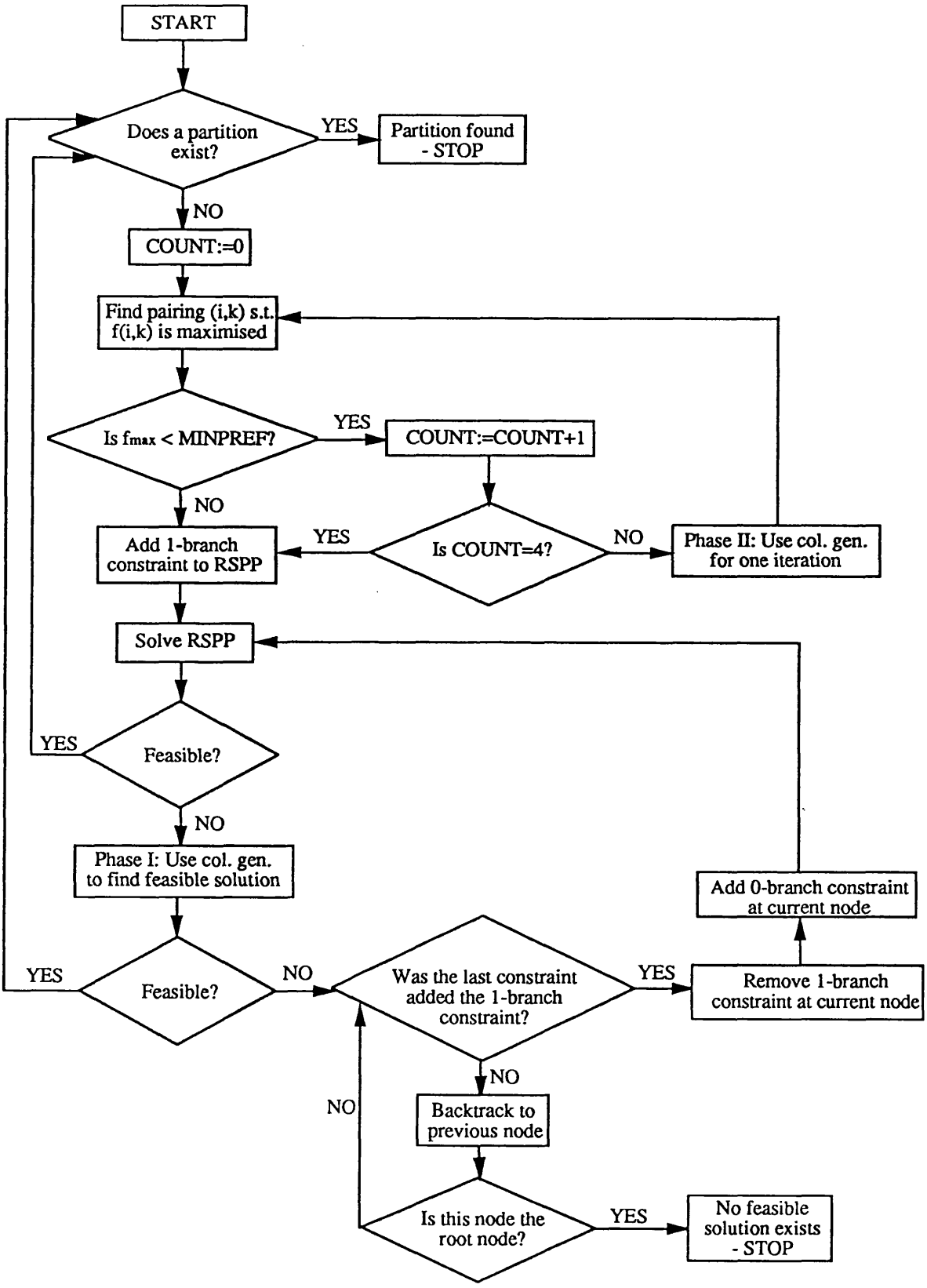

START
Does a partition exist?
YES
Partition found - STOP
NO
COUNT:=0
Find pairing (i,k) s.t. f(i,k) is maximised
Is fmax < MINPREF?
YES
COUNT:=COUNT+1
NO
Add 1-branch constraint to RSPP
YES
Is COUNT=4?
NO
Phase II: Use col. gen. for one iteration
Solve RSPP
YES
Feasible?
NO
Phase I: Use col. gen. to find feasible solution
Add 0-branch constraint at current node
YES
Feasible?
NO
Was the last constraint added the 1-branch constraint?
YES
Remove 1-branch constraint at current node
NO
NO
Backtrack to previous node
Is this node the root node?
YES
No feasible solution exists - STOP

