Column Generation Approaches to Bus Driver Scheduling

by

Sarah Fores

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy

The University of Leeds
School of Computer Studies
March 1996

The candidate confirms that the work submitted is her own and that appropriate credit has been given where reference has been made to the work of others.

Abstract

The bus driver scheduling problem involves assigning bus work to drivers in such a way that all the bus work is covered and the number of drivers and duty costs is minimised. This is complicated by the fact that there are restrictions on the formation of valid duties.

A review of computerised scheduling systems is presented, along with a more detailed description of one such system which uses a set covering model to produce a schedule from a set of previously generated valid duties. This method first solves the Linear Programming relaxation, and then uses Branch and Bound techniques to search for a good integer solution. Improvements to this system are detailed.

Most systems which use mathematical programming methods to solve the driver scheduling problem need heuristics to reduce the size of the problem since there are potentially many thousands of valid duties, even for small problems. Column generation is a technique which implicitly considers a much larger number of duties, whilst retaining a much smaller working duty subset. A specialised column generation method is implemented within the existing set covering system, and the results of tests on seven problems presented. Each problem instance is solved with two sizes of duty set, and timings compared to those tested on the set covering system. Results show an average reduction in execution time of 41% using column generation, and the larger data sets yield better schedules in terms of the number of duties and the overall cost.

Contents

List of Tables

List of Figures

Acknowledgements

I acknowledge Derek Morley - the original inspiration!

I would like to thank my supervisors, Professor A. Wren and Dr. L. G. Proll, for their guidance and ... well ... supervision. Friends at Leeds University have been very supportive, and in particular I would like to mention Lisa Simpson and Jackie Carter. Good luck with your Ph.D.'s!

Finally, I would like to thank Jeremy, and other close friends and family, for their continued patience. I would particularly like to thank my mother who tragically lost her partner in December 1995.

Foreword

The following summary of contents outlines the purpose of each chapter.

Chapter 1 presents the driver scheduling problem, and briefly covers the need for computer systems to be used in the solution process.

Chapter 2 summarises the computerised scheduling systems and techniques available. No scheduling system presented can solve the driver scheduling problem optimally, but methods incorporating column generation give encouraging results and a summary of these methods applied to scheduling problems is presented.

Chapter 3 introduces some of the problems inherent in solving driver scheduling problems. A technique is proposed in which the number of mealbreak chains in any given time period is optimised, and an exhaustive search solution method is considered.

Chapter 4 outlines the solution method and steps involved in using the computerised system TRACS II which was developed at the University of Leeds.

Chapter 5 details the mathematical programming component of TRACS II, which is used to produce a schedule with the fewest number of drivers at least cost from a set of previously generated valid duties. An alternative solution strategy is also detailed which has been shown to reduce computation time.

Chapter 6 introduces the theory of column generation and describes a system which incorporates column generation to solve the driver scheduling problem. An algorithm is proposed to incorporate column generation within the mathematical programming component of TRACS II.

Chapter 7 describes various column generation strategies tested and the results of a successful implemetation on seven problems.

Chapter 8 proposes and tests various potential improvements to the column generation method. Successful techniques are implemented to derive a refined column generation method, and results on the seven problems are presented.

Chapter 9 presents a summary of the successful column generation method, and suggestions for further refinement of the method.

Chapter 1

The Driver Scheduling Problem

1.1 Introduction

The driver scheduling problem presented is that of a bus company but could equally well be considered for other operations requiring the assignment of crews to a predetermined vehicle schedule. Bus companies are required to provide a timetabled bus service for a particular area of operation and also, possibly, contract vehicles to clients, e.g. for the transportation of school children or less regular long-distance holiday services. Normally the timetable would remain constant for a period of months, requiring little or no alteration to the duty contents or the daily operations during that time, although increased competition between operators raises the possibility that schedules are changed more frequently than in the past.

As well as drivers many other staff are employed in bus companies to deal with administration and vehicle maintenance and hence labour costs are high, making it essential to maintain efficient driver allocations. A bus crew can be thought of as a one or two person operation depending on the requirement of a particular bus company to employ a conductor. It is increasingly rare for a company to allocate conductors to vehicles, and certainly in all experimental data provided in this thesis it was only required to assign drivers, but the methodology used can be extended to situations requiring more than one person. Since deregulation in the United Kingdom, many small companies were formed which each covered a relatively small geographical area. In many cases this means that the operation is run from only one depot. It is usual for larger bus companies to divide the bus work by garage in order to use a separate group of drivers for each area with a separate scheduling process. Bus companies may also group

certain routes together and schedule them for a subset of drivers, with enough variety to allow rostering. Even though this system may be more inefficient with regard to the whole of the bus work, it allows drivers to become accustomed to certain routes and provides more familiarity for regular passengers.

A minor alteration of a start or end time for a bus may improve the efficiency of a schedule by avoiding the need for an extra driver or removing unnecessary waiting at the garage. For this and other reasons it may seem preferable to schedule the vehicles and drivers simultaneously to attain global optimality. However, for the majority of cases it is impracticable and the vehicle and driver schedules are compiled separately. Ball [1, 2] has proposed methods for scheduling the processes simultaneously, but in practice bus operators generally wish to inspect and amend the bus schedule in order to satisfy local conditions before allocating the driver work. Driver scheduling must also take into account the fact that drivers have more limitations on work time than the vehicles and so the rules used to compile the two processes are very different. It is, however, useful to have the ability to analyse quickly the impact that changes in a vehicle schedule may have on the number of drivers in a driver schedule.

Please note that Hartley [3] provides a complete description of all terms used in bus and driver scheduling, including variations used in different countries.

1.2 Bus Scheduling

The bus scheduling process takes a set of predetermined journeys to be operated and allocates a vehicle to each trip. The routes and frequency of services are best obtained from knowledge of passenger demand from previous schedules. The objective is to minimise the total number of vehicles and running costs, whilst still serving a particular area. Running costs may be minimised by avoiding the need for unnecessary *dead running* (unproductive time not used to carry passengers) since this incurs driver and fuel costs but offers no gain from fares. Dead running includes idle time at termini which must not be more than is compensatory for late running.

There are predetermined location points throughout the journey, each with a particular arrival and departure time. These are places at which the passengers can board or depart from the vehicle, and/or places which allow a convenient changeover of drivers. The designated locations at which a driver can be relieved by another are known as *relief points* and there may be more

than one of these throughout any particular bus journey. Each relief point has a corresponding time and these point-time pairs are known as *relief opportunities*. The work of each individual bus in a day is known as a *running board*.

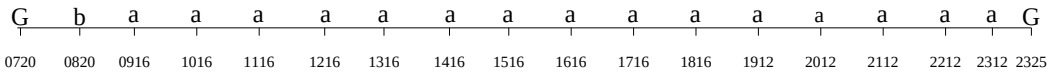

Figure 1.1: Representation of the Work of a Single Bus in a Day

Figure 1.1 shows an example of a running board in which relief points can be identified from the following list :

G = Depot
a = point A (Abel)
b = point B (Baker)

The bus leaves the garage at 0720 and passes relief point **b** at 0820. Relief point **a** is passed at various times until 2312 and then the bus is returned to the depot at 2325.

Bus schedules are usually represented by a series of running boards whose lengths are determined by the total time the bus is in operation away from the garage. Figure 1.2 depicts a bus schedule but for simplicity the relief points are not indicated, only the relief times. The bus does not necessarily continue on the same service for its time away from the depot, and may have dead running to link passenger carrying trips or allow a bus to get to and from its first and last allocated trips. Marked along these lines are possible relief opportunities which are normally logically numbered and each will be situated close to a depot or canteen facilities. The scheduling process becomes complicated if the arrival and departure times for this location differ, and for the purposes of the problem only one time is used so that the vehicle is not left unattended at any point. The interval between any two relief points is known as a *piece of work* and must be covered by a single driver, since by definition this interval is indivisible.

Since buses will be more frequent at times of greatest demand, different schedules have to be formed depending on the day of the week. For urban weekday operations, there will be peaks in demand corresponding to passengers travelling to and from work. It is usual to expect weekend schedules to differ from weekly ones to cater for the change in public demand with often a much

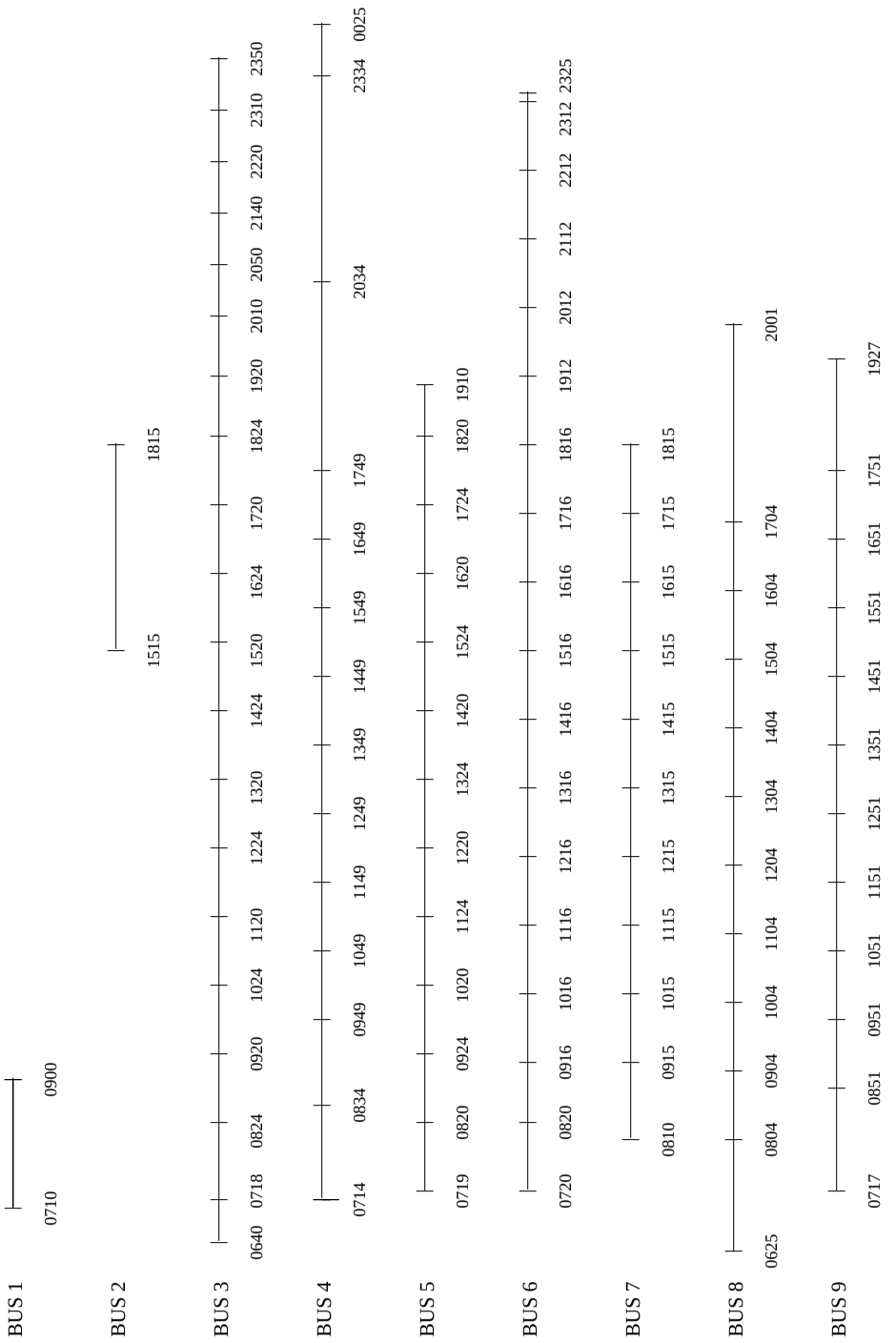

Figure 1.2: A Diagramatic Representation of a Bus Schedule

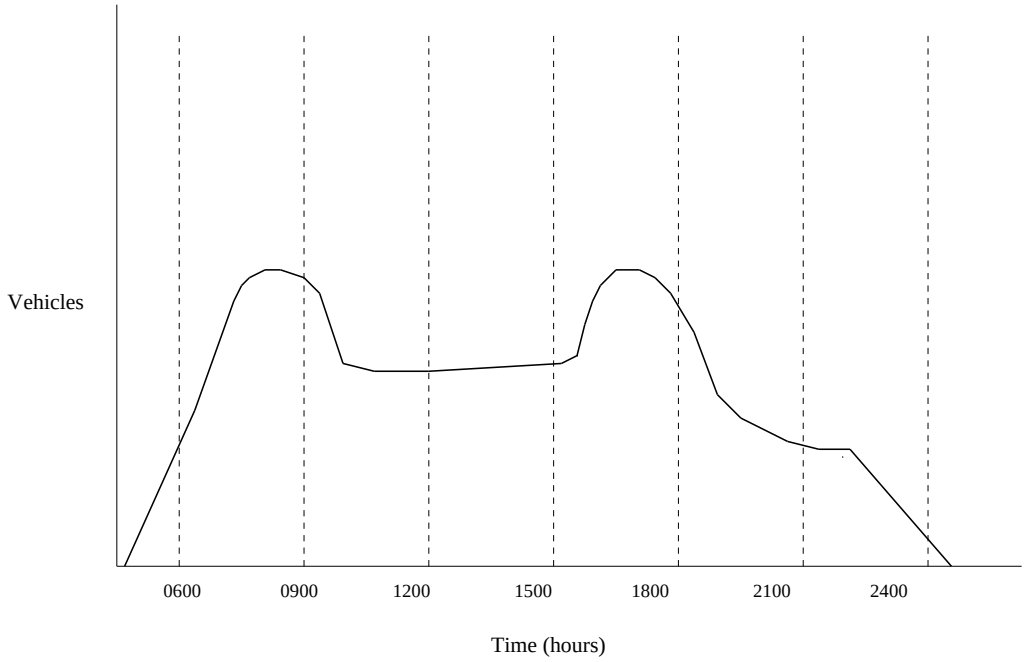

Figure 1.3: Typical Daily Fluctuations on the Number of Buses on a Weekday

reduced operation on a Sunday. Account must also be taken of any contract vehicles such as school buses. Figure 1.3 shows how the demand on an urban weekday may affect the number of buses in operation throughout the day.

1.3 Driver Scheduling

It is a necessity that any bus in operation must have a driver, and possibly a conductor, assigned to it at all times. Labour agreement rules are present to ensure that drivers do not work unacceptable duties and provide certain guidelines restricting the construction of duties. All changeovers of drivers have to take place at relief opportunities (which are defined as any time identified as a relief point on the vehicle schedule). Driver scheduling then involves dividing the bus work into a set of *duties* such that every piece of work in operation is allocated to a driver, unless a period where the bus can be left unattended is allowed, and each duty conforms to the governing rules.

It is usual in the U.K and most European bus companies for drivers to work on two or more buses with a mealbreak taken away from the vehicle. There are some companies however where no mealbreak requirements are necessary, which simplifies the problem as the running boards

can be divided and allocated to a driver. The data presented will be relevant to the first instance where mealbreaks will be necessary. Since it is always possible to cover the bus work, the problem for the company becomes one of minimising the total number of duties and the total duty costs. In fact, the minimisation of the total number of duties is regarded as more important since there are many costs which depend directly on the number of drivers regardless of their wages. Duty costs depend on the combination of work that they contain, incorporating the hourly wage and including penalty costs for undesirable features such as long or unsociable hours. The overall duty preference should also be considered, which may include a requirement that a limit is put on the number of duties of a certain type.

1.3.1 Labour Agreement Rules

The rules governing the construction of duties are mostly determined by past practice and local conditions, and are agreed between the bus company and the union as internal regulations, whilst other regulations are provided by the Government. Some of the rules given are meant as guidelines which can be relaxed (SOFT rules), whereas others must be adhered to (HARD rules). Most rules are relevant to all bus companies even though their parameters differ, but it is likely that each bus company has developed its own additional rules which need to be considered for its drivers requirements.

Typically, global rules will relate to :

- The paid allowance for signing on and off at the garage.
- The maximum time which a driver can work without a mealbreak. This is usually four to five hours and may consist of work on more than one bus with a *joinup* allowed to transfer onto another vehicle.
- The minimum length of a mealbreak. This will usually be a fixed time, say 30 minutes, plus travel time to the canteen.
- The total working time.
- The total spreadover (duration between the beginning and end of a duty).

The tightness of the rules plays an important part in the scheduling process. For instance, schedulers find it more difficult to compile efficient duties where the maximum time that may be worked without a break is short, although the existence of a tight constraint on this time

may help by reducing the number of potential valid duties.

Local rules tend to be built around traditional duty formations, such as drivers retaining the same bus after a mealbreak and large penalties to deter duties with particular features from being included in the final schedule.

1.3.2 Duty Types

There are two main types of duties - STRAIGHT and SPLIT. The straight duties are normally categorised depending on the time of day that they are worked, although this distinction is used more for producing a fair roster than for providing separate rules used in the formation of individual duties. The timing of the peaks in bus service, as illustrated in Figure 1.3, makes it impossible for drivers of straight duties to work both peaks during the duty, illustrating the need to introduce split duties. Figure 1.4 divides a typical weekday operation into the types of duty required to cover the bus schedule, indicating the times of day that each duty type would normally be worked. This however is only an indication since bus companies define the duty types they wish to use and the times at which they will be restricted.

A *spell* of work is defined as a series of consecutive pieces of work on the same bus and a *stretch* is defined as a series of spells without an intervening mealbreak. A *duty* is defined as two stretches of work separated by a mealbreak. *Joinups* may be used to allow a driver to link two spells of work. Although it is legal to have duties which work several buses, it is usual and preferable to restrict the driver to two or three spells of work as time spent changing vehicles within a stretch will count as worked time not contributing to productivity. Enlarging this number may improve overall efficiency, since a duty which is not restricted to the number of buses worked may combine well with good two-part duties giving a better schedule overall, and the construction of more duties allows a greater choice of combination. However it increases the potential number of valid duties considerably and would complicate the schedule from the viewpoint of both the drivers and the bus company.