Figure 9.8.i

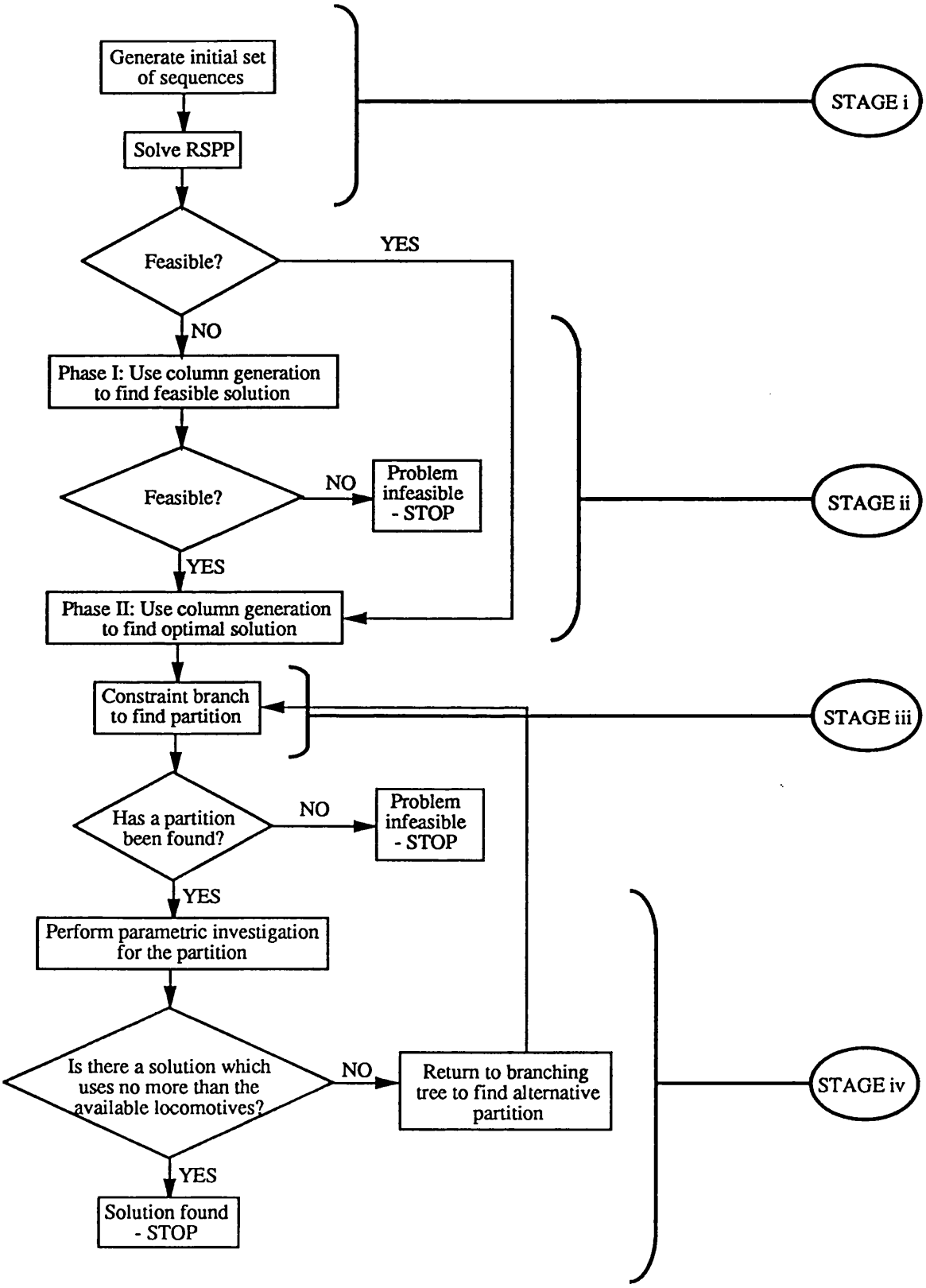

Generate initial set of sequences
Solve RSPP
Feasible?
YES
NO
Phase I: Use column generation to find feasible solution
Feasible?
NO
Problem infeasible - STOP
YES
Phase II: Use column generation to find optimal solution
Constraint branch to find partition
Has a partition been found?
NO
Problem infeasible - STOP
YES
Perform parametric investigation for the partition
Is there a solution which uses no more than the available locomotives?
NO
Return to branching tree to find alternative partition
YES
Solution found - STOP
STAGE i
STAGE ii
STAGE iii
STAGE iv

Figure 9.8.ii

CHAPTER TEN

CONCLUDING REMARKS

10.1 Summary

In this thesis a real world scheduling problem faced by British Rail, the stock diagramming problem, has been considered. Recall the description of the stock diagramming problem. Given a fixed timetable, which is assumed to be repeated on a daily basis, the problem is to work all the trains in the timetable using a non-homogeneous set of locomotives. There is a limited number of locomotives of each type available. Each train is not necessarily compatible with all the available locomotive types thus, for each train, information is provided which specifies those locomotive types compatible with the train. The objective is to schedule all trains in the timetable so as to minimize the total light-run time used without violating the limits on the locomotive availability. The solution to the stock diagramming problem is a set of cyclic schedules. Each schedule describes the order in which the trains should be worked and the type of locomotive which works the trains in the schedule. The set of schedules cover each train in the timetables exactly once. It may take one or more locomotives of a given locomotive type to complete a schedule.

The number of trains and the number of locomotive types in a stock diagramming can give rise to a large combinatorially complex problem. In this thesis a method is provided which successfully tackles the size of the problem and thereby makes a PC-based solution method viable. This method will find a good feasible solution to the stock diagramming problem if such a solution exists. If a feasible solution does not exist, then the method will prove that the problem is infeasible. These facts contrast with the methods developed by other authors who have considered this problem. As well as British Rail's attempts to solve the stock diagramming problem, this problem has received attention from Wright [83] and

Forbes *et. al.* [32]. British Rail and Wright tackle the size of the problem by use of heuristic procedures which break the problem down into components of a manageable size. These heuristic procedures can give rise to good feasible solutions. Unfortunately, such heuristic procedures may also fail to find a feasible solution. In such instances it is not clear whether the problem is infeasible or the heuristic inadequate. Forbes *et. al.* propose an exact method, but they do not suggest a practical means of dealing with the size of the problem nor do they explicitly incorporate the constraint on locomotive availability.

The method used to solve the problem is summarized in chapter nine. The stock diagramming problem is formulated as an integer linear programming problem. This problem is then decomposed into a number of subproblems and a coupling problem, the master problem. Instead of including all possible columns (schedules) in the master problem at the outset, the feasible region is limited by considering only a subset of the set of all possible columns. The method proceeds by solving the linear programming relaxation of the master problem to optimality by using a column generation technique. Additional columns are introduced into the master problem as required. It is the use of this technique which makes the method accessible to large problems. In this thesis the master problem is solved using the simplex algorithm. It was discovered that the subproblems in the column generation technique can be formulated as assignment problems and solved in polynomial time. This use of a polynomial time algorithm to solve the set of subproblems significantly reduces the time taken to execute one iteration of the column generation technique. This also allows for the possibility of solving an increased number of subproblems at each iteration. Once the optimal solution to the linear programming relaxation of the master problem is found, if the solution is not integer, a constraint branching procedure is invoked. The purpose of the constraint branching procedure is to find a partition of the trains according to locomotive type. That is, at the end of the constraint branching procedure each train is assigned a locomotive type. The solution at the end of the procedure may or may not be integer. If the solution is not integer,

instead of using a variable branching procedure to find which schedules cover each train an alternative means has been devised. Based on the set of trains worked by a specific locomotive type an assignment problem is constructed. The assignment problem includes an additional constraint on the number of locomotives of the specified type available. The solution of the assignment problem is a set of schedules which cover the trains in the assignment problem exactly once. In this thesis the assignment problem, with the added constraint, is solved using the simplex algorithm. This method allows alternative schedules to be generated by considering the multiple solutions which may exist for the given number of locomotives. The solutions found are optimal integer solutions at the current node of the branching tree. Also, by varying the number of locomotives of each type available, solutions which use more or fewer locomotives are found. This is the parametric investigation phase of the solution method.

The method proposed in this thesis for the stock diagramming problem may be used to find optimal solutions to the problem. However, it is suggested that it is more practical in a real world problem to find a good feasible solution in a reasonable amount of computing time. Therefore, the column generation technique may be halted before an optimal solution is found to the relaxed linear programming formulation of the master problem and not all branches of the constraint branching tree are explored. The final method summarized in chapter nine proposes that, barring degeneracy, the optimal solution to the relaxed master problem is pursued and, barring infeasibility at a node, only the 1-branch of the branching tree is explored at each node. Hence, this method may best be described as an 'optimal heuristic' procedure.

10.2 Future Work

In this section extensions to the methodology summarized in chapter nine along with suggestions for future work are listed.

1) The next stage in the development of the method is to computerize the entire procedure. This means incorporating the modules of LAMPS in a single program which includes: the use of the column generation technique for Phase I and Phase II procedures; the constraint branching procedure; and, the parametric investigation of the partition.

2) A further development is to devise a swap heuristic to investigate alternative partitions of trains to locomotive types. This idea is mentioned in chapter eight.

3) As mentioned in section 10.1, an extension of the method is to find optimal solutions to the stock diagramming problem. This entails finding an optimal solution to the linear programming relaxation of the master problem at every node of the constraint branching tree. The optimal solution is found by using the column generation technique. In general, the search for an optimal solution will involve a significant increase in the amount of computing time used to solve the problem. For a particular stock diagramming problem it is worth considering the trade-off between the reduction in the objective function and the increase in the computing time required to achieve the reduction. If relatively few nodes of the constraint branching tree are explored before a partition is found, then it is likely that the optimal integer solution at the current node is the optimal or near-optimal integer solution of the original problem.

4) In addition to minimising the light-run cost in a solution, it is also possible to try and minimize the total number of locomotives used. This can be done by the addition of a constraint which limits the total number of locomotives used. For example, the additional constraint could specify that no more than the peak requirement of locomotives are used. Recall that the peak requirement is a lower bound on the number of locomotives required to schedule the trains. (See chapters four and nine for a description and method of calculating the peak requirement.) The

limit on total availability can then be relaxed if the problem is infeasible at the end of Phase I or if the problem becomes infeasible during constraint branching. Instead of adding a constraint which limits the total number of locomotives used it is also possible to tighten the limits on availability for selected locomotive types. For an example of the implementation of this idea see chapter nine. For Wright's data sets, the limits on locomotive availability were chosen so that the total number of locomotives available was no more than the peak requirement. It was only necessary to increase this limit for one of the five data sets and this was done during constraint branching. This strategy is of use if the scheduler wishes to investigate locomotive type requirements for a given timetable.

6) It may be possible to improve the speed of the column generation technique by restricting the total number of columns in the master problem. This can be done by employing a strategy to delete a subset of the non-basic variables in the master problem. A typical strategy is described in chapter seven when the shortest path method is used to solve the subproblems. In this strategy, at each iteration of the column generation technique, all non-basic variables are deleted before the addition of new columns to the master problem. Dantzig [18] also gives a number of strategies for restricting the master problem and these are reported in chapter five. For the final method proposed in this thesis all columns are retained in the master problem as relatively few columns are added during the use of the column generation technique.

REFERENCES

1. Appelgren, L.H. (1969) A Column Generation Algorithm for a Ship Scheduling Problem. *Transportation Science* **3**, 53-68.

2. Assad, A.A. (1978) Flows - A Survey. *NETWORKS* **8**, 37-91.

3. Balas, E. and Toth, P. (1985) Branch and Bound Methods. *The Travel ing Salesman Problem*, Ed. Lawler, E.L., Lenstra, J.K., Rinnooy, A.H.G. and Shmoys, K.D.B., 361-397. John Wiley and Sons Ltd, New York.

4. Ball, M. and Magazine, M. (1981) The Design and Analysis of Heuristics. *NETWORKS* **11**, 215-219.

5. Bellmore, M., Bennington, G. and Lubore, S. (1977) A Multivehicle Tanker Scheduling Problem. *Transportation Science* **5**, 36-47.

6. Benders, J.F. (1962) Partitioning Procedures for Solving Mixed-Variables Programming Problems. *Numerische Mathematik* **4**, 238-252.

7. Berge, C. (1972) Balanced Matrices. *Mathematical Programming* **2**, 19-31.

8. Bertossi, A.A., Carraresi, P. and Gallo, G. (1987) On Some Matching Problems Arising in Vehicle Scheduling Models. *NETWORKS* **17**, 271-281.