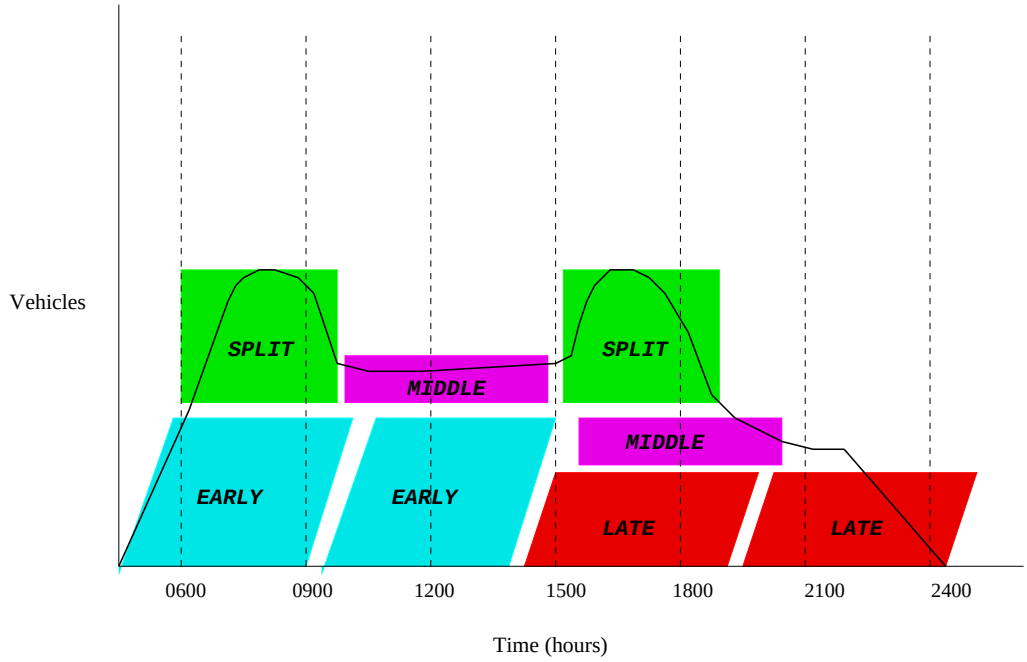

Figure 1.4: The Types of Duty Used to Cover Daily Bus Work

STRAIGHT Duties

These duties can be classified as EARLY, LATE, MIDDLE, and DAY depending on the time of day that they are covering. The drivers operating early duties would be required to take the first bus trip out of the garage, and those with late duties return the buses to the garage after the last trip.

The straight duties are usually formed as two or three spells separated by a mealbreak of around 30 minutes to an hour such that the total spreadover of the duty is around nine hours.

SPLIT duties

The main purpose of the split duty is to provide drivers to cover the peak periods when more buses are in operation. They also assist in maintaining the service whilst drivers from other duties take their mealbreaks. The total spreadover of these duties would be around twelve hours with a longer break between stretches, making them more unpopular with the drivers, and so higher penalty costs are often attached to these duties to restrict the number created. The maximum driving time would generally be the same as for straight duties.

OVERTIME Duties

In some cases bus trips can be covered by overtime work, which consist of one-part duties of around two to four hours. These are in addition to normal duties for relatively high payment and may be useful in covering peaks in demand. Most companies however, prefer not to incorporate overtime duties unless they become necessary due to work remaining uncovered once a schedule has been formed.

1.3.3 Schedule Representation

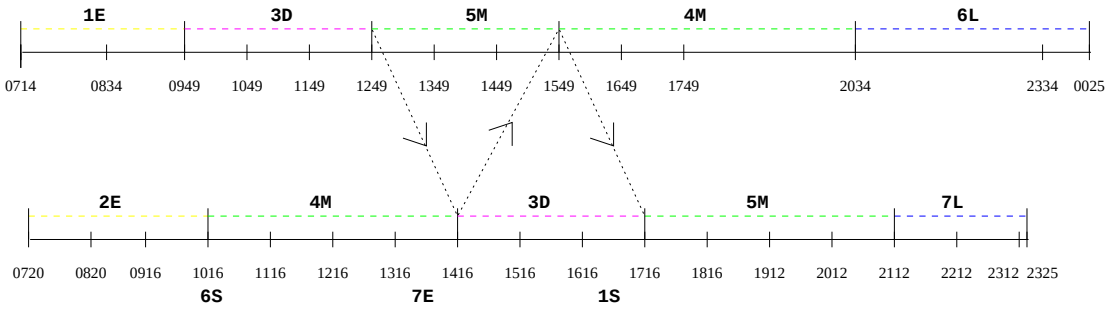

Figure 1.5: An Example of Possible Duties Used to Cover Part of a Schedule

Figure 1.5 represents a possible driver schedule that could be built around the bus work shown in Figure 1.1. The dotted line indicates a mealbreak. Duties are identified by a number and a letter representing the duty type. In order to explain the diagram the work of one duty will be described. Duty 3D begins, after signing on at the garage, on the first bus at 0949. The first stretch lasts for three hours, taking a mealbreak at 1249. Including the travelling time to the canteen and back, the break has a total duration of 1 hour and 27 minutes, returning to relieve the driver of the second bus at 1416. The second stretch also lasts for three hours where the driver then returns to the garage to sign off. Complete duties are only shown for duties 3, 4, and 5 which are respectively Day, Middle and Middle duties. Some companies prefer the driver to retain the same bus after a mealbreak, although this rearrangement is performed independently from driver scheduling.

Figure 1.6 shows how a typical driver schedule might look for the bus schedule given earlier. All duties in this example are only two-part but display all possible duty types.

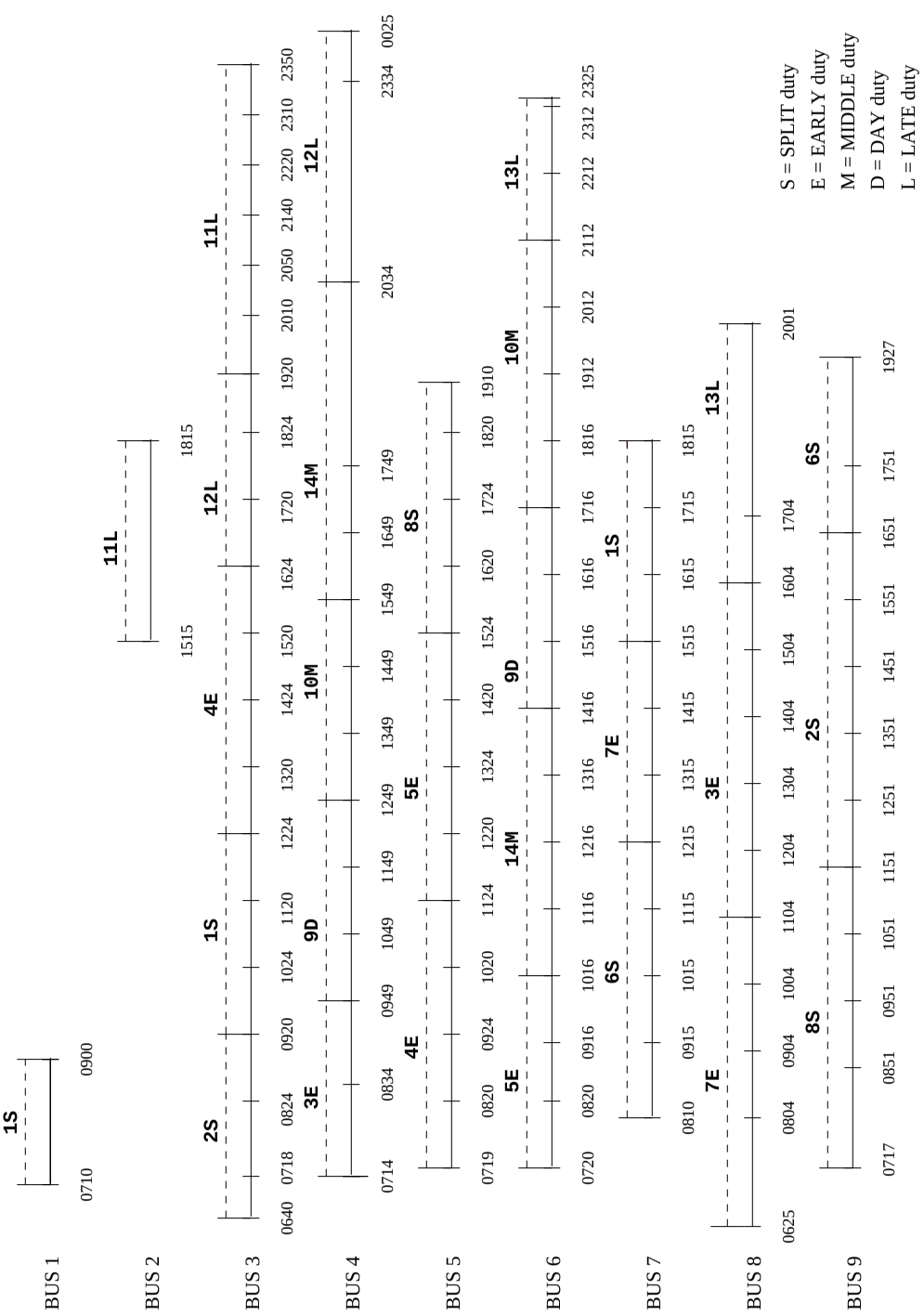

Figure 1.6: A Typical Diagram Depicting a Driver Schedule

1.3.4 Manual Scheduling

Scheduling, both of vehicles and drivers, has been acknowledged as a difficult task for many years [4]. The principles involved are still valid even with the intervention of computer methods.

Manual schedulers have to generate duties which fit together to cover the bus work, with each duty needing to be checked to ensure that it conforms to the rules, and the overall schedule must not violate any constraints restricting particular duty types. Scheduling mealbreaks causes complications as there are many choices for a second stretch of a duty after a mealbreak once a first stretch has been determined. Even if the bus schedule has been divided so that smaller groups of buses are considered at any one time, it is still a complicated manual process.

Experienced manual schedulers may be able to produce a schedule containing the minimum number of duties, but the vast number of valid duty combinations may prevent the most efficient combination from being created. Once a schedule has been produced for a company, a manual scheduler will probably only have to update it when alterations are made to the bus schedule, rather than the process having to be started from scratch each time. The skills and knowledge that a manual scheduler possesses cannot easily be transcribed into rules which can be incorporated into a computerised system.

1.3.5 Computer Scheduling

The earliest successful computer systems for driver scheduling were developed in the 1970s and a series of workshops [5, 6, 7, 8, 9, 10] have been organised since then to discuss the latest techniques and software evolving with rapid developments in technology. Wren and Rousseau [11] give an outline of the driver scheduling problem and computer approaches to solving it, with emphasis on the more recent developments.

The limitations of computer technology meant that earlier systems found heuristic solutions rather than using known mathematical programming models [12, 13]. Most of these systems used heuristics to refine some initial schedule and are not easily portable to other bus companies. More recently mathematical programming techniques have been applied to solve the driver scheduling problem with a given set of valid duties. Due to the large number of valid duties that can be formed for each individual problem, mathematical programming methods cannot be used in isolation and hence most research has been directed at producing systems with a

combination of mathematical programming and heuristics to reduce the problem size. As the user requirements are hard to define and the knowledge of the manual scheduler is invaluable, most computer scheduling systems aim to aid the scheduler rather than provide a 'black box' problem solver. Since computer scheduling was first introduced much more powerful computers have become available, allowing results to be calculated even more quickly, more duties to be considered in forming a driver schedule, and more options to be available to the user. With present competition between operators, many of them require a system which can perform sensitivity analysis on a schedule to predict effects that revisions in the timetable would make on the overall cost. Another popular recent feature of scheduling packages is graphics, not only to make the systems more attractive to clients, but also to allow the user to manipulate schedules by diagram. However, in many cases with particularly large problems, it is still not possible to find the optimal solution, that is, the particular combination of duties which have the lowest possible cost to the operator, taking all legal requirements into consideration. Computer-aided scheduling is intended to calculate more efficient schedules more quickly, with the drawback being the cost in setting up such a system.

Several bus driver scheduling systems are in widespread commercial use and are described in section **2.2**. It should be noted that the quality of existing methods is acceptable to users even though an optimal solution is not guaranteed. The implementation of computer scheduling packages has created large cost savings to bus companies but although any further improvements would allow more savings, small benefits may be outweighed by further purchasing costs or a greater time incurred in the scheduling process.

This thesis is concerned with investigating a mathematical programming technique which will consider more duties from which to form a driver schedule. This is implemented into an existing mathematical programming system and will be tested to ascertain any improvements in driver schedules in terms of number of drivers and overall cost, and also in terms of execution time, which remains an important issue to schedulers who may wish to analyse the effects of timetable alterations quickly.

Chapter 2

Review of Driver Scheduling Methods and Applications of Column Generation to Transportation

2.1 Introduction

Since the early 1960s when computers were first introduced to solve bus and driver scheduling problems, many different techniques have been developed. Whilst some work has concentrated on building complete computer scheduling packages, research has also taken place to test new techniques and algorithms which may aid or improve parts of the solution process. The purpose of this chapter is both to review some of the different approaches which have been used in attempting to produce driver schedules, and also to describe some of the applications of a column generation technique which is used to solve certain types of mathematical programming problems including driver scheduling problems.

2.2 Computerised Scheduling Systems

Since there are many papers detailing the theory and application of computer based driver scheduling systems in the proceedings of the six workshops on computer-aided scheduling of public transport [5, 6, 7, 8, 9, 10], this chapter will concentrate on those published more recently and any others which relate to current work. Willers [14] presents a fuller review of the material in the proceedings.

Complete scheduling packages which are in most common use are : IMPACS [15, 16, 17, 18], HASTUS [19, 20, 21, 22, 23, 24], HOT II [25, 26, 27], UMA(Trapeze) and Teleride Sage. Other systems are in use such as TransTec EDP in Germany, and a system called BERTA [28] is being developed in Berlin to provide automatic bus and driver schedules for an underground, bus and tram system, but details of these systems have not been published. Also, details have not been published for the systems UMA and Teleride Sage and so they will be omitted from this review. Details regarding the driver scheduling approach used by IMPACS are similar to those in TRACS II because they both originated as IMPACS developed at the University of Leeds. TRACS II is the system retained for research purposes and is described in detail in Chapter 4.

2.2.1 A Crews First Approach

The approach, described by Patrikalakis and Xerocostas [29], first develops driver schedules and then forms the vehicle schedules around them. The opposite approach from standard systems was motivated by the fact that driver costs dominate vehicle costs and so if these are optimised first an efficient driver schedule will create large cost savings and will also aid the formation of an efficient vehicle schedule.

The system constructs a set of driver duties which cover all the timetabled trips by decomposing the problem into duty types. Early duties and first halves of split duties are chosen first and a set covering formulation selects some of these. Workpieces covered by these duties are removed from the problem and the duties covering the remainder of the trips are formed by a similar method. The optimisation of the model can only take into account approximate costs of duties since these are dependent upon vehicle locations which have not yet been ascertained.

The vehicle scheduling problem is then formulated as a network flow problem on a graph formed by the duties chosen initially. Solving the minimum cost network flow problem, with side constraints to ensure the compatibility of driver and vehicle schedule, produces a final vehicle schedule. The driver scheduling problem then has to be re-solved using a network flow model to derive a driver schedule which is consistent with the vehicle schedule.

Satisfactory results are reported for two small problems but much possible further work is detailed with regard to extending the model to deal with larger and more complicated problems.

2.2.2 The BDS System

Carraresi et al. [30] describe the BDS (Bus Driver's Scheduling) system which uses combinatorial optimisation and logic programming.

The paper details an implementation of the system at ATAF in Florence incorporating a total of 684 vehicle trips throughout the day. The vehicle schedule is first simplified by selecting some duties and workpieces which will temporarily be removed from the problem. The resulting vehicle schedule is then partitioned by finding feasible paths through a graph representing the vehicle schedule. The workpieces formed are then matched to form valid duties by using a Lagrangean relaxation of the driver scheduling algorithm, and the driver scheduling problem is completed by forming duties for the previously excluded workpieces. At this stage, since some labour agreement rules were not specified initially, the duties formed are modified so as to satisfy all rules. The paper describes an expert system approach which more efficiently schedules drivers for the excluded workpieces but results are not given.

2.2.3 A System for Rural Vehicle and Driver Scheduling

General purpose methods of scheduling drivers are often difficult to achieve due to differences in labour agreement rules between organisations, although the methodology often remains the same. Rural vehicle operations however require a different technique due to particular requirements. It is possibly inefficient to schedule vehicles before drivers in the rural situation due to problems of transferring drivers between locations and needing to take into consideration subsistence payments to drivers for time spent away from their residency.

Tosini and Vercellis [31] describe an interactive method of scheduling vehicles and drivers based upon rules relating to many rural operations in Italy. The drivers are scheduled first as a minimum cost network flow model and the schedule can be amended or improved interactively. The corresponding workpieces are then combined using a minimum cost formulation to create a vehicle schedule.

2.2.4 The COMPACS System

COMPACS (COMPuter Assisted Crew Scheduling) is an interactive driver scheduling system which is described by Wren and Chamberlain [32].

The system first estimates the number of drivers needed to cover the bus work and then aids the construction of the driver schedule by verifying and costing duties entered by the scheduler and by suggesting duties which cover particular pieces of work. At any stage the estimator can be utilised to check how many duties are required to cover the remaining bus work so that any alterations can be made to the chosen duties if necessary.

This interactive system is best used to amend manually or add to a schedule already produced as COMPACS itself does not contain any optimisation techniques.

2.2.5 The EXPRESS system

The EXPRESS system [33] is based on a set partitioning model of the driver scheduling problem, firstly using heuristics to reduce the problem size and then a three-stage approach to develop the driver duties. The set partitioning model is solved using the ZIP package [34] which is also used within TRACS II.

The approach was developed to apply to the New Zealand bus driver scheduling rules which require scheduled mealbreaks, all work to be covered, and the possibility of four or more part duties. The bus work is split into five route groups and each group allocated drivers separately. The driver scheduling process aids the completion of the vehicle schedule by determining the end location of vehicles in order to suit the duties.

The three-stage approach used to allocates drivers to vehicles within each route group is as follows :

- **Stage 1** The early part of the bus schedule is excluded whilst forming middle and late duties. The formation is done using a mathematical programming model to select from a large set of generated duties the minimum number of middle and late duties such that all workpieces which cannot appear in split duties are covered. Workpieces now covered are removed from the schedule.
- **Stage 2** Those pieces of work which would be covered by early duties are identified and a small set is generated from which some early duties are selected by using a mathematical model. Of these duties, those which would start later than split duties have to be chosen. Workpieces covered by these early duties are then removed from the schedule.
- **Stage 3** Extra early duties and all half duties are chosen to cover the remaining two thirds of the vehicle schedule. A set partitioning model ensures that each piece of work is covered exactly once and an assignment algorithm is then used to optimally pair the half-duties.

This method has the effect that interaction is lost between each stage. Side constraints are introduced to limit the number of drivers taking mealbreaks at certain times, avoiding the need to form unnecessary split duties. Also, many side constraints which are used are specific to its implementation in Christchurch.

2.2.6 The HASTUS System

HASTUS is a complete scheduling package developed originally in 1974 by the University of Montreal's Center for Research on Transportation, and consequently in collaboration with GIRO inc., Canada. The system is widely used throughout the world and many papers have been published regarding its method and results [19, 20, 21, 22, 23, 24]. Apart from its use in scheduling vehicles and drivers, related software is available for passenger information systems, transit operation systems and advanced planning tools. Much work has taken place in order to provide companies with a user-friendly system including a database providing interfaces to other systems and a dictionary allowing the use of local terminology. Versions are available in several languages and for several installations.

HASTUS-macro and HASTUS-micro are the driver scheduling components of the HASTUS package although a new driver scheduling system called Crew-Opt has been developed in recent years and will be described in more detail in the column generation review in section **2.6.1** and also in section **6.4.2**.

The method involves first splitting the day into short periods, the time interval being specified by the scheduler, and then forming many possible idealised duties which are valid according to the labour agreements imposed upon them. The duties generally include two stretches and start and finish at the beginning of one of the small time periods. An integer linear programming model then finds the minimum cost schedule from the idealised duties ensuring that each bus which is operational during the defined short time period is covered by at least one duty. A matching procedure then amends the individual duty stretches and fits them to the actual buses using a least squares fit. The stretches are then paired optimally and heuristically amended to produce the final schedule.

2.2.7 The HOT II System

HOT II (Hamburg Optimisation Techniques) is marketed by HanseCom and can provide either individual scheduling modules or a comprehensive planning system. The complete system incorporates modules to address individual tasks; data management, sensitivity analysis, vehicle scheduling, driver scheduling and driver rostering; all accessing information from a common database. Improvements have taken place resulting in the most recent version which has been described by Völker and Schütze [27] but many of the concepts introduced in the original version HOT [25, 26] remain.

The driver scheduling process is basically heuristic and may need some extensions to adapt to the needs of new users.

The method schedules duties in order of type. Firstly the early duties are formed one by one by splitting almost all of the pieces of work on each bus up to the latest relief opportunity which could be included, and then recombining them to derive many valid duties. Wage and penalty costs which are added to each duty define which one is chosen at any stage to be included in the schedule. An attempt to form mealbreak chains aids the ordering of the formation of duties. Once all reasonable early duties have been formed, a similar process forms the late duties.

The remaining work is recombined into bus work and then cut to form half-duties which are combined using an assignment algorithm to form respectively straight and split duties. Any uncovered work is left for manual amendment or overtime work.

The most recent version attempts to minimise the reliance on schedulers' knowledge to optimise the schedule. It has introduced more parameters to reduce the user interaction and can generate four-part duties.

2.2.8 The INTERPLAN System

INTERPLAN [35] is an interactive system for driver scheduling and rostering using an heuristic algorithm to split the vehicle work and combine the resulting trips to form a selection of the best duties from many possible valid duties. A matching algorithm is used to determine the optimal combination of two-part duties.

2.2.9 The RUCUS II-System

RUCUS-II (RUn CUtting and Scheduling) [36, 37] is a newer version of the original RUCUS system [38] which was developed in the 1960s and is now believed to be obsolete. The original RUCUS system used the technique of making improvements to an initial set of duties but was difficult to operate as many parameters had to be defined and many runs were necessary to achieve satisfactory results. RUCUS-II improved the parameter specification entry and slightly altered the method of producing the initial solution, but still used the same heuristics which consider swapping parts of duties or shifting them to an alternative relief opportunity. The heuristic search is still very inefficient and the enhanced version of RUCUS-II has amended the switching and swapping heuristics to use a matching algorithm which minimises a cost function. This version achieved better results than previously although they are dependent upon the quality of the initial schedule.

2.3 Driver Scheduling Processes

Apart from research into mathematical programming and heuristic methods of solving the driver scheduling problem, some work has been carried out to investigate if other techniques

could be used to improve part or all of the processes involved in scheduling.

2.3.1 Driver Duty Estimator

Contained within the TRACS II system is a decomposition process to solve larger problems, which incorporates an analysis of the bus schedule to produce a lower-bound estimate of the number of duties that will be needed to cover the bus work. This idea provoked further investigations into how the estimator may be made more sophisticated, providing a tighter lower bound, by incorporating knowledge from individual schedulers [39, 40]. There have been few previous attempts to analyse the bus scheduling problem in order to take advantage of the structure of the problem and match duties to its critical features.

Work began in 1992 on developing an heuristic search system based upon a driver duty *Estimator* whose function is to predict how many duties will be required to cover a given set of bus work. This process then requires a new duty generation process which builds up the schedule progressively. A *Generator* uses rules gained from knowledge of the schedule to cut down the search space. An additional process called the *Organiser* then performs the necessary ordering and backtracking of the search based upon updated information from the Estimator after each new duty has been added to the partial schedule.

Further work concentrated on developing the driver duty Estimator rather than incorporating it into a search system. Zhao, Wren and Kwan [41] reported that the Estimator would also provide other useful information, such as which relief opportunities are critical and a quick evaluation of how the driver schedule will be affected by any changes in vehicle schedule. The latter is a useful tool for users whose existing systems require complete system execution for each individual vehicle schedule. The Estimator incorporates rules based upon knowledge of both current problems and previous problems and becomes more accurate with the growth of its rule-base. The Estimator calculates the duty requirement based on the number of vehicles in service throughout the day, in particular the maximum number in the peaks gives an absolute minimum number of drivers required and the schedule is further analysed regarding mealbreak chains and other factors to give a tighter lower bound. Zhao [42] provides a full description of the Estimator method. Results give identical total duty estimates to the TRACS II system for all problems tested, although the individual breakdown of duties differed because the use of different relief opportunities may result in different duty types appearing in the schedule estimate.

2.3.2 Genetic Algorithms

A feasibility study was carried out by Wren and Wren [43] in order to test whether or not genetic algorithms could be used to solve larger driver scheduling problems more robustly, more quickly and more cost-effectively than any of the methods detailed in the previous sections. The initial test aimed to replace the Integer Linear Programming technique within TRACS II, which selects the final schedule from a large previously generated set of duties. By representing the pieces of bus work as chromosomes such that the values of each gene identifies the duties which cover it, many possible valid schedules can be formed by discarding or choosing duties from the complete set until the bus work is covered and no duty is redundant. Processes based upon genetic algorithm techniques of crossover (forming a schedule from a combination of two or more others) and mutation (slightly altering a schedule in some small way) can be applied to the schedules in the hope of producing better solutions and allowing a limited number of schedules to evolve. Results obtained using a genetic algorithm approach without the option of mutation and with limited constraints on small test problems produced very good solutions quickly, encouraging further investigations.

Further work was carried out by Clement and Wren [44] which depicted chromosomes as an unordered set of duties, each with a binary value dependent on whether or not the duty is present in the schedule. Different methods of crossover and mutation techniques were experimented with and tested on three real world problems. Although the genetic algorithm was successfully applied to real world scheduling problems with relatively limited research, the results produced were generally poorer than those of more established techniques. However, genetic algorithms potentially perform better on larger problems.

Further investigations into the use of genetic algorithms to solve the driver scheduling problem involved combining its strengths with the strengths of the intelligent rule based Estimator described in section **2.3.1** [45]. This hybrid technique uses the genetic algorithm approach to determine which set of relief opportunities will produce the best schedules. The 'fitness' (quality) of any subset of relief opportunities depends upon the number of duties needed to cover the bus work defined by the subset, and this can be predicted by the Estimator. Duties are then generated using only the relief opportunities selected and the Integer Linear Programming technique used within TRACS II is utilised on the smaller duty set.

The introduction of genetic algorithm techniques to solve scheduling problems continues to be

an active area of research.

2.4 Review of Computer-Aided Bus Driver Scheduling

Although earlier attempts at driver scheduling were heuristic based and later attempts are mathematical programming based, it is often difficult to categorise systems as many now use a combination of the two.

Heuristic systems (Rural Based System, COMPACS, HOT II) rely upon the knowledge of expert schedulers to build schedules or restrict the duty formation to those duties which are likely to appear in good schedules. They often have the advantage of being understood and modified more easily by the user of the computer system. The disadvantage of using heuristic based systems is that the schedules produced are unlikely to improve significantly upon the manual schedule and the methods cannot consider optimisation routines.

Many systems use matching algorithms (BDS, HASTUS, RUCUS-II, INTERPLAN) to solve parts of the driver scheduling process. These involve cutting the bus work into part-duties and then matching these in pairs to form complete duties. Although some systems have been developed to incorporate three-part duties, separate rules have to be formed to do so and tend to be considered after two-part duties have been paired. Matching algorithms also cut the bus work without taking into consideration the overall bus schedule and so it is possible that the part-duties cannot be all be paired to form valid duties or do not form an overall efficient schedule leaving pieces of work to be covered by overtime pieces. Also, matching approaches tend not to be able to incorporate any side constraints.

Mathematical Programming approaches (Crews First, EXPRESS, TRACS II) work on the definition that each variable represents a duty, and constraints represent pieces of work that need to be covered. Set partitioning models therefore attempt to cover each piece of work with exactly one duty, and set covering models allow pieces of work to be overcovered. Since the number of potential valid duties for any problem is typically very large, most systems incorporate heuristics to eliminate duties which are unlikely to be used in good schedules, and so the reduced model can be solved much faster. The disadvantages of using mathematical programming approaches are that optimality is limited to the set of duties generated, and the scheduler

has little control over the process after initially setting up any constraints. The scheduler may also wish to add further constraints depending on the resulting schedule. Although computing time for these systems tends to be longer than for those of heuristic or matching based systems, the schedules produced tend to be better.

The methods described in this thesis are based on a version of TRACS II, which is described in full in Chapter 4.

2.5 Overview of Other Driver Scheduling Systems

2.5.1 Train Driver Scheduling

Wren et al. [46] observed that although many bus driver scheduling systems have been developed and reported over the years, there appeared to be no practical driver scheduling system for the rail industry. Some of the differences between bus driver and train driver scheduling include the larger distances between relief opportunities for rail services, the peaks of vehicle service, and the often 24 hour service of trains. However, train driver scheduling is considered to be more similar to bus driver scheduling than is air crew scheduling, and a TRACS II-based model was successfully adapted to estimate likely costs of each of a wide variety of rail scheduling problems. Parker et al. [47] detail the modifications which were needed to accurately model the train driver rules, and also the proposed changes to the rules which were tested to observe their effect on the resulting schedules. Much more work has been done recently to further customise the system to train driver scheduling [48].

2.5.2 Airline Crew Scheduling

The airline crew scheduling problem requires duties to consist of a sequence of flights over a number of days such that the crew start and end their duty at the same airport. Labour agreements will vary greatly from those of the bus and rail industries as mealbreaks do not have to be specifically formed and typically only a maximum working time and a minimum rest period need to be defined. All recently published airline crew scheduling systems appear to use mathematical programming based methods which tend to be solved more easily than similarly solved bus driver scheduling problems.

Marsten and Shepardson [49] use a set partitioning method to solve the relaxed LP model over a set of generated duties and then perform a branch and bound search to form an integer schedule. Results report that in many cases the continuous solution was integral and where it was not, an integer solution could be found very quickly.

Graves et al. [50] describe an interactive system which solves a sequence of subproblems by mathematical or local search techniques and can produce optimal solutions for small problems.

Hoffman and Padberg [51] have developed a system which uses a set partitioning method and can run independently of any company. This system also produced many integer LP solutions and for the remaining problems an integer solution could be formed after applying a cutting plane method.

Some papers have been published which use column generation techniques to solve the airline crew scheduling problem and some of these are discussed in section **2.6.2**.

2.6 Column Generation Applications to Transportation

Set partitioning or set covering models are guaranteed to find optimal solutions to the driver scheduling problem only if all valid duties can be represented and the model does not use any heuristics which may compromise the optimality in any way. Most mathematical programming driver scheduling systems limit the generation of duties initially and/or heuristically reduce the generated duty set before solving the model. Also, branch and bound processes often terminate the search for an integer solution once one has been found, in order to reduce execution times.

At the point of solving a mathematical programming model using the simplex method, columns are *priced out* to find a new column which will improve the solution. Column generation methods attempt to create new columns as required so as to implicitly consider all valid columns without the need to generate them beforehand. Ford and Fulkerson [52] describe the method where the problem is a shortest-path problem and Dantzig and Wolfe [53] describe the decomposition of a linear programming problem. Early applications of a column generation technique were reported by Gilmore and Gomory [54] which used column generation principles to solve

the cutting-stock problem.

Although column generation techniques have successfully been applied to many problems, e.g. the binary cutting stock problem [55], graph colouring problems [56], and traffic assignment problems [57], sections **2.6.1, 2.6.2** and **2.6.3** refer only to column generation methods which have been implemented to solve transportation problems.

2.6.1 Bus Driver Scheduling

Carraresi et al. [58] describe mathematical models and algorithms which have been used to solve bus driver scheduling problems, but propose its application for driver scheduling of other transportation types. A network is defined where the nodes represent feasible pieces of work and the edges represent feasible duty parts, where the corresponding nodes can be allocated to the same driver without violating any union restrictions. Solving the Lagrangean dual of a corresponding network model will provide a solution lower bound and a reasonably sized set partitioning problem, but column generation and row deletion techniques are required to adequately solve such a large problem. Columns are generated which have negative reduced costs relative to the Lagrangean multipliers by using a k-shortest paths enumeration.

Crew-Opt [59, 60] is a new driver scheduling module implemented within the HASTUS system, which uses a column generation technique to solve a set covering formulation of the problem. A detailed description of the system will be given in section **6.4.2**. The system uses a shortest path algorithm to generate further duties which have negative reduced costs until the optimal solution to the relaxed model has been reached. A branch and bound algorithm then forms an integer solution where necessary. Rousseau [61] reports on the results obtained by using Crew-Opt on a number of different problems. Bus companies which assign drivers line-by-line produced solutions at least as good as the HASTUS-Micro solutions although solution times are not given for these problems. Typically the smaller problems consist of schedules containing up to 40 drivers. Crew-Opt also produced better results than those produced manually for a problem with around 77 duties but a problem with around 160 duties had to be decomposed in three subsets. The larger problem also produced better results but on the version available the execution time was around 24 hours on a Sun Sparc10/31.

2.6.2 Airline Crew Scheduling

The airline crew scheduling problem has previously been modelled as a set partitioning problem and requires the assignment of flight crews to a schedule for a single aircraft type at the minimum cost. A duty period is a single workday for a crew and these are then paired to form a sequence of duty periods with overnight rests between them. The formation of duty periods and pairings are restricted by legal considerations but the formation of duty periods is heavily constrained so that the problem lies in the large number of possible crew pairings from these periods.

The problem of assigning crews to flight services at minimum cost has been investigated by Minoux and Lavoie et al. [62, 63]. The crew pairings are formulated as valid paths on a graph which is deduced from the flight service information. At each optimal subproblem the crew pairing with the minimum reduced cost amongst the remaining crew pairings equates to finding the shortest path in the graph. The column generation subproblem is then solved using Dijkstra's algorithm. Early experimentation introduced only one column per iteration which proved time consuming and so multiple pricing was introduced to allow more columns to be added for each call to the column generation procedure. For each call the method introduces any negative reduced cost crew pairing into the subproblem. For any crew pairing provided by the generator its *mirror image*, relating to flight services over the second week of a two week period, is also added since it is intuitive that it will also have a negative reduced cost. Results from this implementation report some significant cost savings over manual solutions and execution times are competitive compared to previous computational methods. In a high proportion of cases the solution for the relaxed model is integral as a consequence of the integrality property of network flow problems, but the paper suggests that a way of forming near-optimal integer solutions from fractional solutions is to apply an integer linear programming routine to the set covering problem consisting of the generated pairings.

Vance et al. [64] present a column generation approach for solving the domestic daily flight problem which operates over a hub and spoke structure. The problem is decomposed into two stages of selecting duty periods and then forming the crew pairings so that the pairings are built from good sets of duty periods. Two types of column then need to be generated in order to ensure overall optimality. Columns representing duty periods can be priced out by solving a set partitioning problem with costs derived from the sum of the cost and simplex multiplier for that period. This can be used as a lower bound to terminate the column generation process

when a solution within a specific tolerance of the optimal solution has been found. Columns with negative reduced costs representing crew pairings are derived by a constrained shortest path procedure over a network. Results were reported to converge more quickly with feasible starting solutions rather than artificial ones but the formulation was further modified to speed up convergence. An alternative strategy for generating duty period columns involves using general mixed integer programming techniques to solve the set partitioning problem, and where the LP to be solved is large a *keypath* formulation allows duty exchanges which represent possible modifications to the set of duty periods. The pairing subproblem is then only solved after every sixth call to the duty period subproblem as it takes more computational effort. The solution provided by the LP relaxation of the decomposed system gives a better bound than that from the traditional set partitioning problem over the pairings but is more difficult to solve.

Barnhart et al. [65] tackle the problem of the long-haul flights which typically do not operate on a daily basis. The problem becomes one of reducing the amount of extended rest periods for crews by assigning them to flights as passengers for repositioning. This process is known as *deadheading*. The column generation method on the linear programming relaxation begins with an initial set consisting of some or all of the airline's flights and then selects further flights to add to the set from which deadheading is allowed. At any iteration the selection of flights to add to the subset is based upon eliminating certain types of flight which are unlikely to improve the crew pairing problem, eliminating all flights which do not price out favourably, and finally all flights which are not contained within the most negative reduced cost pairings. The pricing strategy is achieved by using an approximation scheme which estimates the minimum reduced cost pairing containing any deadhead by evaluating the reduced costs of partial pairings and summing them along arcs in a network. Calculations for many potential deadhead flights can then be achieved by summing an arrival profile at a station, a departure profile at another station and the costs of the deadhead flight, and if this is negative then the flight prices out favourably. The process terminates when the objective improvement between iterations is sufficiently small and the deadheads then provide the integer programming package with a good set of crew pairings from which to choose.

Brusco et al. [66] consider the problem of scheduling ground personnel at airports. The relaxed model is solved over a subset of possible legal duties and the dual variables are used to select a further subset of candidate duties whose reduced costs are calculated. The generation method is not explicitly defined. Only the duty with the greatest potential objective improvement is added to the duty subset at each iteration. Once a good continuous solution has been found,

the generated duties and the number of employees assigned to these duties are considered by a local search heuristic which attempts to find the lowest cost solution.

2.6.3 Vehicle Scheduling

The vehicle scheduling problem requires a set of timetabled trips to be covered by a fleet of vehicles, possibly from different depots, such that the cost is minimised. The solution procedure differs from that of driver scheduling because no labour agreement rules need to be considered. Since the model differs from that used to schedule drivers no algorithmic details regarding the column generation procedure will be given here.

Riberio and Soumis [67] and Bianco et al. [68] use column generation techniques to solve the multiple depot vehicle scheduling problem. Bianco et al. [69] extend the multiple depot vehicle scheduling problem to freight transport. Reddington [70] incorporates a column generation technique into a solution procedure to schedule locomotives to work a set of timetabled trains.

2.7 Review of Column Generation Approaches

Column generation systems for driver scheduling solve a relaxed set partitioning or set covering model and generate further duties from the dual prices. In this way all columns can be considered implicitly whilst retaining a much smaller set of columns from which to produce the optimal continuous solution. The method of generating further duties varies widely between problems, with methods consisting only of heuristics, a shortest path enumeration, or an elimination technique. Also, the number of columns selected to enter the subset varies and can range from only one column to all further columns which have a negative reduced cost. One of the difficulties inherent in column generation systems is the costing of columns. In particular, network representations of problems consider an objective function which merely minimises duty costs which may not include any penalties for undesirable features, and may not adequately prioritise an objective which wishes to minimise the number of duties in a final schedule. The pricing strategy may also vary in order to reduce calculations or ensure an efficient set of duties is considered. Where an integer solution needs to be formed different methods have been adopted, including a branch and bound technique on the generated duties, a column generation technique within the branch and bound, and a local search heuristic.

Column generation methods which have been used to solve the driver scheduling problem

can produce guaranteed optimal solutions on small problems, but the duty costing does not apparently incorporate any penalty costs added to duties with undesirable features. This makes the network formulation less applicable to the driver scheduling problem presented in Chapter 1. Also, although it is reported that the methods will be extended to solve much larger problems, currently they are embedded within a heuristic process so that integer schedules are no longer guaranteed to be optimal. Results, however, are encouraging and a column generation strategy which does not use a network formulation is implemented within an existing set covering system and described and tested in this thesis.