9. Bodin, L.D. and Golden, B.L. (1981) Classification in Vehicle Routing and Scheduling. *NETWORKS* **11**, 97-108.

10. Bodin, L., Golden, B., Assad, A. and Ball, M. (1983) Routing and Scheduling of Vehicles and Crews: The State of the Art. *Computers and Operations Research* **10**, 63-211.

11. Bott, K. and Ballou, R.H. (1986) Research Perspectives in Vehicle Routing and Scheduling. *Transport Research - A* **20a**, 239-243.

12. Carpaneto, G., Dell'amico, M., Fischetti, M. and Toth, P. (1989) A Branch and Bound Algorithm for Multiple Depot Vehicle Scheduling Problem. *NETWORKS* **19**, 531-548.

13. Carpaneto, G. and Toth, P. (1986) Some New Branching and Bounding Criteria for the Asymmetric Traveling Salesman Problem. *Management Science* **26**, 736-743.

14. Carraresi, P. and Gallo, G. (1984) Network Models for Vehicle and Crew Scheduling. *European Journal of Operational Research* **16**, 139-151.

15. Cohn, P.M. (1974) *Algebra - Volume 1*, John Wiley and Sons, New York.

16. Crainic, T.G. and Rousseau, J-M. (1987) The Column Generation Principle and the Airline Crew Scheduling Problem. *INFOR* **25**, 136-151.

17. Cullen, F.H., Jarvis, J.J. and Ratliff, H.D. (1981) Set Partitioning Based Heuristics for Interactive Routing. *NETWORKS* **11**, 125-143.

18. Dantzig, G.B. (1963) *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, U.S.A.

19. Dantzig, G.B. and Fulkerson, D.R. (1954) Minimizing the Number of Tankers to Meet a Fixed Schedule. *Naval Logistics Quarterly* **1**, 217-222.

20. Dantzig, G.B. and Wolfe, P. (1959) Decomposition Principle for Linear Programs. *Operations Research* **8**, 101-111.

21. Desrochers, M. and Soumis, F. (1989) A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science* **23**, 1-13.

22. Desrosiers, J., Dumas, Y. and Soumis, F. (1988) The Multiple Vehicle Dial-a-Ride Problem. In *Lecture Notes in Economics and Mathematical Systems 308: Computer-Aided Transit Scheduling*, Ed. Daduna, J.R. and Wren, A., 15-27.

23. Desrosiers, J. and Soumis, F. (1983) Locomotive Maintenance Scheduling and Train Assignment (Extended Abstract). *Methods of Operations Research* **45**, 217-219.

24. Desrosiers, J., Soumis, F. and Desrochers, M. (1984) Routing with Time Windows by Column Generation. *NETWORKS* **14**, 545-565.

25. Desrosiers, J., Soumis, F., Desrochers, M. and Suavé, M. (1986) Vehicle Routing and Scheduling with Time-Windows. *Mathematical Programming Study* **26**, 249-251.

26. Dexter, K.L.W. (1981) Scheduling an Urban Railway. In *Computer Scheduling of Public Transport*, Ed. Wren, A., North Holland Publishing Co., Amsterdam, 147-180.

27. Etcheberry, J. (1977) The Set-Covering Problem: A New Implicit Enumeration Algorithm. *Operations Research* **25**, 760-772.

28. Falkner, J.C. and Ryan, D.M. (1988) Aspects of Bus Crew Scheduling Using a Set Partitioning Model. In *Lecture Notes in Economics and Mathematical Systems 308: Computer-Aided Transit Scheduling*, Ed. Daduna, J.R. and Wren, A., 91-103.

29. Ferland, J.A. and Michelon, P. (1988) The Vehicle Scheduling Problem with Multiple Vehicle Types. *Journal of the Operational Research Society* **31**, 577-583.

30. Fisher, M.L. (1981) The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science* **27**, 1-18.

31. Fisher, M.L. and Jaikumar, R. (1981) A Generalized Assignment Heuristic for Vehicle Routing. *NETWORKS* **11**, 109-124.

32. Forbes, M.A., Holt, J.N. and Watts, A.M. (1991) Exact Solution of Locomotive Scheduling Problems. *Journal of the Operational Research Society* **42**, 825-831.

33. Ford, L.R. and Fulkerson, D.R. (1957) A Suggested Computation for Maximal Multicommodity Network Flows. *Management Science* **5**, 97-101.

34. Foster, B.A. and Ryan, D.M. (1976) An Integer Programming Approach to the Vehicle Scheduling Problem. *Operational Research Quarterly* **27**, 367-384.

35. Geetha, S. and Vartak, M.N. (1989) Time-Cost Trade-Off Analysis in Some Constrained Assignment Problems. *Journal of the Operational Research Society* **40**, 97-101.

36. Geoffrion, A.M. (1970) Elements of Large Scale Mathematical Programming Part I: Concepts. *Management Science* **16**, 652-675.

37. Geoffrion, A.M. (1970) Elements of Large Scale Mathematical Programming Part II: Synthesis of Algorithms and Bibliography. *Management Science* **16**, 676-691.

38. Golden, B., Assad, A., Levy, L. and Gheysens, F. (1982) The Fleet Size and Mix Vehicle Routing Problem. *Management Science and Statistics* Working Paper Series MS/S 82-020.

39. Hartley, T. and Wren, A. (1985) Two Complementary Bus Scheduling Programs. In *Computer Scheduling of Public Transport 2*, Ed. Rousseau, J-M., Elsevier Science Publishers B.V., North-Holland, 345-369.

40. Klee, V. and Minty, G.J. (1972) How Good is the Simplex Algorithm? In *INEQUALITIES III*, Ed. Shisha, O., New York: Academic Press, Inc., 159-175.