Chapter 3

Inherent Scheduling Problems Explored

3.1 Introduction

In order to investigate the problem of bus driver scheduling more fully, it is appropriate to consider some of the problems and complexities encountered by both manual and computer schedulers and look at possible methods of solution.

This chapter is split into two sections. The first is concerned with driver mealbreaks, where efficient changeovers of drivers to form breaks is fundamental in creating a solution with the minimum workforce. The second section investigates the problems associated with attempting to find an optimal schedule by considering every possible duty combination from a previously generated set.

3.2 Mealbreak Chains

A subset of the driver scheduling problem is concerned with efficient timetabling of driver mealbreaks.

Figure 3.1 shows typical bus work at the beginning of the day for three vehicles, where the

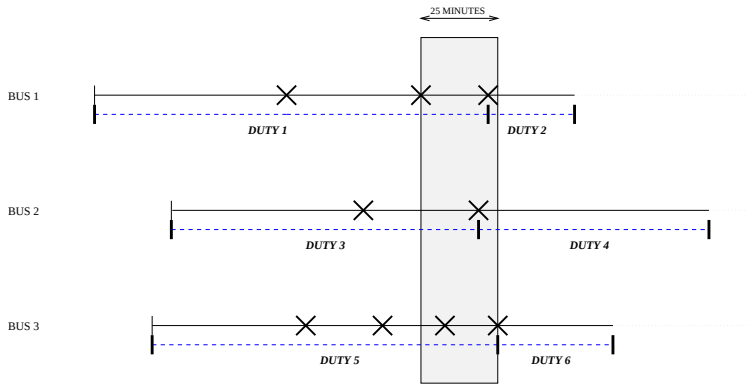

Figure 3.1: A Possible Duty Combination for a Portion of Bus Work

"X's" are available relief opportunities and the shaded portion contains the latest relief opportunities at which the first driver can leave the vehicle. This maximum time limit for a driver to work without a break is specified in the labour agreement rules. Supposing that the minimum length of a mealbreak is 30 minutes and the width of the shaded portion only represents 25 minutes, then it is not possible in this scenario for a driver to finish his/her mealbreak and return to relieve another driver within this time interval. Since the objective is to minimise the total number of duties, the worst possible case for the example is where the first three drivers work the maximum time allowed for a first stretch and three new drivers are allocated to the remaining work as indicated on the diagram. Hence the total number of drivers needed to cover this bus work would be six (assuming the existence of other buses does not create other duty combinations).

Now consider figure 3.2. If the driver of bus 1 is relieved at an earlier relief opportunity, it is possible for the driver to finish his/her mealbreak and return to relieve the driver of bus 3 at the point shown. This driver in turn can then return to bus 2 after a mealbreak and the total number of drivers needed reduces to four.

When it becomes necessary for drivers to take mealbreaks it is more efficient to relieve them by drivers having already finished mealbreaks as opposed to new drivers.

In turn this suggests that :

A driver schedule with the minimum number of drivers will require mealbreak chains to be formed, to limit the number of new drivers that are introduced throughout the schedule.

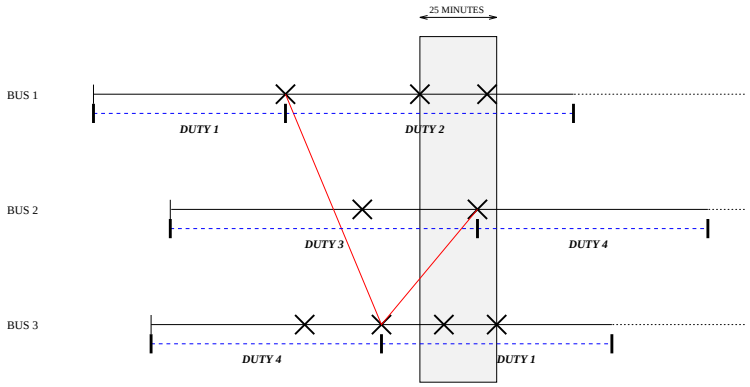

Figure 3.2: A Possible Mealbreak Chain

Since the chaining of mealbreaks is fundamental in producing an efficient schedule, driver scheduling systems which systematically build duties without regard to the complete vehicle schedule are unlikely to produce satisfactory results. It is also the case though that if the mealbreak chaining mechanism is built up systematically then the later duties which are formed may be very short or invalid.

3.2.1 Mealbreak Chain Example

Consider the bus schedule as shown in Figure 3.3. The relief opportunities are marked with their respective relief times and each bus has been labelled alphabetically for the purpose of identification. Note that the example given is being used to demonstrate the principles behind mealbreak chaining and hence depicts only a subset of an operative bus schedule. Also it treats all times in the labour agreement rules as strict times so as not to introduce further complexities. These properties will be discussed later.

It is unusual for a minimum stretch length to be specified as a labour agreement rule, but since the introduction of computer scheduling it has become a feature introduced to reduce the number of valid duties which could be formed. The minimum stretch length is a lower limit normally agreed by the scheduler and the bus company to avoid creating inefficient duties. Together with maximum stretch lengths we have a window of times within which we can consider possible relief opportunities for any drivers.

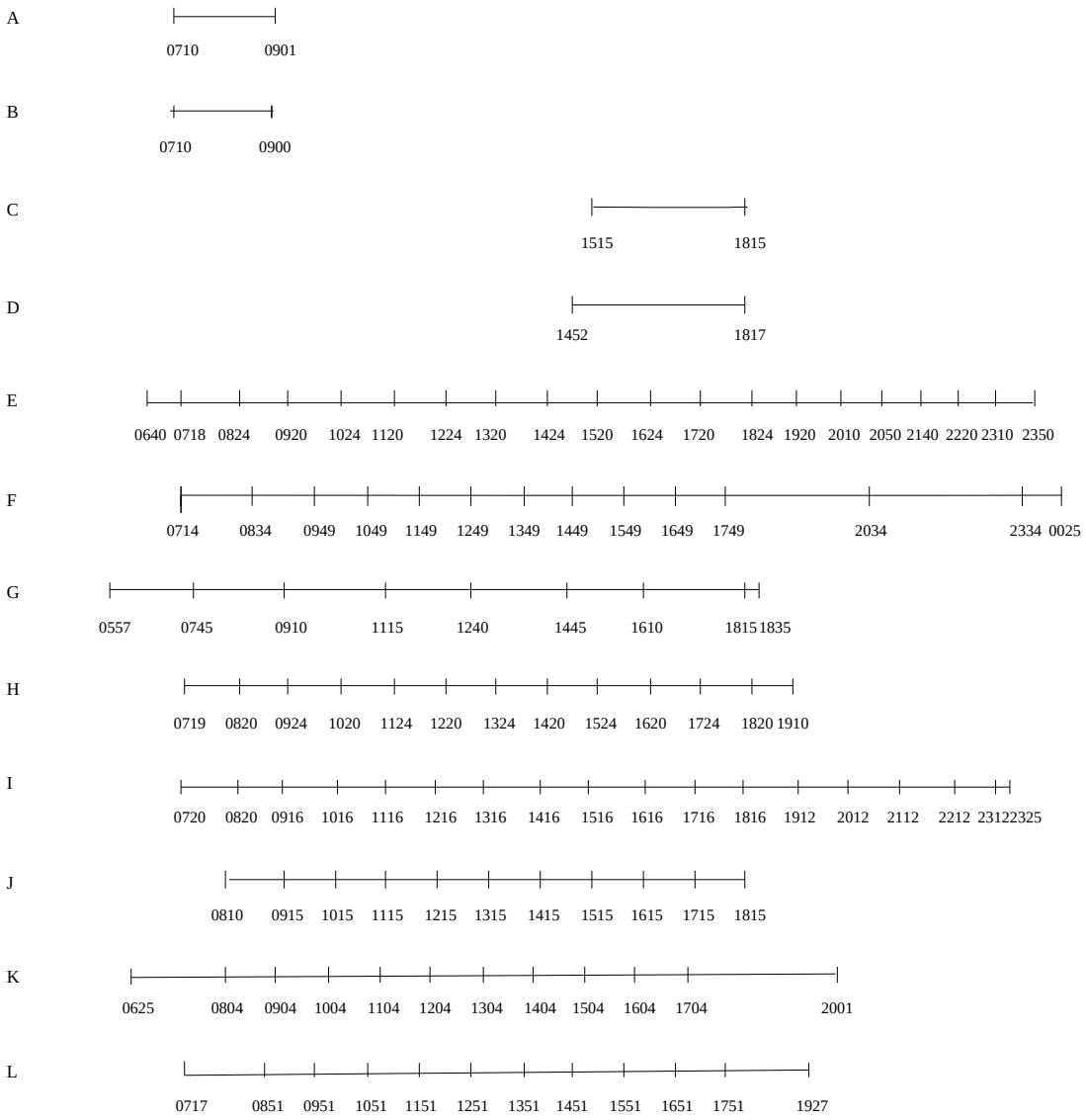

Figure 3.3: A Busgraph Representing Part of the Cleveland Transit Operation

By first observing drivers taking the morning bus out of the garage we are at present only dealing with duties which would be classed as EARLY. Hence we only need consider the relevant time limits for such duties as specified in the labour agreement rules.

For this problem:

the MINIMUM LENGTH OF 1ST STRETCH is defined as 2 hours and 15 mins
the MINIMUM LENGTH OF 2ND STRETCH is defined as 2 hours
the MAXIMUM STRETCH LENGTH is defined as 5 hours and 30 mins
the MINIMUM LENGTH OF A SPELL is defined as 1 hour
the MINIMUM LENGTH OF A MEALBREAK is defined as 1 hour
the MAXIMUM LENGTH OF A MEALBREAK is defined as 1 hour and 25 minutes

Since the majority of bus companies prefer to avoid three-part duties, for simplicity they will only be considered through necessity. In this example the interval between the earliest and latest start times is 2 hours and 13 minutes. This is insufficient to allow any driver to complete the first stretch of a duty and be allocated to the first journey of another early bus after a mealbreak. This implies that ten drivers are required to begin their duties on buses A,B, and E to L.

The drivers leaving buses A and B at 0901 and 0900 respectively have done insufficient work for our defined first stretch length of 2 hours and 15 minutes. They are valid first spells and thus will have joinups to other buses, creating necessary three-part duties. Therefore this gives the following list of possible relief opportunities available to the other eight drivers having taken the vehicles out of the garage:

```
E1 : 0920  E2 : 1024  E3 : 1120
F1 : 0949  F2 : 1049  F3 : 1149
G1 : 0910  G2 : 1115
H1 : 1020  H2 : 1124  H3 : 1220
I1 : 1016  I2 : 1116  I3 : 1216
J1 : 1115  J2 : 1215  J3 : 1315
K1 : 1004  K2 : 1104
L1 : 1051  L2 : 1151
```

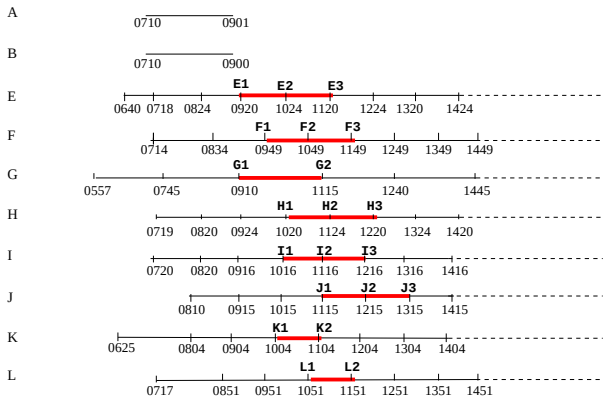

Figure 3.4: The Window of Relief Opportunities Available

Figure 3.4 displays the working subset of relief opportunities on the early section of the bus-graph. As in the previous example, for any one vehicle the window of opportunities is not sufficiently wide to allow an early stretch to be formed within it, implying that as soon as one of the above relief opportunities has been chosen for each bus the others can be eliminated as further possible changeovers.

E1 and G1 will be chosen as the most efficient joinups for the drivers having taken buses A and B from the garage, since they are the earliest available changeovers thus reducing waiting time at the relief point. (These are valid changeovers in that travel time between the two relief opportunities does not exceed the available time) Since we are using E1 and G1, we can omit relief opportunities E2, E3 and G2 as potential links in a mealbreak chain.

A table of possible changeovers can be formed using information regarding the acceptable lengths of mealbreaks, and from this table it can be shown that since F3 never appears as a possible changeover it is superfluous to our problem at this stage, although it may be reintroduced later. Figure 3.5 represents the simplified version of the table. The numbers entered in the boxes represent costs, which are calculated to be the excess minutes over a one hour mealbreak. The bold sections identify groups of relief opportunities from which only one can be selected as a changeover of drivers. From Figure 3.5, it can be seen that an upper bound on the number of mealbreak assignments is four as there are at most four relief opportunities which can begin a second stretch.

The object is thus to :

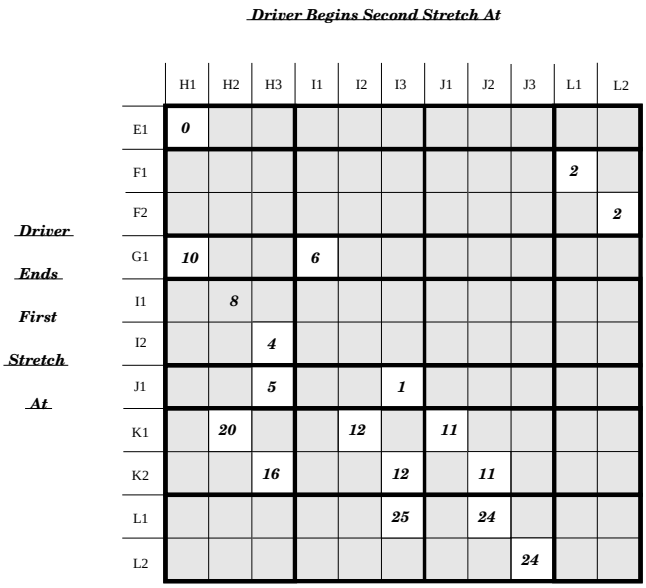

Figure 3.5: Matrix of Potential Mealbreak Links

MINIMISE THE TOTAL COST OF CHAINING THE MAXIMUM NUMBER OF MEAL-BREAK LINKS.

Since the example given is relatively small, by repeated examination of the possible links it was possible to deduce the optimal solution as :

```
E1->H1
F1->L1
K1->J1->I3

TOTAL NO. OF LINKS = 4
TOTAL COST = 14 minutes
```

It was considered how to solve the general problem of assigning mealbreaks so as to achieve optimality for any given part of a vehicle schedule.

3.2.2 Possible Solution Methods

ASSIGNMENT PROBLEM

The task is to assign as many second stretches to drivers having completed their mealbreaks as possible, i.e. given the above example there are 11 potential relief opportunities where drivers can begin their mealbreaks and 11 available relief opportunities which can start the second stretch. A feature of the Assignment Problem is that *ONE* relief opportunity would be assigned to *ONE AND ONLY ONE* driver. The problem need not be square in that a dummy variable can be assigned to any extra drivers. Given that the optimal solution contains only four links some invalid assignments would have to be formed, although high costing on such possibilities would discourage choosing too many invalid links over valid ones. Invalid assignments would have to be removed from the final solution.

The main complication arises in that the row(column) entries are not necessarily independent of each other. In the example it is not permitted to use more than one of the available relief opportunities for any one bus because it leaves too short a spell of work to be covered. In the general case there may be some relief opportunity combinations which are valid and some which are not. Methods of solving the assignment problem, e.g. the Hungarian algorithm, involve choosing an assignment and then eliminating its corresponding row and column entries in the table. For the dependent-entried table, groups of column and row entries would have to be eliminated, as identified in the table. This is further complicated by the fact that rows and columns are not independent of each other either, and an assignment of one relief opportunity in a particular column(row) affects other rows(columns) containing alternative relief opportunities for that bus.

A possible way to overcome the above complications might be to choose only one relief point for each bus and solve the resulting table as an assignment problem using standard techniques. However, even using such a small example as the one above there are 216 combinations of relief opportunities and each selection leads only to a local optimum. The solution could be refined by swapping relief opportunities and resolving the problem at each stage creating a tree search, but intelligent heuristics would be needed to decide on which relief point to branch. Certain costly links can be bounded because their inclusion is unlikely to improve on the current best solution. For large problems however, since the penalty costs are low in comparison to the total cost of a feasible solution, it is less obvious which relief opportunities are likely to be detrimental to the overall solution.

NETWORK PROGRAMMING

It was also considered how to solve the problem graphically. The difficulty of choosing only one relief point per bus to start a second stretch can be overcome by introducing one sink node for each bus. Similarly the beginning of mealbreaks can be depicted as source nodes with only one source node per bus. The network flow diagram is illustrated in diagram 3.6.

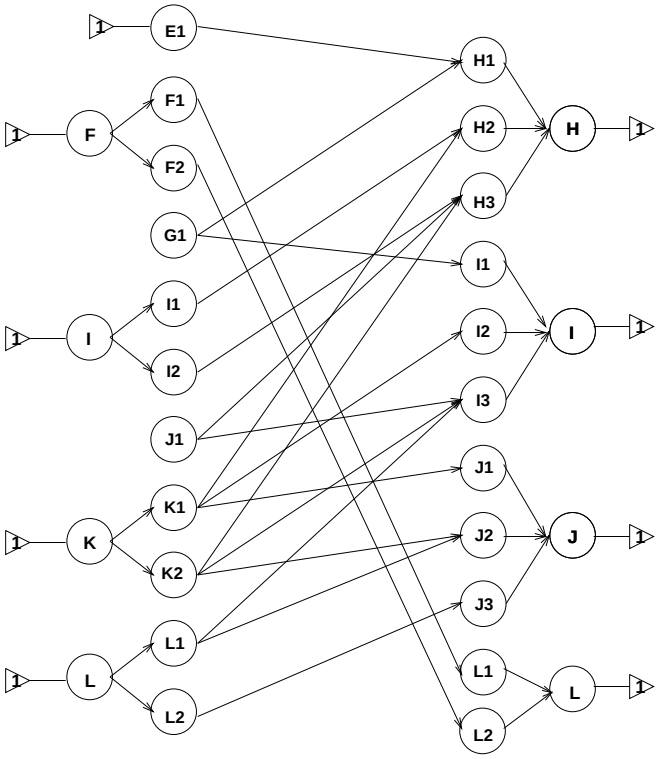

Figure 3.6: Netform Representation of Mealbreak Chaining Problem

On each link a lower bound of 0 and an upper bound of 1 is placed. This reflects whether a link is used or not. The previously defined costs can then be placed on each link with a zero cost to/from the newly defined source and sink nodes. By solving the network program using KHNET [71] on a Silicon Graphics (Iris Indigo Workstation with 33MHz R3000 MIPS processor) the optimal mealbreak chain is ascertained.

However there remains the problem which occurred with the assignment problem technique regarding different relief opportunities on the same bus being available as both the start and end of a mealbreak. The netform diagram shows that whilst a restriction is placed on the

combinations of either source or sink nodes, there is no restriction banning certain sink nodes from being used if certain source nodes are used and vice versa. These constraints cannot be formulated in a manner which preserves the (linear) network program.

One possible approach which can incorporate such constraints is Lagrangean Relaxation. The constraints can be modelled in integer programming terms and used as a penalty on the objective cost if the constraints are not satisfied.

MATHEMATICAL PROGRAMMING MODEL

By using the variables $link_{ij}$ to represent every cell in the matrix and the parameters $used_{ij}$ to ban invalid links, we can formulate the problem in mathematical programming terms. The objective is to minimise the cost of using the maximum number of mealbreak links, subject to constraints which ensure that the combination of mealbreak links which are selected is valid.

The main objective is to maximise the number of mealbreak links and so this can be isolated as the only objective at present; the cost objective can be considered afterwards.

The objective function can be written as:

$$\text{Maximise} \quad \sum_{i=1}^{11} \sum_{j=1}^{11} used_{ij} link_{ij} \tag{3.1}$$

$$used_{ij} = \begin{cases} 0 & \text{if cell (i,j) contains an invalid link} \\ 1 & \text{otherwise} \end{cases}$$

$$link_{ij} = \begin{cases} 1 & \text{if cell (i,j) is used as a link in the solution} \\ 0 & \text{otherwise.} \end{cases}$$

We need constraints to ensure that a driver relieves only one other and similarly that one driver is relieved by only one who is eligible to do so. This is equivalent to ensuring that there is only one entry per row or column of the table.

$$\sum_{i=1}^{11} link_{ij} \leq 1 \quad \text{for all j = 1,11}$$

$$\sum_{j=1}^{11} link_{ij} \leq 1 \quad \text{for all i = 1,11}$$

(3.2)

Once a relief opportunity has been chosen for a particular bus the others are eliminated as possible changeover points. These constraints involve ensuring that at most one relief point per bus can have the value 1. The first case to be considered is the columns, where only one column can be chosen from each block shown in the matrix.

Constraints :

$$\sum_{j=1}^{3} \sum_{i=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus H - points 1,2,3)}$$

$$\sum_{j=4}^{6} \sum_{i=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus I - points 1,2,3)}$$

$$\sum_{j=7}^{9} \sum_{i=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus J - points 1,2,3)}$$

$$\sum_{j=10}^{11} \sum_{i=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus L - points 1,2)}$$

(3.3)

The equivalent constraints for the rows of the table are as follows :

$$\sum_{i=2}^{3} \sum_{j=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus F - points 1,2)}$$

$$\sum_{i=5}^{6} \sum_{j=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus I - points 1,2)}$$

$$\sum_{i=8}^{9} \sum_{j=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus K - points 1,2)}$$

$$\sum_{i=10}^{11} \sum_{j=1}^{11} link_{ij} \leq 1 \qquad \text{(Bus L - points 1,2)}$$

(3.4)

However, there is also the situation where some relief opportunities for buses appear both in rows and columns. Therefore every combination of choice of relief opportunity for a particular

bus must be eliminated.

$$\begin{aligned}
&\textstyle\sum_{j=1}^{11} link_{5j} + \sum_{j=5}^{6}\sum_{i=1}^{11} link_{ij} \leq 1 && \text{(Constraint on relief point I1)}\\
&\textstyle\sum_{j=1}^{11} link_{6j} + \sum_{i=1}^{11} link_{i4} + \sum_{i=1}^{11} link_{i6} \leq 1 && \text{(Constraint on relief point I2)}\\
&\textstyle\sum_{j=1}^{11} link_{7j} + \sum_{j=8}^{9}\sum_{i=1}^{11} link_{ij} \leq 1 && \text{(Constraint on relief point J1)}\\
&\textstyle\sum_{j=1}^{11} link_{10j} + \sum_{i=1}^{11} link_{i11} \leq 1 && \text{(Constraint on relief point L1)}\\
&\textstyle\sum_{j=1}^{11} link_{11j} + \sum_{i=1}^{11} link_{i10} \leq 1 && \text{(Constraint on relief point L2)}
\end{aligned} \tag{3.5}$$

The package XPRESS-MP [72] on a Silicon Graphics (Iris Indigo Workstation with 33MHz R3000 MIPS processor) was used to solve the model for this problem to give:

```
E1->H1
G1->I1
K1->J1
F1->L1

TOTAL NO. OF LINKS = 4
TOTAL COST = 19 minutes
```

To then minimise the cost, we could again use the same constraints as above, in a mathematical model, with the new objective function :

$$\text{Minimise} \quad \textstyle\sum_{i=1}^{11}\sum_{j=1}^{11} cost_{ij} link_{ij} \tag{3.6}$$

Rather than using the $used_{ij}$ values to ban certain links from a solution, it is adequate to set high costs for these links, so that the minimisation process distinguishes between them.

The previous objective solution would then be added in as a constraint:

$$\sum_{i=1}^{11} \sum_{j=1}^{11} link_{ij} = 4 \tag{3.7}$$

This gives the optimal solution :

```
E1->H1
F2->L1
K1->J1->I3

TOTAL NO. OF LINKS = 4
TOTAL COST = 14 minutes
```

This procedure accurately indicates the most efficient chaining of mealbreaks for a given time interval and corresponding relief opportunities. However, even on such a small problem, many constraints had to be modelled. The amount of computational effort would increase enormously for larger problems, particularly if smaller mealbreaks are allowed (increasing the number of possible links) or if more complicated labour agreement rules were available (increasing the number and complexity of constraints).

ALTERNATIVE METHODS

In the example, constraints on combinations of relief points which could occur in a solution create an obvious upper bound on the number of allocations which could be formed. It may be possible to directly use this figure in the cost minimisation as the definitive number of links necessary for a solution. The value would be reduced where no solution could satisfy this constraint. However, it will not always be the case that only one relief point from each bus can be chosen, as this was solely determined from the labour agreement limit on the minimum length of a stretch. In practice a spell within a three-part duty could cover a considerably smaller length of time on any bus, and restrictions on relief point combinations become fewer. Hence, in theory, the upper bound could be merely the lower of the number of rows or number of columns in the matrix.

Using the above approach takes two separate models to determine the minimum cost solution. Since the first model merely returns the maximum number of links, where the details of the resulting mealbreak allocations are not required, an alternative strategy may be useful. The two objective functions could be combined, making sure that the objective which ascertains the maximum number of links is prioritised by a weighting factor. This could be achieved by using a Sherali weighted objective function which has been implemented by Willers [14] into the TRACS II scheduling system developed at Leeds University and is detailed in section **5.3**.

For this problem, combining the objectives first involves ensuring that the objectives do not conflict by turning both into minimisation problems. A Sherali weight is then added to prioritise the maximisation of the number of mealbreak chains, which should be defined as an upper bound on the objective cost.

In this example this evaluates as (maximum number of links) * (maximum cost of link).
The maximum number of links could be calculated crudely as the minimum of the number of rows or columns, i.e. 11, and the maximum cost is the maximum allowed minutes over a one hour mealbreak, i.e. 25. By then introducing high costs to discourage invalid links from being included in the solution, the Sherali weighted objective function is then :

$$\text{Minimise} \quad \sum_{i=1}^{11} \sum_{j=1}^{10} -275 * (25 - cost_{ij}) link_{ij} . \tag{3.8}$$

This has been proved to produce the optimal solution using XPRESS-MP [72]. The method speeds up the the process by only having one optimisation stage, but it is still be necessary to model all constraints as before.

LIMITATIONS

The method of using a mathematical model to solve the mealbreak chaining problem requires human interaction to ensure that the relief opportunities and potential links chosen are valid within a complete schedule, so that duties or part-duties formed can be incorporated into it.

The following list describes some of the difficulties a scheduler would have to consider in order to set up the mathematical programming model.

- For each chaining problem careful note must be taken of whether a relief point is the start of a mealbreak, the end of a mealbreak, or potentially both. With a combination of start times and end times available it may be the case that constraints on combinations of relief opportunities become much more complicated to generate. Also, by selecting a relief point which acts as the start of a second stretch, a limit may be placed on the length of the previous driver's work.

- The depth of search is also important. By considering a large selection of relief times unnecessary computation may be introduced, along with a wider variety of duty types which can be considered. However, if the time span is not wide enough then vital assignments may never be considered.

- Some assignments are not valid due to the non-availability of convenient relief times later in the day, e.g. a driver may exceed the maximum spreadover because an assignment to a second stretch provided no opportunity to leave the vehicle within the specified interval.

- All labour agreement rules would have to be strictly adhered to without the benefit of hindsight, previous experience, or impression of the overall work which needs to be covered. For any alteration in times the whole process has to be repeated. It would be difficult to develop intelligent heuristics which could relax some soft labour agreement rules in order to produce the most efficient schedule, if the relief opportunity selection were to be automated.

- Where no chains can be found at any stage decisions would have to be made as to where new duties should be introduced, or which duties will be split over the peak bus work.

3.2.3 Using Mealbreak Chaining to Produce a Driver Schedule

Using mathematical programming methods to find mealbreak chains at intervals throughout the day, a driver schedule was formed for the given example with 20 duties; the driver scheduling package TRACS II found a solution with only 18 duties. The major discrepancies between the solutions were caused by the relaxation of some of the soft labour agreement rules in the TRACS II version. In particular the short working on buses A and B were allowed as valid stretches.

Also TRACS II produced a two hour period which was covered by more than one driver. The tendency was for the chaining solution to include many duties whose total spreadover was under the minimum paid day value and hence quite inefficient for the bus company as well as requiring more duties in total. Since the chaining problem maximises the number of changeovers it will often reduce the working content of a duty to create an additional changeover rather then using a later opportunity in the next time interval. Also, whether the solution progresses chronologically or towards the middle of a bus schedule, due to the 'greedy' nature of the solution method the existence of efficient or valid duties in the final stages of forming a schedule cannot be guaranteed. Manual swapping and shifting of duties may be needed to complete the process.

3.2.4 Conclusion

The difficulties described above suggest that it would be extremely difficult to build an automatic system which could create a schedule based solely on systematically forming mealbreak chains. The resulting schedule may serve the purpose of creating an initial solution but is unlikely to produce the optimal solution.

3.3 Exhaustive Search

A driver schedule for any given problem can be guaranteed to be optimal over its available duty set if every possible duty combination has been explored. It would obviously be too time- and memory-consuming to consider every possibility for realistic data sets, but an exhaustive search which incorporates heuristics to limit searching on certain schedules may be a feasible technique for finding the optimal driver allocation for subproblems.

3.3.1 Search Tree

A driver schedule consists of group of duties which adequately[1] cover a bus schedule. By depicting the duties as branches and building up a tree structure based upon valid combinations of duties, a schedule will be any complete route through the tree. Once an initial solution is found, duties can be 'swapped' at nodes to see if any improvements can be made. The current best solution can then be used to limit searching on certain branches in two ways:

[1] It is preferable to cover each piece of work with exactly one driver, but in some circumstances it is accepted that some pieces of work are covered by more than one duty and the duties are manually adjusted afterwards.

- The number of duties in the current best schedule will limit the depth of search, since the number of branches equates to the number of duties.
- The current lowest cost schedule will limit searching on branches where it is known that by adding further duties the cost will be exceeded. This is achieved most efficiently by searching in increasing cost order.

Figure 3.7 displays a very small set of test data which will be used to illustrate a simple implementation of a search tree.

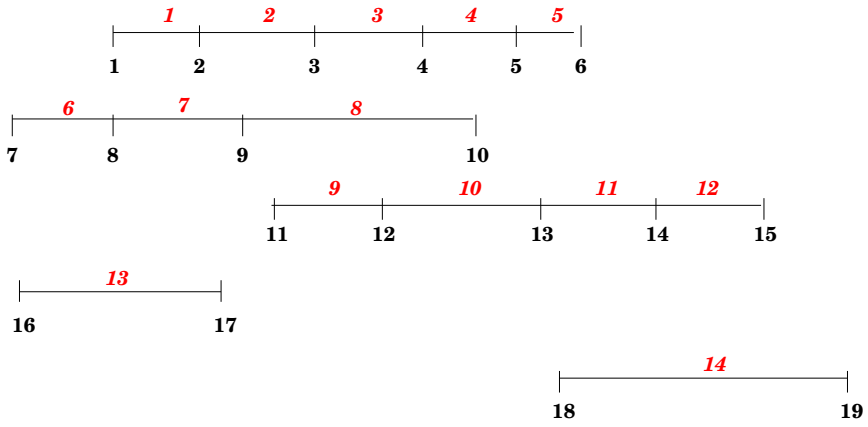

Figure 3.7: Busgraph Test Data to Illustrate Searching Procedures

Since schedules are based upon pieces of work being covered it would be beneficial to store the duties in lists depending on the pieces of work that they cover. One method might be to have a list of pieces of work and for each a corresponding list of duties which cover it. Each time a piece of work is uncovered this duty list can be traversed and a duty chosen to be included in the current schedule. However, for each piece of work there will be a large number of duties which cover it and since duties often cover many pieces of work there will be a large amount of repetition. A method has been implemented which extracts each unique spell that is contained within the duty set. Since certain spells will appear often, the list size will be limited. Each spell can then be given an identifier and a corresponding list of duties which contain it, and the original duty list can be written in terms of the spell identifier that it covers. Duties with only two spells will then only appear in two lists.

Duty no.	Pieces Covered (Stretch 1)	Pieces Covered (Stretch 2)	Spell ID 1	Spell ID 2	Cost
1	$2 \rightarrow 3$	$11 \rightarrow 12$	1	2	389
2	$3 \rightarrow 3$	$14 \rightarrow 14$	3	4	396
3	$13 \rightarrow 13$	$14 \rightarrow 14$	5	4	399
4	$6 \rightarrow 8$	$14 \rightarrow 14$	6	4	400
5	$9 \rightarrow 9$	$5 \rightarrow 5$	7	8	407
6	$8 \rightarrow 8$	$11 \rightarrow 12$	9	2	412
7	$1 \rightarrow 2$	$14 \rightarrow 14$	10	4	421
8	$7 \rightarrow 8$	$12 \rightarrow 12$	11	12	430
9	$1 \rightarrow 2$	$10 \rightarrow 12$	10	13	441
10	$6 \rightarrow 6$	$9 \rightarrow 11$	14	15	442
11	$13 \rightarrow 13$	$11 \rightarrow 12$	5	2	450
12	$1 \rightarrow 3$	$11 \rightarrow 12$	16	2	451
13	$1 \rightarrow 3$	$14 \rightarrow 14$	16	4	456
14	$9 \rightarrow 10$	$14 \rightarrow 14$	17	4	457
15	$2 \rightarrow 3$	$14 \rightarrow 14$	1	4	462
16	$6 \rightarrow 7$	$4 \rightarrow 5$	18	19	466
17	$13 \rightarrow 13$	$3 \rightarrow 5$	5	20	467

Table 3.1: Possible Valid Two-Part Duties for the Example Problem

Without the use of any labour agreement files to ensure that duties are valid, table 3.1 displays a small list of possible two-part duties for problem shown in Figure 3.7, in terms of the pieces of work that each spell covers. The duties have been sorted in order of cost to aid the pruning of the search tree. As each unique spell is identified it is labelled. Table 3.2 then displays all unique spells in the generated duty set, along with corresponding lists of those duties which incorporate each spell.

3.3.2 Solution Strategy

It is necessary firstly to construct an initial solution and then attempt to improve upon it either by finding a schedule with fewer duties, or one with the same number of duties but at a lower cost. This can be done by appending duties to part-schedules until all of the bus work is covered. By allocating a flag to indicate the status of the pieces of work it is possible to

id	start of spell	end of spell	duties including this spell	id	start of spell	end of spell	duties including this spell
1	2	3	1,15	11	7	8	8
2	11	12	1,6,11,12	12	12	12	8
3	3	3	2	13	10	12	9
4	14	14	2,3,4,7,13,14,15	14	6	6	10
5	13	13	3,11,17	15	9	11	10
6	6	8	4	16	1	3	12,13
7	9	9	5	17	9	10	14
8	5	5	5	18	6	7	16
9	8	8	6	19	4	5	16
10	1	2	7,9	20	3	5	17

Table 3.2: List of all Spells Appearing in Generated Duties

ascertain when a complete schedule has been formed. It is then possible to backtrack and see if any improvement can be made by making different choices of duty to be included at each stage. Every duty combination can be depicted as a series of branches (duties) in the tree formulation. Where it is known that no improvement can be made to the current best schedule by exploring certain branches they can be fathomed.

Initial Solution

Initially no piece of work will be covered. The search looks through the list of statuses to find the first uncovered unit, i.e unit one. By inspecting the list of available spells it can be seen that spell id 10 covers units 1 and 2, and spell id 16 covers the first 3 units. By arbitrarily considering spell id 10 first it is possible to identify duties 7 and 9 which are suitable for our current search. Since the duties are in cost order duty 7 will be chosen as the cheapest alternative, and its second spell also covers the units relating to spell id 4. By defining the values; -1 to represent a piece of work which is covered, and 1 to represent pieces of work which are not covered, the status values for the problem are now updated to coincide with the choice of duty as follows :

```
-1 -1  1  1  1  1  1  1  1  1  1  1  1 -1
```

The first uncovered unit is now piece three, i.e a driver needs to take over bus one at relief point three. For each duty that is available to cover this unit we calculate two values. *MOST*

signifies the total number of units that this duty will cover in the schedule, and *PENALTY* signifies how much of this duty is already covered. Hence we will attempt to find duties which cover the least previously covered work, and from this set the duty covering the most work is chosen. Where more than one duty displays equally good characteristics the cheaper duty will be chosen. By introducing a measure of the quality of duty added at each stage it is hoped that a good initial solution will be found, and the better the initial solution the more branches can be fathomed in the branching strategy.

The following list displays the corresponding *MOST* and *PENALTY* values for all appropriate duties covering unit 3.

```
Spell ID 1 :
   Duty 1  : MOST = 4,    PENALTY = 1
   Duty 15 : MOST = 3,    PENALTY = 2

Spell ID 3 :

   Duty 2  : MOST = 2,    PENALTY = 1

Spell ID 16 :

   Duty 12 : MOST = 5,    PENALTY = 2
   Duty 13 : MOST = 4,    PENALTY = 3

Spell ID 20 :

   Duty 17 : MOST = 4,    PENALTY = 0   <---
```

The most appropriate duty covering unit 3 is duty 17. (In practice, to reduce time, searching is halted as soon as the first duty with a value of 0 for *PENALTY* is found).

Again the unit statuses are updated, and this process is continued until all pieces of work have been covered.

Figure 3.8 shows the first branch of the search tree, which forms an initial solution for the simple example. The initial solution has 4 duties with no overcovered pieces of work, at a total cost of 1760

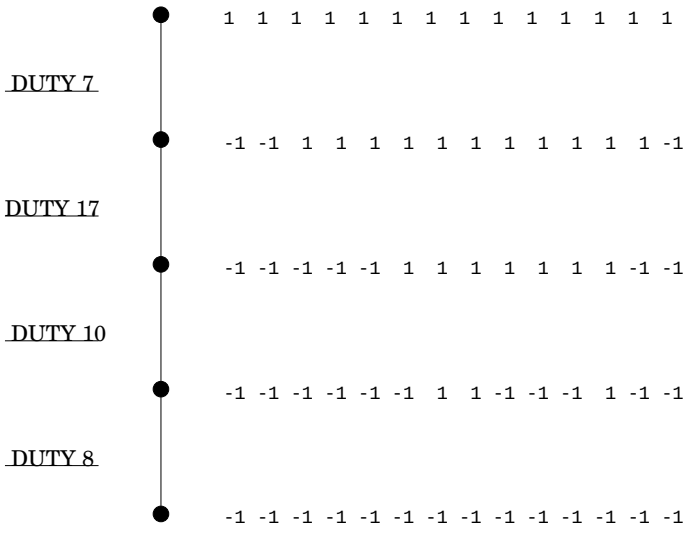

Figure 3.8: Construction of an Initial Solution

Branch And Bound

The objective now is to either

i) find a solution with ≤ 4 duties
or ii) find a cheaper solution with 4 duties.