41. LAMPS User Manual, Version 1.66 (1991). Advanced Mathematical Software, Yukon Court, 4 Yukon Road, London. SW12 9PU.

42. Lasdon, L.S. (1970) *Optimization Theory for Large Systems*. The Macmillan Co., New York.

43. Lavoie, S., Minoux, M. and Odier, E. (1988) A New Approach for Crew Pairing Problems by Column Generation with and Application to Air Transportation. *European Journal of Operational Research* **35**, 45-58.

44. Lawler, E.L. (1976) Integrality of Flows and the Unimodular Property. In *Combinatorial Optimization: Networks and Maitroids*, Holt, Rinehart and Winston, U.S.A., 160-165.

45. Lemke, C.E., Salkin, H.M. and Spielberg, K. (1971) Set Covering by Single-Branch Enumeration with Linear Programming Subproblems. *Operations Research* **19**, 998-1022.

46. Levary, R.R. (1981) Heuristic Vehicle Scheduling. *OMEGA* **9**, 660-663.

47. Levin, A. (1970) Scheduling and Fleet Routing Models for Transportation Systems. *Transportation Science* **5**, 232-255.

48. Lin, S. (1965) Computer Solutions to the Traveling Salesman Problem. *The Bell System Technical Journal* **44**, 2245-2269.

49. Lundy, M. and Mees, A. (1986) Convergence of an Annealing Algorithm. *Mathematical Programming* **34**, 111-124.

50. Marsten, R.E. (1971) An Algorithm for Large Set Partitioning Problems. *Management Science* **20**, 774-787.

51. Marsten, R.E. and Shepardson, F. (1981) Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications. *NETWORKS* **11**, 167-177.

52. Mitra, G. and Darby-Dowman, K. (1985) CRU-SCHED - A Computer Based Bus Crew Scheduling System Using Integer Programming. In *Computer Scheduling of Public Transport 2*, Ed. Rousseau, J-M., Elsevier Science Publishers B.V., North-Holland, 223-232.

53. Mitra, G. and Welsh, A.P.G. (1981) A Computer-Based Crew Scheduling System Using a Mathematical Programming Approach. In *Computer Scheduling of Public Transport*, Ed. Wren, A., North Holland Publishing Co., Amsterdam, 281-296.

54. Nemhauser, G.L., Trotter, L.E. and Nauss, R.M. (1974) Set Partitioning and Chain Decomposition. *Management Science* **20**, 1413-1423.

55. Nemhauser, G.L. and Wolsey, L.A. (1988) *Integer and Combinatorial Optimization*, John Wiley and Sons, New York.

56. Orloff, C.S. (1976) Route Constrained Fleet Scheduling. *Transportation Science* **10**, 149-167.

57. Oyama, T. and Han-I, S. (1987) Application of Discrete Optimization Techniques to Train Scheduling Problems. *Asia-Pacific Journal of Operational Research* **4**, 158-186.

58. Padberg, M.W. (1973) Perfect Zero-One Matrices. *Mathematical Programming* **2**, 199-215.

59. Paixão, J. and Branco, I.M. (1988) Bus Scheduling with a Fixed Number of Vehicles. *Lecture Notes in Economics and Mathematical Systems 308: Computer-Aided Transit Scheduling*, Ed. Daduna, J.R. and Wren, A., 28-40.

60. Papadimitriou, C.H. and Steiglitz, K. (1982) *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall Inc., New Jersey, U.S.A.

61. Parker, M.E. and Smith, B.M. (1981) Two Approaches to Computer Crew Scheduling. In *Computer Scheduling of Public Transport*, Ed. Wren A., North Holland Publishing Co., Amsterdam, 193-211.

62. Popken, D.A. (1988) Multiattribute Multicommodity Flows in Transportation Networks. Ph.D. Thesis, Air Force Institute of Technology, Wright-Patterson AFB OH 45433-6583, University of California, Berkeley.

63. Riberio, C.C. and Soumis, F. (1991) A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem. *Presented at the 14th International Symposium on Mathematical Programming*, Amsterdam.

64. Rousseau, J-M., Ed. (1985) *Computer Scheduling of Public Transport 2*, Elsevier Science Publishers B.V., North-Holland.

65. Rubin, J. (1973) A Technique for the Solution of Massive Set Covering Problems with Application to Airline Crew Scheduling. *Transportation Science* **7**, 34-48.

66. Ryan, D.M. (1992) The Solution of Massive Generalized Set Partitioning Problems in Aircrew Rostering. *Journal of the Operational Research Society* **43**, 459-467.

67. Ryan, D.M. and Falkner, J.C. (1988) On the Integer Properties of Scheduling Set Partitioning Models. *European Journal of Operational Research* **35**, 442-456.

68. Ryan, D.M. and Foster, B.A. (1981) An Integer Programming Approach to Scheduling. In *Computer Scheduling of Public Transport*, Ed. Wren A., North Holland Publishing Co., Amsterdam, 269-280.

69. Scott, D. (1985) A Large Scale Linear Programming Approach to the Public Transport Scheduling and Costing Problem. In *Computer Scheduling of Public Transport 2*, Ed. Rousseau, J-M., Elsevier Science Publishers B.V., North-Holland, 473-491.

70. Seshan, C.R. (1981) Some Generalisations of the Time Minimising Assignment Problem. *Journal of the Operational Research Society* **32**, 489-494.

71. Shepardson, F. (1985) Modelling the Bus Crew Scheduling Problem. In *Computer Scheduling of Public Transport 2*, Ed. Rousseau, J-M., Elsevier Science Publishers B.V., North-Holland, 247-261.

72. Simonnard, M. (1962) *Integer Programming*. Prentice-Hall Inc., New Jersey, U.S.A.

73. Smith, B.M. and Wren, A. (1981) VAMPIRES AND TASC: Two Successfully Applied Bus Scheduling Programs. In *Computer Scheduling of Public Transport*, Ed. Wren, A., North Holland Publishing Co., Amsterdam, 97-119.