A 'depth-first' search is used so that duties added to the initial schedule later will be swapped for other appropriate duties in order to see if any improvement in the final schedule can be made. As all possibilities become exhausted the method backtracks to earlier workpieces.

There are two available pruning methods.

- Firstly, if the number of duties exceeds four then we need look no further along the branch.

- Secondly, if at any later stage in the tree a set of unit statuses identical to one already explored is encountered, they are compared. Since we no longer take into account the overcover of duties, but merely branch in cost order, unless this new node has less duties to reach this point we need not look any further down this branch.

Figure 3.9 shows how a search tree for the simple problem develops, reading from left to right.

A tree search as outlined above was coded onto a UNIX workstation in the C programming language. The final solution produced is the optimal solution :

```
DUTY 9
DUTY 17
DUTY 4
DUTY 5

COST = 1715
```

This result was achieved in 0.2 seconds and traversed 63 nodes; however the schedule itself was formed at the 36th node.

3.3.3 Results

In order to test the feasibility of the tree search on realistic problems the program was run on a relatively small data set containing 4199 duties. After 1000 nodes and 3.5 hours, the best solution found contained 39 duties. The known optimal integer solution contains 34 duties. This solution method is unrealistic on real data sets in terms of time and memory.

3.3.4 Possible Improvements

The search tree is an obvious method of guaranteeing an optimal solution over a set of duties, but is very time consuming. The heuristics which have been added to the branching strategy serve to reduce the problem size without compromising optimality but it is still inefficient even on small problems.

The current method of selecting the piece of work to cover is based solely on its logical numbering within the problem. Since the branching strategy relies on a depth first search it would be

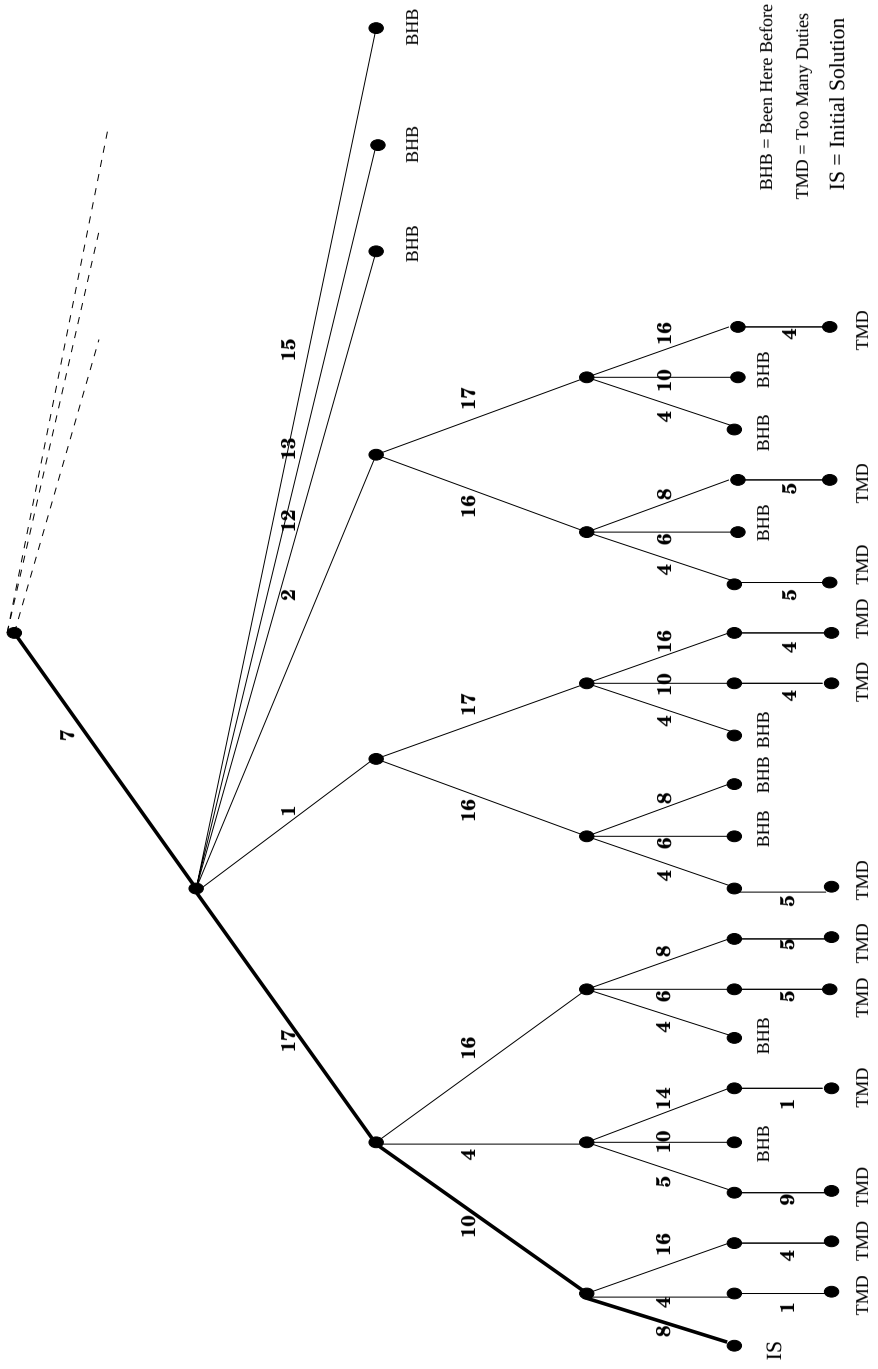

Figure 3.9: Formation of a Search Tree

more efficient if the pieces of work chosen earlier in formation of the schedule had fewer duties available to cover them. This would then allow the schedules to develop more quickly at the bottom of the tree. It could be implemented by selecting the pieces of work to be covered in the initial solution in increasing order of the number of duties covering them.

Further possibilities for the search tree include incorporating more intelligent heuristics to reduce the execution time of the program. However, in order to reduce the timings to what would be acceptable to users may require introducing heuristics which may compromise the optimality of the solution. One possible heuristic which could be incorporated is an Estimator developed by Zhao [40, 41, 42] and described briefly in section **2.3.1**. The Estimator would aid the pruning of the tree by giving information as to how many duties are needed to cover the schedule, and which duties are likely candidates for the final solution. Since the Estimator was originally written for the PC some alterations were required to operate it on the UNIX workstation. No more work has yet been done on including this into the tree as the Estimator works on different input files.

3.3.5 Applications

If memory and time allowed, the exhaustive tree search would be the obvious method for obtaining an optimal subset of duties to cover a bus schedule. Hence it was a useful exercise to program this search so that tests can be made to determine the size of problem that can easily be solved by this method. If a future improvement to the algorithms involves reducing the size of the duty list, or if a problem creates a relatively small duty list, the exhaustive search would be a possible method of solution. Alternatively, the tree search could be extended to include heuristics to prune the tree, so that larger data sets could be used.

3.4 Overview

The problem of assigning mealbreak chains throughout a driver schedule is essential in producing an efficient schedule, but the number and complexity of constraints inherent in any problem make it difficult to solve. Certainly methods can be adopted which optimally pair stretches of work for any period in time but this cannot be done systematically to produce the final schedule any more efficiently than existing systems.

An exhaustive searching mechanism would find the optimal schedule for any given set of generated duties, but without heuristic reduction it is currently unrealistic as a driver scheduling system.

These two examples emphasize the difficulties inherent in driver scheduling problems. The remainder of this thesis is concerned with the TRACS II scheduling system developed at the University of Leeds, and ways of improving its mathematical programming component which produces a schedule from a set of previously generated valid duties.

Chapter 4

The TRACS II Scheduling System

4.1 Introduction

This chapter describes a computerised scheduling system which was developed at the University of Leeds by Smith and Wren [15, 16, 17]. The IMPACS (Integer Mathematical Programming for Automatic Crew Scheduling) system was installed by the University in London Transport (now London Buses Ltd.) in 1984 and Greater Manchester Transport in 1985. It has been maintained and altered for commercial use within the BUSMAN system [73] whilst an updated version is retained at the University of Leeds for research purposes and will be referred to as TRACS II (Techniques for Running Automatic Crew Schedules). Both versions attempt to find driver schedules using the same processes and the following sections outline both the proposed method of solution, and the separate stages of solution. Subsequent chapters will describe alternative strategies for part of the system.

4.2 The TRACS II Model

TRACS II consists of a suite of Fortran programs which first generate a set of valid duties, then reduce the set to a manageable size and finally select from them a set of duties which cover the bus work. Since not all valid duties are available for selection it may be necessary to have some pieces of work covered by more than one duty. These pieces of work are referred to as being

overcovered.

Once a set of valid duties has been formed we can define N and M where,

N denotes the total number of duties available
M denotes the total number of pieces of work in the bus schedule.

Since pieces of work are defined as indivisible time intervals between two relief opportunities, it is assumed that any relief opportunities which are not used by at least one duty will be deemed redundant and the adjoining pieces of work combined. The following can now be defined :

For $j = 1, .., N$

$$x_j = \begin{cases} 1 & \text{if duty } j \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

For $i = 1, .., M$

$$u_i = \begin{cases} 1 & \text{if workpiece } i \text{ is uncovered} \\ 0 & \text{otherwise} \end{cases}$$

$$o_i = \text{number of times that workpiece } i \text{ is overcovered}$$

(4.1)

The existence of the variables u_i and o_i allows workpieces to remain uncovered and have more than one duty covering them respectively. The desired situation would be where a piece of work is covered by exactly one duty so that the variables u_i and o_i would both have the value zero.

4.2.1 The Objective Function

The complexity of requirements to produce an efficient driver schedule leads to five objectives being necessary. They are listed in decreasing order of importance as:

- To minimise the number of uncovered pieces of work.
- To minimise the number of duties used in the schedule.

- To avoid duties which contain undesirable features.
- To minimise wage costs.
- To minimise the total duration of overcovered pieces of work.

This has been represented by an objective function of the form :

$$\text{Minimise} \sum_{j=1}^{N} C_j x_j + \sum_{i=1}^{M} D_i u_i + \sum_{i=1}^{M} E_i o_i. \tag{4.2}$$

The most important requirement is to remove uncovered pieces of work, indicating that the coefficient D_i should be set at a large value to ensure that this issue is addressed first. Normally a large constant is added to the duration of the particular piece of work so that it becomes expensive to leave it uncovered.

$$\begin{aligned} D_i &= \text{cost of not covering workpiece i} \\ &= \text{(constant to deter undercover)}^1 + \text{duration of workpiece i.} \end{aligned} \tag{4.3}$$

The E_i coefficient on the other hand should be lower to reflect a less important objective. The calculation of this value involves multiplying the duration of the overcovered workpiece by a small constant.

$$\begin{aligned} E_i &= \text{cost of overcovering workpiece i} \\ &= \text{(cost of overcover per minute)}^1 \text{ * duration of workpiece i.} \end{aligned} \tag{4.4}$$

The remaining objectives require the C_j coefficient to reflect duty costs, penalty costs and wage costs. However, high penalty costs reflecting duties which are unattractive to either a bus company or its drivers would normally be added in later so as not to detract from the more important objective of minimising the number of drivers. To prioritise the objectives a large constant is added to each duty cost so that minimisation favours a schedule with fewer duties. The duty costs themselves are expressed as the number of minutes of work for which a driver is

[1] Constants used in weighting the objectives are defined by the scheduler in the ZIP parameter file. An example of the values used can be found at the end of this chapter.

paid. If work content is such that the guaranteed minimum wage is higher than the duty cost, the duty cost will be replaced by this higher value.

$$\begin{aligned} C_j &= \text{cost of duty j} \\ &= \text{wage cost} + (\text{constant to prioritise reduction of duty total})^1. \end{aligned} \tag{4.5}$$

4.2.2 Constraints

Since some duties will be inefficient and the time taken to solve the model is dependent upon the number of variables used, heuristics are used to limit the number of duties to be entered into the model. The heuristics attempt to retain individually efficient duties and hence discard many shorter potential duties. This may result in a final set of duties which do not easily fit together without overlapping when considered over the whole bus schedule.

For this reason a set partitioning model which would ensure that every workpiece is covered by *exactly* one duty is not used. Although it is unacceptable to schedulers to allocate more than one duty to the same piece of work, a set covering approach is chosen which ensures that every workpiece is covered by *at least* one duty. This can be written as:

$$\sum_{j=1}^{N} A_{ij} x_j \geq 1 \text{ for } i = 1, .., M \tag{4.6}$$

where the A_{ij} identify which pieces of work are covered by which duties:

$$A_{ij} = \begin{cases} 1 & \text{if duty j covers workpiece i} \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

Since the objective function includes minimising wage costs and also the amount of overcover, in general very few pieces of work will be covered by more than one duty in the final schedule. In practice any overcover which appears in a solution will most commonly occur when two duties overlap. In this situation the duties can be manually edited so that only one driver is assigned to any piece of work, with any excess being put on standby. It is also possible that larger

amounts of overcover are produced, but in these cases it often highlights an inefficiency caused by a condition in the labour agreement. If set partitioning were to be used on these problems it would be much more difficult to determine the reason for achieving either no solution or a very expensive solution.

The TRACS II system is based upon an elastic set partitioning model [74] where both slack and surplus variables are assigned to each constraint so that the model is effectively set covering. Constraint **(4.6)** can therefore be written as follows:

$$\sum_{j=1}^{N} A_{ij}x_j + u_i - o_i = 1 \text{ for } i = 1, .., M. \qquad \textbf{(4.8)}$$

Manington [75] used equality constraints to cut down the problem size when forming an initial solution, and the inclusion of such constraints has remained in the model as a requirement by the constraint branching strategy within the branch and bound algorithm, as described in section **5.4.3**.

In order to incorporate some equality constraints into the model it is important to note in which situations it is particularly preferable not to allow overcover. The number of drivers needed to cover the morning peak is governed by the number of buses in service at that time plus any drivers required to cover problematic pieces of work. Hence, by allowing overcover on the first few pieces of work for any bus, this will introduce an extra driver signing on before the rise in the number of vehicles in service. Similarly, it is not advisable to allow overcover on the last pieces of work for a bus as it necessarily implies inefficiency later in the schedule. Manual schedulers tend to form earliest-starting and latest-finishing duties first because they restrict the way that the other duties are formed, and hence they are unlikely to produce overcover on these corresponding pieces of work.

The algorithm incorporated into TRACS II automatically identifies the workpieces which shall be formulated as equality constraints, chosen in such a way that no early duty covers more than one of the early equality constraints, and no late duty covers more than one of the late equality constraints.

Hence, constraint **(4.8)** can now be split into the following:

$$\begin{aligned} \textstyle\sum_{j=1}^{N} A_{ij}x_j + u_i &= 1 \text{ for } i = 1,..,L \\ \textstyle\sum_{j=1}^{N} A_{ij}x_j + u_i - o_i &= 1 \text{ for } i = L+1,..,M \end{aligned} \tag{4.9}$$

where L denotes the number of workpieces which can be identified as equality constraints.

4.2.3 Side Constraints

It was mentioned earlier that a bus company can specify six duty types *EARLY, LATE, MIDDLE, SPLIT, DAY, OVERTIME*. The user can then impose certain constraints on any of these duties to ensure that the final schedule does not contain too many or too few of any particular type. Constraints of this form can also be used to limit the number of three-part duties occurring, limit the total number of duties in the final schedule, and also eliminate undercover. The constraints can be imposed either initially or when reoptimising an existing solution.

The constraints can be expressed as follows:

$$\textstyle\sum_{j=1}^{N} \delta_{kj}x_j \leq U_k \tag{4.10}$$

where U_k is an upper limit on the number of duties of type k, and

$$\textstyle\sum_{j=1}^{N} \delta_{kj}x_j \geq L_k \tag{4.11}$$

where L_k is a lower limit on the number of duties of type k.

The δ_{kj} are defined as :

$$\delta_{kj} = \begin{cases} 1 & \text{if } x_j \text{ is to be in the side constraint} \\ 0 & \text{otherwise} \end{cases}$$

(4.12)

Given then that each x_j is classified as exactly one duty type at the generation stage, for any side constraint on this duty type the corresponding value of δ_j will be one. Similarly, duties can be classed as two or three-part duties so that a side constraint limiting the number of three-part duties will require that the δ_j value for such duties is one. If there is a side constraint limiting the total number of duties in the schedule then all δ_j constants will have the value one.

4.2.4 Model - Summary

Minimise $\sum_{j=1}^{N} C_j X_j + \sum_{i=1}^{M} D_i u_i + \sum_{i=1}^{M} E_i o_i$

Subject to $\sum_{j=1}^{N} A_{ij} x_j + u_i = 1$ for $i = 1, .., L$

$\sum_{j=1}^{N} A_{ij} x_j + u_i - o_i = 1$ for $i = L+1, .., M$

Plus any user-defined side constraints

$x_j = 0$ or 1, for $j = 1, .., N$

$u_i \geq 0$

$o_i \geq 0$

4.3 Running TRACS II

In order to create a driver schedule which covers an existing bus schedule it is necessary to go through a number of different stages. Each stage corresponds to running a Fortran program

and the details of each of these can be found in the user manual [18] and are explained briefly in the following sections.

4.3.1 BUSGRAPH

A program is available to print the bus schedule in graphical form, i.e. a line graph displaying the times that each bus is in service and at which times relief points occur. Although it is not necessary, it is wise to use this facility early on in order to analyse the driver requirement and relief opportunities. It may also be beneficial to print the busgraph after certain relief opportunities have been discarded.

4.3.2 DIV

Sometimes a problem would be classed as 'too large' to be solved based upon the frequency with which buses pass relief points, the range of acceptable duty lengths, the number of duties having been generated etc. For this reason restrictions on problem size have been coded within TRACS II, which are currently a maximum of 22000 duties and 1000 pieces of work. In practice TRACS II has solved a problem with 429 constraints and 10775 duties in around 15 minutes. As computer technology improves, the limits defining 'too large' can be increased, but the ability to split the bus schedule into smaller components may still be useful. The program works by analysing the bus schedule and placing individual bus workings into groups to form separate subproblems. Each grouping should contain work with complementary characteristics so that they will be covered by the same set of duties, thus minimising the number of extra duties created through certain buses no longer being combined. In the process of splitting the data the program produces an estimate of the likely numbers of duties needed in the final schedule, and identifies any relief times which may be critical in the formation of a good schedule.

4.3.3 STIMES

According to Smith [16], in order for the model to be useful both the number of variables and constraints must be restricted. As mentioned in the previous section, each piece of work will correspond to a constraint in the mathematical programming model. The STIMES program considers each relief time for the bus schedule entered with the purpose of selecting or rejecting it, based mainly on the allowed stretch lengths of duties. A whole bus schedule or a subproblem can be studied. Note that rejected relief points at this stage can be reinstated by the user

if required. Any duties generated will only be formed using the relief opportunities selected during this process, and hence the total number of duties entering the model will be reduced.