74. Solomon, M.M. and Desrosiers, J. (1988) Time Window Constrained Routing and Scheduling Problems - A Survey Paper. *Transportation Science* **22**, 844-853.

75. Swersey, A.J. and Ballard, W. (1984) Scheduling School Buses. *Management Science* **30**, 844-853.

76. Ward, R.E., Durrant, P.A. and Hallman, A.B. (1981) A Problem Decomposition Approach to Scheduling the Drivers and Crews of Mass Transit Systems. In *Computer Scheduling of Public Transport*, Ed. Wren, A., North Holland Publishing Co., Amsterdam, 297-312.

77. Wolfe, P. (1963) A Technique for Resolving Degeneracy in Linear Programming. *SIAM J.* **11**, 205-211.

78. Wren, A. (1981) *Computer Scheduling of Public Transport*, North Holland Publishing Co., Amsterdam.

79. Wren, A. (1981) General Review of the Use of Computers in Scheduling Buses and their Crews. In *Computer Scheduling of Public Transport*, Ed. Wren, A., North Holland Publishing Co., Amsterdam, 3-16.

80. Wren, A. and Holliday, A. (1972) Computer Scheduling of Vehicles from one or more Depots to a Number of Deliver Points. *Operational Research Quarterly* **23**, 333-344.

81. Wren, A. and Smith B.M. (1988) Experiences with a Crew Scheduling System Based on Set Covering. *Lecture Notes in Economics and Mathematical Systems: Computer-Aided Transit Scheduling*, Ed. Daduna and Wren, 263-278.

82. Wren, A., Smith, B.M. and Miller, A.J. (1985) Complementary Approaches to Crew Scheduling. In *Computer Scheduling of Public Transport 2*, Ed. Rousseau, J-M., Elsevier Science Publishers B.V., North-Holland, 263-278.

83. Wright, M.B. (1989) Applying Stochastic Algorithms to a Locomotive Scheduling Problem. *Journal of the Operational Research Society* **40**, 187-192.

84. Wright, B.M. (1992) *Personal communication.*

APPENDIX A

Matrix Multiplication Method

Version 1

The matrix multiplication method finds the shortest paths between all pairs of nodes in a network.

Define

ϕ_{ij} = the length of a shortest path from i to j,

$\phi_{ij}^{(m)}$ = the length of a shortest path from i to j, subject to the condition that the path contains no more than m arcs.

Then define

$$\phi_{ij}^{(1)} = C_{ij},$$

where C_{ij} is the cost associated with arc i→j. Notice that if $C_{ii} \geq 0$ then it is known that any negative cycle through i passes through more than one node and it is therefore possible to set $C_{ii}=0$ and not overlook any negative cycles which may exist. So, defining,

$$\phi_{ij}^{(m+1)} = \min_k \{\phi_{ik}^{(m)} + C_{kj}\}.$$

If $\phi_{ii}^{(m)} < 0$ for some m, then there is a negative cycle through i. As no cycle can contain more than n arcs, if $\phi_{ii}^{(m)} \geq 0$ for all m, $1 \leq m \leq n$, then there are no negative cycles through i.

In these equations, each $\phi_{ij}^{(m)}$, of which there are approximately n^3 requires about n computations, giving a complexity of $O(n^4)$.

Version 2

This complexity can be improved by defining a matrix multiplication, denoted by ⊠ and defining the product of two matrices A⊠B by

$$A \boxtimes B = (p_{ij}),$$

where

$$p_{ij} = \min_k \{a_{ik} + b_{kj}\}.$$

Defining $\Phi^{(m)}$ to be the matrix $(\phi_{ij}^{(m)})$, then defining

$$\Phi^{(1)} = C,$$

where $C = (C_{ij})$. As before a check is made that $C_{ii} \geq 0$ for all i and then C_{ii} is set equal to zero for all i.

$$\Phi^{(m+1)} = \Phi^{(m)} \boxtimes C.$$

Repeated squaring gives

$$\Phi^{(2m)} = \Phi^{(m)} \boxtimes \Phi^{(m)}.$$

If for some m $\Phi^{(m)}$ has a negative i^{th} diagonal entry then there is a negative cycle through i.

There are approximately n^3 $\Phi^{(m)}$ matrices, and the maximum number of operations ⊠ to prove that there are no negative cycles is $O(\log_2 n)$. Therefore the

complexity of this method is $O(n^3 \log_2 n)$.

Version 2 is used in this thesis in conjunction with the Bellman-Ford algorithm.

APPENDIX B

Bellman-Ford Algorithm

Define $\phi_j^{(m)}$ to be the length of a shortest path the start node i to node j, subject to the condition that the path contains at most m arcs. Define

$$\phi_j^{(1)} = C_{ij}$$

where C_{ij} is the cost of traversing the arc $i \rightarrow j$ in the network, and

$$\phi_j^{(m+1)} = \min \{\phi_j^{(m)}, \min_{k \neq j}\{\phi_k^{(m)} + C_{kj}\}\}.$$

Having checked that $C_{ii} \geq 0$ for all i, then any negative cycle will pass through more than one node and it is possible to set $C_{ii}=0$ without overlooking any negative cycle which may exist. Then, $\phi_i^{(1)}=0$.

If $\phi_i^{(m)} < 0$ for any m then the method has located a negative cycle and the algorithm terminates. Since no cycle from i can contain more than n arcs, if $\phi_i^{(m)} \geq 0$ for all m, $1 \leq m \leq n$ then there are no negative cycles through i. Each evaluation of $\phi_j^{(m)}$ requires n-1 additions and n-1 comparisons. There are approximately n^2 $\phi_j^{(m)}$'s, and therefore the complexity is $O(n^3)$.