4.3.4 GEN

This is the program which generates valid duties for the bus schedule in question. As well as using the duty generation parameter file (see section **4.4.2**) which contains some of the labour agreement rules, three specially written Fortran subroutines, often specific to the bus company, are also used to ensure the validity of the duty, incorporate any penalty costs, and calculate wage costs.

Although the task of the duty generation process is to form valid duties, because of the limitations on problem size required to enter the mathematical programming model, a few simple rules are used to limit the formation of less sensible duties. For instance, very short duties may be valid but are relatively costly and unlikely to be found in schedules with the minimum number of duties. Also, for every first stretch of duty that is formed, the potential number of relief times for which the second stretch can start is limited to avoid forming duties with longer mealbreaks. The number of potential three-part duties is very large and so restrictions based upon knowledge of where they may best be utilised in a schedule are used.

Although the duty generation has to be limited, and intelligent heuristics are used to do this, it cannot be guaranteed that a duty vital to the overall optimal solution will be generated. Along with further reductions of the duty size this possibly restricts the quality of the overall schedule, whereas a method which allowed all valid duties to be considered would guarantee optimality.

4.3.5 COMPARE

In order to reduce the number of duties generated, and thus the time it will take to produce a driver schedule, heuristics are needed to eliminate any duties which are unlikely to be useful. COMPARE incorporates one such heuristic, which is that any duty which is contained wholly within another duty will not be chosen as it does not cover as many pieces of work. If the final schedule contains overcover then by editing a duty it may be that the shorter and probably

cheaper duty is ultimately included in the final schedule, even though it was banned at this stage.

4.3.6 EVEN

EVEN uses an heuristic to reduce the duty set still further by limiting any repetitious covering of pieces of work. An upper limit called the *Coverage Value* can be entered by the user, e.g. 50 so that if all the pieces of work covered by a duty are also covered by 50 other duties, then it is not significantly contributing to the choices available and can be dropped. The generated duties are ranked in terms of their efficiency and the EVEN process removes duties in increasing order of efficiency. This efficiency is determined by expression **(5.3)**.

4.3.7 ZIP

Although this program will be discussed in more detail in the following chapter, a brief description will be given as to its purpose. Once the final duty set has been produced ZIP selects from it a driver schedule using the mathematical programming model described in section **4.2**. The strategy firstly involves finding an intermediate solution which need not necessarily be integer, but provides a lower bound on the number of duties required for the final solution. A branch and bound process is then used to ensure that an integer solution is formed, i.e. the duty combinations are feasible.

4.3.8 SPRINT

This program performs some heuristic improvements to the schedule formed by ZIP and then prints a duty schedule in the form of a duty list followed by the busgraph with duties marked on it.

4.3.9 OVER

In the solution to any problem which has been divided, whilst there will be many duties which are acceptable there may be some which are considered inefficient in some way. This program allows the user to identify these latter duties, to carry the bus work contained in these duties over to the next subproblem, and to save satisfactory duties. The program will, if desired, carry over bad duties automatically.

4.3.10 COMBINE

This program is used to combine the resulting schedules for a divided problem into one driver schedule.

4.3.11 INSPECT

It may sometimes be helpful to examine some of the duties formed. A scheduler may wish to outline characteristics of a duty and INSPECT will display any duties in the set possessing these characteristics. For instance, the scheduler can request to see all two-part duties starting at a particular relief time, or see if a complete duty was formed at the duty generation stage.

4.3.12 Improvement routines

Improvement techniques are available for refining a schedule produced by ZIP. It is possible that an integer solution, i.e. feasible schedule, is not optimal and so an improvement routine may find a cheaper solution. Also, since all duties cannot be considered in the set covering model, further investigations may indicate any improvement which could be gained from introducing a duty previously removed from the set. Finally, certain preferences and criteria may not easily be incorporated into the set covering model, such as drivers signing on and off in the same order to balance the workload, and an improvement routine is available to rearrange duties.

4.4 Data Files

Information about the existing vehicle schedule and also the rules governing the formation of duties are required in order to generate the set of duties from which the schedule will be chosen. This information is provided by the bus company and translated into standard formats as shown in the following two files:

4.4.1 Bus Schedule Data File

In order to provide information about the current bus schedule under consideration, a data file is constructed. Information contained within this file will include times and locations at which drivers may be relieved, and travel times to these locations. Example 1 depicts a typical bus schedule data file.

- The first line has a heading of up to 50 characters which will be printed on all output.
- The second line indicates the number of relief points available as potential driver changeover locations.
- The next group of lines relate to the relief points. There is one line for each point, and the number of points must correspond to that specified on the second line of the file. The first point must be the garage. The first character is a code for the relief point to be used in graphical output. This is followed by the name of the point to be used in the duty listing.
- The following lines give, for the work of each bus, the bus running number and then for each relief opportunity, the time and the relief point number. In order to distinguish between the arrival and departure at a relief point, when it is not necessary to allocate a driver to the period in between, this may be indicated by putting zero for the relief point following the departure time. (See bus 5 time 1038 in example 1). A bus leaving the depot for the second time can be identified by altering the bus running number. (Bus 1008 is the second time out of the depot for bus 8).
- The next block of data shows for every pair of relief points, the minimum mealbreak which must be allowed if the break starts at one point and finishes at the other.
- The next block of data shows for every pair of relief points, the portion of the mealbreak which is paid (in mins).
- The next block gives the minimum joinup time required between each pair of relief points (in mins).
- Finally, the last set of data shows the signing on and signing off allowances at each relief point (in mins).

```
Greater Manchester Buses 10/10/1988
2
G Harling Garage
h Harling
1    0522 1  0625 2  0852 2  0955 2  1222 2  1325 2
     1552 2  1655 2  1922 1
2    0555 1  0822 2  0925 2  1152 2  1255 2  1522 2
     1625 2  1852 2  2012 2  2215 2  2312 1
3    0655 1  0922 2  1025 2  1252 2  1355 2  1622 2
```

```
      1725 2  2015 2  2112 2  2315 2  2412 1
4     0725 1  0952 2  1055 2  1322 2  1425 2  1652 2
      1755 2  1918 1
5     0652 1  0755 2  1022 2  1038 0  1125 2  1352 2
      1455 2  1722 2  1825 2  1948 1
6     0537 1  0722 2  0825 2  1052 2  1155 2  1422 2
      1525 2  1752 2  1912 2  2115 2  2312 2  2415 1
7     0516 1  0752 2  0855 2  1122 2  1225 2  1452 2
      1555 2  1822 2  1925 1
8     0624 1  1000 1
1008  1515 1  1900 1
9     0640 1  0737 2  0840 2  0937 2  1040 2  1137 2
      1240 2  1337 2  1440 2  1537 2  1640 2  1737 2
      1840 2  2045 2  2345 1
10    0607 1  0807 2  0910 2  1007 2  1110 2  1207 2
      1310 2  1407 2  1510 2  1607 2  1710 2  1807 2
      1910 1
MINIMUM MEALBREAK
40 35
35 30
PAID MEALBREAK
40 35
35 30
MINIMUM JOINUP
15 10
15 5
SIGNON AND OFF
15 10
15 10
```

EXAMPLE 1 : A Bus Schedule Data File

4.4.2 Duty Generation Parameter File

A data file is created which controls the formation of valid duties from which the schedule will be selected. Some forms of labour agreement rules, such as maximum stretch lengths,

are common to all bus companies and can be parameterised in this file. Other data, such as maximum length of mealbreak, is used to restrict the total number of valid duties that are formed by discouraging the construction of duties which are unlikely to be useful in the overall duty schedule. Example 2 represents a typical duty generation parameter file where all times are shown in hours and minutes, e.g. 245 is 2 hours and 45 minutes, unless specified otherwise. Some headings have been identified by the letters (a) - (y) which are not found in the files, but appear here to clarify the descriptions as follows:

- The first parameter indicates whether a file of selected relief times has been set up or not. If 'NO' is specified, all relief times from the bus schedule are available to be used for the driver scheduling. If a relief time selection process has been used to identify relief times which are unlikely to be useful then the response would be 'YES'.
- The next three parameters are :
  1) The maximum break in a split duty (which minimises unproductive time in long duties).
  2) The maximum length of a joinup (in minutes) between relief points.
  3) The minimum joinup (in minutes). Even if a driver changes buses at the same relief point, this limit allows for late running of the first bus.
- The minimum length of spell (in mins) is not a legal requirement and as such is determined at the discretion of the scheduler to avoid the use of three-part duties and wasteful short lengths of time on any vehicle.
- 'LIMIT' is the minimum length of a spell at the start or end of a running board (in mins). Since the driver is taking the vehicle from or to the garage, it may be necessary to differentiate between this value and that specified above.
- The minimum long break on a split duty can be left at 0 in which case the minimum break will be governed by by the minimum mealbreak values from the bus schedule data file. Some companies, however, specify a much longer time for split duties because of the increased total duty time compared to straight duties.
- The minimum spreadover for a split duty is usually either specifically stated in the labour agreement, or implied by the fact that split duties must have longer spreadovers than other duty types.

The next section gives parameters relating to the five duty types:

EARLY LATE MIDDLE SPLIT DAY

- (a) The minimum length of the first stretch specifies the minimum time from the start of the duty to the start of a mealbreak.
- (b) The minimum length of the second stretch similarly defines the time from the end of the mealbreak to the end of the duty.
- (c) Maximum mealbreak limits the length of a mealbreak to what could realistically occur. The maximum mealbreak for split duties given here only applies to labour agreements which allow 3-bus split duties to have two mealbreaks, in which case this refers to the shorter.

The following four sections can be used to restrict the generation of nonsensical duties.

- (d) Earliest start of mealbreak
- (e) Latest start of mealbreak.
- (f) Earliest finish of mealbreak.
- (g) Latest finish of mealbreak.
- (h) The minimum acceptable cost discourages the system from creating a lot of short duties.
- (i) Minimum cost for three-part duties. Since three-part duties are often very long and not very productive, it may be possible to specify a higher minimum cost than for two-part duties. If three-part duties for a particular type are not to be formed, one may enter a minimum cost value greater than the maximum cost for that duty type.
- (j) Maximum cost is often governed by the labour agreement. In order to differentiate between duty types, a set of timings regarding the work content for each is defined. The following seven lines are self-explanatory.
- (k) Earliest signing-on time.
- (l) Earliest time on bus.
- (m) Latest starting time.
- (n) Earliest finishing time.
- (o) Latest finishing time.
- (p) Latest signing off time.

- (q) Maximum length of stretch without a mealbreak.
- (r) Maximum platform time limits the total time spent on the bus.
- (s) Maximum spreadover limits the total duty span from signing on to signing off. This parameter could be used to prevent certain duty types from being formed if set at zero.
- (t) A minimum daily payment may be specified in an agreement to guarantee a minimum wage irrespective of the amount of time worked.
- (u) It is possible for the user to specify particular duties which should be included in the schedule. It is important to note that these duties will not be checked for validity. This is a feature which is lacking in the commercial system.
- (v) Maximum number of mealbreaks. Since three-part duties are considered, it is possible that rather than a short joinup between buses, the scheduler requires that 2 mealbreaks can be taken in this situation.
- (w) Overtime pieces to be formed. This is where the user can allow one-piece duties to be formed. Stretches of work will be cut from running boards and the remainder is left to be scheduled among the other duty types. However, since these duties vary considerably between companies a separate routine must be written if the response to this header is 'YES'.
- (x) Up to six periods of time can be specified here as times when joinups cannot start. This information limits the number of three-bus duties which are formed.
- (y) Number of second stretches requires an entry for each duty type. When the duties are being generated the first stretch is formed and then a number of possible second stretches after a mealbreak must be considered in order of time. The values placed here indicate the maximum number of second stretches which can be linked with the first stretch to form duties. This heuristic prevents too many duties from being formed with long breaks. It does, however, eliminate valid duties from being formed in which the penalty for a longer mealbreak does not outweigh the relative cost benefit of the duty in a schedule.

```
SELECT
YES
MAX-SPLIT-DUTY-BREAK MAX-JOINUP MIN-JOINUP
600                      40         5
MIN. LENGTH OF SPELL
```

```
      40
      LIMIT
      40
      MIN. LONG BREAK IN SPLIT DUTY
      210
      MIN. SPREADOVER IN SPLIT DUTY
      807
(a)   MIN LENGTH OF 1ST STRETCH
       230  230  230  230  230
(b)   MIN LENGTH OF 2ND STRETCH
       230  230  230  230  230
(c)   MAX MEALBREAK
       130  130  130  600  130
(d)   EARLIEST START OF MEALBREAK
      0000 1430 0930 0000 0000
(e)   LATEST START MEALBREAK
      1315 2130 1830 1125 1315
(f)   EARLIST FINISH MEALBREAK
      0000 1500 1000 1335 0000
(g)   LATEST FINISH OF MEALBREAK
      1345 2200 1900 1700 1345
(h)   MIN COST
       600  600  600  600  600
(i)   MIN COST (3-BUS DUTIES)
       600  600  600  600  600
(j)   MAX COST
       806  806  806 1100  806
(k)   EARLIST SIGNING ON TIME
      0000 0000 0000 0600 0000
(l)   EARLIST START ON BUS
      0000 0000 0000 0559 0000
(m)   LATEST START ON BUS
      0800 1800 1500 0900 0900
(n)   EARLIST FINISH OFF BUS
      0000 2130 1600 1700 0000
```

```
(o)   LATEST FINISH OFF BUS
      1559 3200 2129 1929 1559
(p)   LATEST SIGNING OFF TIME
      1559 3200 2129 1929 1559
(q)   MAXIMUM STRETCH LENGTH
       530  530  530  530  530
(r)   MAXIMUM PLATFORM TIME
       806  806  806 1100  806
(s)   MAXIMUM SPREADOVER
       806  806  806 1230  806
(t)   MINIMUM PAID DAY
       000  000  000  000  000
(u)   PRESPECIFIED DUTIES
      0
(v)   MAXIMUM NUMBER OF MEALBREAK
      1
(w)   OVERTIME PIECES TO BE FORMED?
      NO
(x)   PERIODS WHEN JOINUPS CANNOT START
      0
(y)   NO. OF SECOND STRETCHES
        10   10   10   10   10
```

EXAMPLE 2 : A Duty Generation File

4.4.3 ZIP file

A further file is constructed to contain information regarding the running of the ZIP process. Example 3 represents a ZIP file as it has been used in the version developed at the University of Leeds, but this file has since been updated to incorporate different features and solution strategies throughout the current research.

- The START parameter dictates the way in which the model solver will be executed, depending on whether the user wishes to reoptimise a current solution and on the requirement to continue to find an integer solution.

- The ADD-PENALTIES parameter specifies whether the penalty costs should be included in the duty cost initially or at a later stage.

- The following line of data is used by the program to control screen output and the solution strategy. Some of these properties will be described in more detail in the following chapter.

  The ZIP data file includes the constant coefficients used in the objective function (see section **4.2.1**) where :

  - COST-OF-DUTY defines the large constant added to the wage cost

  - COST-OF-OVER-COVER defines the constant to be multiplied by the duration of the overcovered piece of work

  - COST-OF-UNCOVERED-PIECE defines the constant to be added to the duration of an uncovered piece of work.

- Finally, the user can add any extra side constraints to this file.

```
START  ADD-PENALTIES
  4        0
(IPRINT  NGOOD  ZLIM  REDUC  NITER)
1 10 0.005 TRUE  99999
COST-OF-DUTY COST-OF-OVER-COVER COST-OF-UNCOVERED-PIECE MIN-OVERTIME
   2000            3.000                8000.000           1.000
EXTRA CONSTRAINTS
```

EXAMPLE 3 : A ZIP Parameter File

4.4.4 Info file

The scheduling system produces a data file at the end of the duty generation phase which holds information identifying the relief points that refer to the start of each vehicle. Also, for each

relief opportunity, it calculates the number of generated duties which use it as the start or end of a spell.

4.4.5 Duty file

Once the duties have been generated they are each coded into a file. Each row of this file corresponds to a valid duty with each column representing a particular feature of the duty, e.g.

```
2   1   3  22  24   0   0 475   1   5   475
3 155 157 114 115  28  29 441   3  15  1731
```

The first value refers to the number of spells of work contained within the duty. The next six values pair off to form the code of the start and end relief point for each of the three potential spells of work. Since there are only two spells in the first example, the third pair of numbers are both zero. The next number is the wage cost of duty which was calculated by a specific user routine mentioned in section **4.3.4**. The following number is a code representing the duty type. In this case a 1 indicates that it is an early duty and the 3 indicates a middle duty. The next column gives an indication of the efficiency of the duty. This figure will be explained in more detail in section **5.2**. Finally, the last column gives the total cost of the duty including any penalties for undesirable features. In the first example no extra cost is incurred, which would have discouraged it from being in the final solution. However the second example has a penalty cost added to discourage too many three-part duties from appearing in a schedule.

4.5 Overview

The TRACS II driver scheduling system uses a set covering formulation to produce a schedule from a set of previously generated valid duties. Not all duties are available to the mathematical programming component as heuristics are required firstly to reduce the problem size. The following chapter details the mathematical programming component of TRACS II.

Chapter 5

Solving the TRACS II ILP Model

5.1 Introduction

This chapter contains a brief description of how the mathematical programming part of TRACS II works to select a subset of valid duties in order to cover bus work and minimise costs. Based on experience of this method, alterations and improvements which were made to the original model and method of solution, leading up to the implementation of a column generation technique, will be explained.

5.2 Solution Strategy

At this stage in the solution process a set of potential duties has been defined, along with any user-defined side constraints and constants reflecting the coefficients of the variables in the objective function. It is possible to execute the model solver in a number of different ways, such as starting or stopping at different points in the solution strategy; however, the method will be described in full as if there were no user intervention in the process.

For the purpose of simplicity side constraints will not be included in any of the models outlined in this chapter because they are not affected by the developments described. Thus the original model is represented as follows:

$$
\begin{aligned}
&\text{Minimise} && \textstyle\sum_{j=1}^{N} C_j x_j + \sum_{i=1}^{M} D_i u_i + \sum_{i=1}^{M} E_i o_i && \\
&\text{Subject to} && \textstyle\sum_{j=1}^{N} A_{ij} x_j + u_i = 1 && \text{for } i = 1, .., L \\
& && \textstyle\sum_{j=1}^{N} A_{ij} x_j + u_i - o_i = 1 && \text{for } i = L+1, .., M \\
& && x_j = 0 \text{ or } 1, && \text{for } j = 1, .., N \\
& && u_i \geq 0 && \\
& && o_i \geq 0. &&
\end{aligned}
\tag{5.1}
$$

The model and the terms used have been presented in section **4.2**.

The penalty costs are not normally added into the objective function initially, because duties displaying several undesirable features incur a very high cost, and although it is hoped that these duties do not appear in a schedule it is actually preferable to include some of them rather than to exceed the minimum number of duties.