APPENDIX C

Hungarian Method

For the complete bipartite graph B=(V,U,E) with $|V| = |U| = n$ under the nxn cost matrix c_{ij}, the Hungarian Method is described by the following steps.

1. *Reduce:* For each $i \in V$ let $\alpha_i = \min_j c_{ij}$ and update $c_{ij} = c_{ij} - \alpha_i$. For each $j \in U$ let $\beta_j = \min_i c_{ij}$ and update $c_{ij} = c_{ij} - \beta_j$.

2. *Initialize:* For each $i \in V$ set mark(i)=-1 and for each $j \in U$ set mark(j)=-1. Set the value of nummkd=0. Then for each $i \in V$, if there exists $j \in U$ with mark(j)=-1 and $c_{ij}=0$ set mark(j)=i, mark(i)=j and nummkd=nummkd+1. Continue until each $i \in V$ has been considered.

3. *Check:* If nummkd=n a complete matching has been found, go to 8. Otherwise, for all $i \in V$ set label(i)=-1, and for all $j \in U$ set label(j)=-1.

4. *Unmarked:* Find an $i \in V$ with mark(i)=-1 and set label(i)=0.

5. *Label:* Find $i \in V$ with label(i)$\neq$-1. If there is $j \in U$ with mark(j)$\neq$i and $c_{ij}=0$ and label(j)=-1 set label(j)=i. Continue until all $i \in V$ with label(i)$\neq$-1 have been considered. If there exists $j \in U$ with label(j)$\neq$-1 and mark(j)=-1 then an augmenting path has been found; set OGPATH=j, go to 6. Otherwise, for each $j \in U$ with label(j)$\neq$-1, if mark(j)=i, for some $i \in V$, and label(i)=-1, set label(i)=j. Continue until all $j \in U$ with label(j)$\neq$-1 have been considered. Go to 7.

6. *Augmenting path:* For j^*=OGPATH, find $i^* \in V$ such that i^*=label(j^*). Then, set mark(i^*)=j^*, mark(j^*)=i^*. Update j^* such that j^*=label(i^*). If j^*=0 then set nummkd=nummkd+1, go to 3. Otherwise, update i^* with i^*=label(j^*). Continue marking nodes until a row labelled 0 is found.

7. *Update:* Update the matrix c_{ij} to find another 0. For all $i \in V$ with label(i)$\neq$-1, and $j \in U$ with label(j)=-1, find $\delta=\min_{ij} c_{ij}$. For all $i \in V$ with label(i)$\neq$-1 update $\alpha_i=\alpha_i+\delta$ and $c_{ij}=c_{ij}-\delta$. For all $j \in U$ with label(j)$\neq$-1 update $\beta_j=\beta_j-\delta$ and $c_{ij}=c_{ij}+\delta$. Go to 4.

8. *Solution:* The assignment is given by: neighbour of $i \in V$ is mark(i). The cost of the assignment is $\Sigma_i \Sigma_j (\alpha_i+\beta_j)$.

APPENDIX D

The Lagrangean relaxation with respect the resource constraint PAP_k3 in the formulation PAP_k given in chapter eight is:

$LPAP_k$

$$\text{MIN } \Sigma_{i\in P(k)}\Sigma_{j\in P(k)} (c_{ij}+\mu m_i+\mu M_{ij})v_{ij}^{\,k} - \mu U_k$$

s.t.

$$\Sigma_{j\in P(k)} v_{ij}^{\,k}=1 \qquad \forall\, i\in P(k) \quad \textbf{LPAP}_k\textbf{1}$$

$$\Sigma_{j\in P(k)} v_{ji}^{\,k}=1 \qquad \forall\, i\in P(k) \quad \textbf{LPAP}_k\textbf{2}$$

$$v_{ij}^{\,k} \geq 0 \qquad \forall\, i,j\in P(k) \quad \textbf{LPAP}_k\textbf{3}$$

The set P(k), the variables and data are as defined for the formulation PAP_k, and μ is the Lagrangean multiplier associated with constraint set PAP_k3. Constraint PAP_k4 of PAP_k is now redundant as the solutions to $LPAP_k$ only take zero-one values.

The problem PAP_k is referred to by Nemhauser and Wolsey [55] as a flow problem with budget constraint. The problem $LPAP_k$ is solvable in polynomial time and is considerably easier to solve than the original problem PAP_k. To see the relationship between the problem PAP_k, the linear programming relaxation of PAP_k and the problem $LPAP_k$, consider figure Di shown below.

In the diagram the points denoted by I_i are integer points and the points R_r are non-integer points. The shaded polytope represents the feasible region of the problem PAP_k. The polytope described by $R_1R_3I_7I_4$ represents the feasible region of the linear programming relaxation of PAP_k. The polytope described by $I_1I_{10}I_4$ represents the feasible region of the problem $LPAP_k$ and the bold line L1 is the constraint PAP_k3 which dissects this feasible region. Therefore, the polytope $I_1R_2I_7I_4$ is the intersection

of the convex set of the points which satisfy constraint set PAP_k3 with the polytope $I_1I_{10}I_4$.

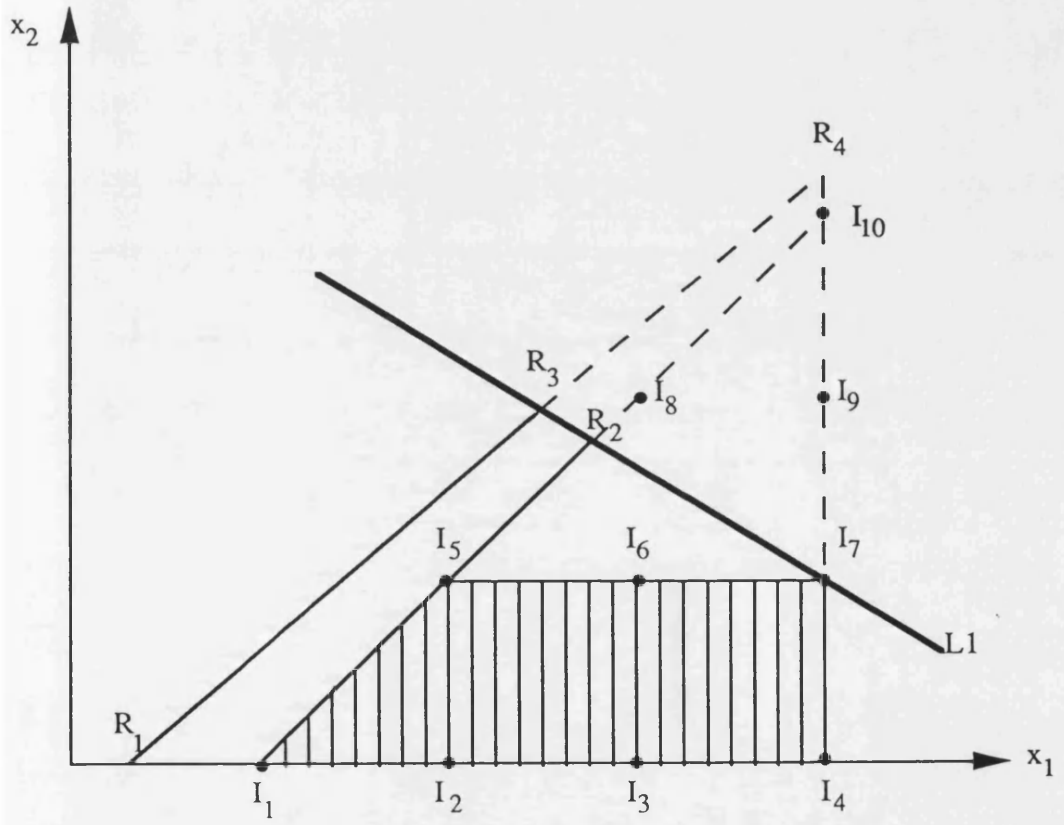


Figure Di

It is clear from the diagram that by an appropriate choice of the value of the Lagrangean multiplier μ an optimal solution of $LPAP_k$ which satisfies the constraint PAP_k3 locates at least one of the extreme points I_1, I_4 and I_7 of the polytope $I_1R_2I_7I_4$. The integer solutions found using this method depend on the values of c_{ij}, m_i and M_{ij}

in $LPAP_k$. In addition, by choosing other values of μ which correspond to the constraint PAP_k3 being ineffective or violated, it may be possible to locate other integer points of the polytope described by $I_1I_{10}I_4$. This can be viewed as being equivalent to increasing or decreasing the value of U_k in constraint PAP_k3, and thereby causing a shift in the line L1. Therefore, it is clear that by varying the value of the Lagrangean multiplier μ it is possible to find the optimal integer solutions to the problem PAP_k which correspond to different values of U_k.

The alternative method for locating optimal integer solutions to PAP_k for different values of U_k, is to vary the value of U_k in constraint PAP_k. As before, altering the value of U_k is equivalent to a shift in the line L1. For the position of the line L1 indicated in the diagram, an optimal solution to the linear programming relaxation of the problem PAP_k is at least one of the extreme points R_1, R_3, I_7 and I_4. By shifting the line L1, other points become extreme points of the linear programming relaxation of PAP_k. Therefore, the parametric investigation of the right-hand side of constraint PAP_k3, along with a branch and bound procedure, can be used to locate optimal integer solutions to PAP_k for different values of U_k.

It is straightforward to implement the parametric investigation of the right-hand side of constraint set PAP_k3 using the facility available in LAMPS. This method is especially attractive if all or almost all of the solutions found are integer and very little branch and bound work is required. Initially, the use of assignment algorithm to solve the problem $LPAP_k$ was rejected in favour of this method, on the grounds that it is more cumbersome to implement and the values of μ are chosen arbitrarily. However, it was belatedly realised that by using an alternative formulation for $LPAP_k$, it is possible use a facility in LAMPS which allows the user to perform a parametric investigation of the objective function coefficient of a specified variable. The alternative formulation is given by:

$PPAP_k$

$$\text{MIN } \Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\ \xi_1 + \mu\xi_2 - \mu U_k$$

s.t.

$\Sigma_{j\in P(k)}\ v_{ij}^k=1$	$\forall\ i\in P(k)$ **$PPAP_k1$**
$\Sigma_{j\in P(k)}\ v_{ji}^k=1$	$\forall\ i\in P(k)$ **$PPAP_k2$**
$\Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\ c_{ij}v_{ij}^k=\xi_1$	**$PPAP_k3$**
$\Sigma_{i\in P(k)}\Sigma_{j\in P(k)}\ \mu(m_i+M_{ij})v_{ij}^k=\xi_2$	**$PPAP_k4$**
$v_{ij}^k \geq 0$	$\forall\ i,j\in P(k)$ **$PPAP_k5$**
$\xi_1,\ \xi_2 \geq 0$	**$PPAP_k6$**

This problem is essentially the same as $LPAP_k$ as constraints $PPAP_k3$ and $PPAP_k4$ are simply used to define the new variables ξ_1 and ξ_2. So, the variables v_{ij}^k take only zero-one values and, as the data c_{ij}, m_i and M_{ij} take integer values, the variables ξ_1 and ξ_2 only take integer values. Using this formulation it is possible to continuously vary the value of μ, the objective function coefficient of ξ_2, and find alternative integer solutions to the problem PAP_k for different values of U_k. When performing the parametric investigation of μ the constant term μU_k can be dropped. The advantage of this method over the use of a parametric investigation of the value U_k in PAP_k is that the solutions found using this method are always integer.