For this reason the solution strategy has historically been to devise two pre-emptively ordered objectives which are solved in two stages. The first is to minimise a function combining the costs of the individual duties with a fixed cost per duty, so as to give a significant bias towards minimising the number of duties; the second is to minimise a cost function, incorporating penalty costs, with the added constraint that the total number of duties does not exceed the minimum already ascertained at the end of the first stage. These objectives can be formulated as follows:

$$\begin{aligned} &1) \quad \text{Minimise} \quad \textstyle\sum_{j=1}^{N} C_{1j}x_j + \sum_{i=1}^{M} D_i u_i + \sum_{i=1}^{M} E_i o_i \\ &2) \quad \text{Minimise} \quad \textstyle\sum_{j=1}^{N} (C_{1j} + C_{2j})x_j + \sum_{i=1}^{M} D_i u_i + \sum_{i=1}^{M} E_i o_i \end{aligned} \tag{5.2}$$

where

C_{1j} includes wage costs and the large constant duty cost and
C_{2j} is the appropriate penalty cost.

The solution stages which follow correspond to those necessary to solve the formulation of model **(5.2)**.

Stage P1

Duties are ranked in order of 'desirability' during the generation phase. This rank has already been used in the **EVEN** program described in section **4.3.6** to remove inefficient duties whilst ensuring that each piece of work is covered by a specified number of duties. The value is an integer calculated by expression **(5.3)**. If the value of the expression is less than one then it is rounded up to the value one.

$$\lfloor \frac{\text{Maximum duty content} - \text{Actual duty content}}{12} \rfloor. \tag{5.3}$$

This is a simple model which will assess the efficiency of any particular duty based upon the amount of unproductive time spent during that duty. It should be noted that since this model does not take into account certain features, e.g. the type of duty or times of day over which the duty operates, the figure calculated here will only provide a rough guide to the efficiency of the duty.

In the first stage of the solution all three-part duties and duties with a rank above four are temporarily excluded from the duty set. The integrality constraints on the duties are relaxed so that a piece of work may be covered by fractions of several different duties. The model is then solved using customised Mathematical Programming software with the remaining duties.

Stage P2

The previously excluded duties are restored and the LP relaxation is reoptimised with the complete set of duties.

Stage P3

Having found a solution with the minimum number of duties, the penalty costs are added to the costs of all appropriate duties. To ensure that this action does not affect the solution with regard to the two main priorities of the bus company as stated in section **4.2.1**, the following two side constraints are now added :

$$\begin{aligned} \textstyle\sum_{i=1}^{M} u_i &\leq 0 \\ \textstyle\sum_{j=1}^{N} x_j &\leq T. \end{aligned} \tag{5.4}$$

The first ensures that there can be no uncovered work in the solution, which was the most important consideration. The second constraint uses a target number of duties to ensure that the minimisation of the number of duties in the solution is still addressed. If the minimum number of duties is integral then T will take this value. However, it is more likely that the minimum number of duties is fractional and since an integer solution will be the final requirement T will take this number rounded up to the next integer. Any further solutions therefore need at most T duties. This new model is solved with the integrality constraints still relaxed.

Stage P4

If the total number of duties in the current solution is non-integral and of the form $I.f$, the side constraint

$$\sum_{j=1}^{N} x_j \quad \geq I + 1 \tag{5.5}$$

is added. This is because the current solution is minimal for a relaxed LP model, and so any integer solution must require at least the next highest integer number of duties. The reason for adding this constraint is to act alongside the constraint added in **Stage P3**, so that the resulting number of duties must be integral. This tight bound helps limit the search in the final stage of the solution and reduces execution times. The current solution however will be necessarily infeasible for this new constraint and the new model must be re-solved.

If the total number of duties in the current solution is already integral then no side constraint is added and no reoptimisation performed.

Stage P5

This stage will only be necessary for an optimal schedule which contains fractional duties. In this case a branch and bound method is used to determine which duties will appear in the final schedule. The customised branch and bound process is described in detail in section **5.4.3**.

5.3 An Alternative Solution Strategy

Sherali [76] noted that there are several disadvantages in using a sequential approach to find a solution which satisfies two objectives. Apart from the fact that two separate linear programs have to be solved which are essentially doing the same task but with a different cost coefficient, a high degree of degeneracy is likely to occur with typically large numbers of iterations. Also the introduction of side constraints to maintain the subjective ordering of the objectives results in a more complex model needing to be solved which may increase execution times over a simple model for which efficient solution codes exist.

An alternative approach to solving the driver scheduling model has been explored by Willers et al. [77] and Willers [14]. The approach used is that of merging the two objectives using

equivalent weights to create a single objective, simultaneously reflecting the importance of each original objective. This would certainly be a more efficient way of solving the LP relaxation, provided that the processing time of the new model does not outweigh the benefits of reducing the number of solution stages.

In order to simplify the two objectives it was noted by Willers [14] that a side constraint is added to the model to prohibit any undercover. He deemed it unnecessary for the u_i variables to appear in the objective function. Willers [14] also states that overcover variables having non-zero objective function coefficients may cause the minimisation of wage costs not to be achieved. Overcover is unproductive and is discouraged by the minimisation of wage costs as it would suggest that more than one driver is being paid to cover an identical piece of work and so zero costs are therefore attached to the o_i variables.

It was also proposed by Willers [14] that since the main objective is to ensure that the number of duties in the solution is minimised, the cost coefficient can be removed until **stage P3**. At this point both a wage cost and any penalty costs will be attached to each duty variable and the new problem will be reoptimised. The large constant used to prioritise the duty minimisation process is no longer required.

The two objective functions are now:

$$\begin{aligned} &1) \quad \text{Minimise} \quad \textstyle\sum_{j=1}^{N} x_j \\ &2) \quad \text{Minimise} \quad \textstyle\sum_{j=1}^{N} C_j x_j \end{aligned} \tag{5.6}$$

where

C_j combines wage and penalty costs for duty j.

5.3.1 Sherali Strategy for a Single Objective Model

Methods have been proposed by Sherali [76] to convert certain multi-objective models into single objective models. The situation created by TRACS II where the model has two objectives with integral objective function coefficients was addressed by Willers [14]. The method involves weighting the objectives and combining them. For the TRACS II problem with two objective functions, the single objective formulation is as follows:

$$\text{Minimise} \quad W_1 \sum_{j=1}^{N} x_j + W_2 \sum_{j=1}^{N} C_j x_j . \tag{5.7}$$

W_1 and W_2 must have appropriate values to ensure that the objectives are correctly ordered. Sherali proposed two methods of calculating these values, of which the second method was recommended as it generally produces lower weights. Willers [14] has experimented further with the values assigned to the two weights but the Sherali method described has remained throughout the column generation implementation which is described in this thesis. This produces:

$$\begin{aligned} W_1 &= 1 + UB[\textstyle\sum_{j=1}^{N} C_j x_j] - LB[\sum_{j=1}^{N} C_j x_j] \\ W_2 &= 1 \end{aligned} \tag{5.8}$$

where

$$\begin{aligned} &UB[\textstyle\sum_{j=1}^{N} C_j x_j] \text{ is an upper bound value on the duty costs.} \\ &LB[\textstyle\sum_{j=1}^{N} C_j x_j] \text{ is a lower bound value on the duty costs.} \end{aligned} \tag{5.9}$$

As the model only takes values between 0 and 1 at the LP relaxation stage, a lower bound can simply be evaluated as 0 and an upper bound can be calculated by setting all duty values to 1. This upper bound however corresponds to summing all duty costs for the N duties generated, which would produce a very large weight. A smaller weight can be calculated by adapting the upper bound so that :

$$W_1 = 1 + \text{sum of } X \text{ largest } C_j \text{ values} \tag{5.10}$$

where X must be an upper bound on the number of duties in the schedule. Willers [14] has shown that, for the TRACS II model, an appropriate weight can be ascertained by defining X to be the number of duties in the initial solution since the sum of the highest X cost values in this case must be an upper bound to any further schedule costs calculated.

Given that there are no longer two objectives to optimise, **stage P3** in the original strategy, which added the penalty costs, is not needed.

5.3.2 The New Model

By defining :

$$D_j = W_1 + C_j \tag{5.11}$$

as the new cost coefficient for every duty variable, the new model can be defined as follows :

$$\begin{aligned}
&\text{Minimise} && \textstyle\sum_{j=1}^{N} D_j x_j \\
&\text{Subject to} && \textstyle\sum_{j=1}^{N} A_{ij} x_j = 1 \quad \text{for } i = 1, .., L \\
& && \textstyle\sum_{j=1}^{N} A_{ij} x_j \geq 1 \quad \text{for } i = L+1, .., M \\
& && x_j = 0 \text{ or } 1, \qquad \text{for } j = 1, .., N.
\end{aligned} \tag{5.12}$$

5.3.3 New Solution Strategy

The stages necessary to solve the Sherali model are as follows :

Stage S1

Ban inefficient duties by using the same method as with the original model, and solve the LP relaxation of the new model.

Stage S2

Restore the banned variables and solve the LP relaxation of the new model over the whole duty set.

Stage S3

If the sum of the duty variables is integral at the Branch and Bound phase then a tighter bound is introduced and the execution times reduced. Hence it will often be necessary to introduce a side constraint which increases the current optimal duty total to the next integer.

$$\sum_{j=1}^{N} x_j \geq I + 1 \tag{5.13}$$

where the current optimal number of duties is $I.f$. The LP relaxation is then resolved with the added constraint.

Stage S4

Find an integer solution using the branch and bound technique contained in the original model.

5.4 Running ZIP

The optimisation process is based on the ZIP package [34] which consists of a suite of Fortran subroutines incorporating the necessary major processes involved in the solution of set covering and set partitioning problems. There are core routines which contain the basic structure of the standard algorithms, and these call a set of Harwell Library subroutines [78] which have been developed to handle sparse linear programming bases. There is also a set of user routines which control the route through the program in order to reflect the characteristics of the driver scheduling problem. The package has been customised for use within the TRACS II system, and Willers [14] has adapted it to solve the Sherali model.

Firstly the relaxed model is solved using the Revised Simplex Method, where the minimum cost schedule is built up from many duties but each duty's contribution may be fractional. An integer solution is then formed, based on the relaxed solution, by means of a branch and bound approach which chooses whether a particular duty should contribute wholly to the schedule or not at all.

The Revised Simplex Method improves any current basic feasible solution by attaching a price to the rows of a matrix. A column then has a reduced cost based on these prices, and any column which has a negative price will potentially improve the objective value. In order to initialise this procedure it is necessary to create a basic feasible solution to the problem.

5.4.1 The Initial Solution

It would be possible to create an initial solution by selecting the slack variable corresponding to every workpiece constraint. This equates to having every piece of work uncovered, which is feasible unless there are side constraints preventing this. However, since the quality of any initial solution affects the number of subsequent iterations, it is important to use intelligent heuristics in order to produce a reasonable solution whilst not compromising the overall process time. The heuristic originally implemented in TRACS II considers each piece of currently uncovered work in ascending order of the number of duties available to cover it, and selects a duty to be included in the initial solution which minimises the nominal cost function **(5.14)** without causing any workpiece constraint or upper-limit side constraint to be violated.

$$\text{Min} \quad (\tfrac{C_j}{NU_j}) + OC_j \tag{5.14}$$

where

C_j = Duty cost of duty x_j
NU_j = number of currently uncovered workpieces covered by x_j
OC_j = increase in overcover costs caused by selecting x_j.

In order to take into account the overcover variables no longer being costed and the duty costs now incorporating Sherali weights, a new function can be defined as :

$$\text{Min} \quad \tfrac{D_j}{NU_j} \tag{5.15}$$

where

D_j = Sherali cost of duty x_j
NU_j = number of currently uncovered workpieces covered by x_j.

This procedure continues choosing uncovered workpieces in increasing order of the number of available duties covering them until all have been considered. For an initial solution in which any duty is wholly contained within a number of other duties, this duty is removed to create a cheaper solution. The remaining duty variables form part of the starting basis.

For some side constraints, e.g. an upper limit on a particular duty type or duty total, the feasibility of a solution can be checked during the procedure. However, lower limits on duty types may be difficult to satisfy in an initial solution and if it is the case that the initial solution is infeasible, the corresponding constraints will be temporarily removed and applied at a later stage.

It is likely that a reasonable lower limit on the total number of duties will automatically be satisfied due to the inefficiency of any starting solution, but to satisfy a realistic upper limit it is possible that some workpieces will remain uncovered. For this reason the 'no uncovered work' constraint is not applied initially.

To complete the basis, for all workpiece constraints that are uncovered the corresponding undercover variable is set to one, and for all workpieces which are overcovered the corresponding overcover variable is assigned the appropriate value. For each side constraint the corresponding slack or surplus variable is assigned the appropriate value so that the constraint is satisfied. There is also a process for determining zero-level basic variables.

5.4.2 Solution of Relaxed LP

At any stage in the solution procedure the Primal Simplex Algorithm is used to improve any basic feasible solution, and the Dual Simplex Algorithm is initiated by a primal infeasible basic solution.

The Primal Simplex Algorithm

ZIP uses the Primal Simplex Algorithm to exchange a non-basic variable with a favourable reduced cost for a duty already in the basis. The resulting solution will be basic and feasible and possibly with an improved objective cost.

The standard method of selection attempts to perform the greatest improvement in objective cost for a unit change in the entering variable. This is equivalent to choosing the entering variable with the most negative reduced cost. In terms of the convex polytope however this only considers the edges in the space of the current non-basic variables, whereas Harris [79] suggested that it would be more beneficial to establish the steepest edge over all variables. Smith [15] incorporated a method developed by Goldfarb and Reid [80] into ZIP which calculates the edge gradients using recurrence relations. Due to the number of columns in a typical driver scheduling model, calculating the gradient of every non-basic variable for each steepest edge iteration takes much longer to perform than in a standard Primal Simplex iteration. However, results have shown that the total number of iterations required was reduced to the extent that

the solution times were improved.

The Dual Simplex Algorithm

Given an infeasible solution to a linear programming problem in which all reduced costs are positive, a feasible solution can be obtained by exchanging basic variables for non-basic variables in such a way that the solution remains basic and the reduced costs remain positive. The method used is the Dual Simplex Algorithm and is incorporated into the ZIP package in order to solve solutions which are currently infeasible, such as when a lower bound is placed on the target number of duties in **stage S3**.

5.4.3 Branch and Bound

Assuming that, by relaxing the integrality conditions, the resulting LP solution produced a schedule with fractional numbers of duties, an integer solution must be found.

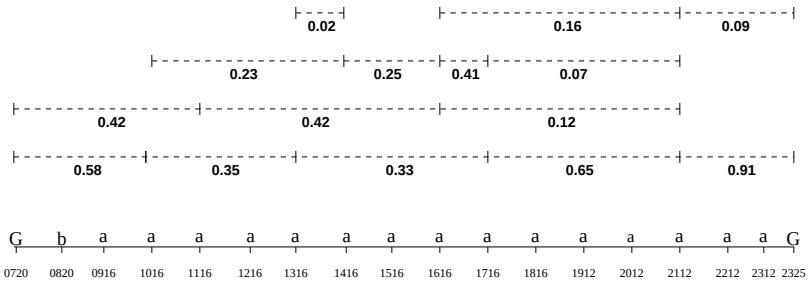

Figure 5.1: Example of Fractional Coverage of a Bus

Figure 5.1 shows how the work of one bus may be covered by fractional duties. For every part duty displayed there will be be one or two more parts covering pieces of work on other buses at the same level so that the complete bus schedule is covered by fractions of valid duties. Note that in this example there is an example of overcover between 1616 and 1716, where the sum of duty variables covering this piece of work totals 1.02, whereas all other pieces of work shown are covered by a total of one duty.

Heuristics, used in the earlier parts of TRACS II have removed the guarantee of finding the optimal schedule over all possible valid duties. Hence, since the optimal integer solution to the

current problem will not necessarily equate to the optimal schedule, and users can seldom specify what they mean by optimality, the emphasis is on finding a good integer solution quickly. The integer solution is found by developing a branch and bound tree, where the lower bound on the objective cost is given by the optimal continuous solution. At this stage the Sherali weights could be removed, but their inclusion in the branch and bound tree should have no adverse effect on the execution times and so they remain part of the cost coefficient of the duty in the objective function.

Once an integer solution has been found the nodes of the tree are fathomed if their cost is greater than or equal to a scheduler specified percentage of the current best integer cost. Willers [14] explored the possibility of increasing the percentage to take into account the higher costs with Sherali weights added but the implemented percentage of 99.5 has been retained throughout the current research. In most cases the first integer solution found fathoms all of the remaining active nodes and hence the branch and bound process normally terminates with a possibly non-optimal integer solution.

Smith [15] developed a technique to reduce the size of the duty set entering the branch and bound phase, which assumes that an acceptable integer solution can be found by restricting the choice of relief opportunities to the subset used in the LP optimum. In this way any duty which does not use any of these times can be excluded from the search. The REDUCE procedure uses the principle that the LP optimum gives a good indication of how the bus work would be covered in an integer solution, and extensive practical application has adequately produced an integer solution in a much reduced time for the majority of data sets. It is possible that the constraint limiting the search to find a schedule with an exact number of duties may not be satisfied with the limited set of relief opportunities. However, in general there are many different integer solutions, some of which should be contained within the duty set available. The other possibility is that the reduced duty set produces a worse integer solution than one with the whole set, but since the process terminates with the first good solution its quality is merely dependent on the choice of path through the tree, not on the number of duties which are considered.

Branching strategy

Various different branching strategies were considered during the development of TRACS II. Initially a constraint branching strategy developed by Ryan and Foster [34, 81] was implemented but was superseded by a relief time branching strategy.

The method considers the optimal LP solution containing fractions of duty variables covering pieces of work. For each relief time its *changeover* is defined as the sum of the duty variables which finish (or start) a spell at that time. For example, in Figure 5.1 the changeover for relief time 1316 is 0.35. Since a feasible solution requires that the changeover for each relief time should have the value 0 or 1 (except where there is overcover), branching on a selected relief time forces the changeover to take one of these values. For the zero branch, all duties which start or end a spell at that relief time must be banned and for the one branch, all duties working through that relief opportunity must be banned.

The relief times are sorted into chronological order and preference is given to branching on relief points alternately from the beginning and end of the day. At any point the next node to be selected is based upon the evaluation of a function. The calculation involves an estimate of the objective value which would result if the node were solved and the sum of the integer infeasibilities at the parent node. For this reason the first branch to be evaluated is always that which forces the fractional changeover to whichever is the nearer of 0 and 1.

It is possible that a schedule is still fractional with all changeover values integral and in this case TRACS II initiates a constraint branching strategy of the type originally implemented and developed by Ryan and Foster [34, 81]. This considers pairs of pieces of work, where one branch forces both pieces of work to be covered by a single duty and the other branch forces different duties to cover them. For relief times which have overcover on either piece of work adjacent to it, the relief point is not used to form a branch. If no constraint branch can be found then a traditional variable branching strategy is used.

Solutions

At each node in the tree a certain amount of information must be held about the current solution including the basis matrix. For this reason a limit was placed on the size of the tree.

With improved technology the current maximum number of nodes which can be created has been increased to 500.

It is generally the case that if a solution cannot be found within a much smaller number of nodes then for some reason an integer solution is being prohibited. If the branch and bound phase fails to find an integer solution within the node limit then this may indicate that although an integer solution exists, its objective cost is much higher than that of the relaxed LP solution and hence the branching strategy used has difficulty in locating it. Another failure to find an integer solution is when all the nodes of the tree have been fathomed through infeasibility. In this case a solution cannot be found satisfying the specified number of duties with the duty set available and either the target number of duties may have to be increased by one, or a larger set of duties entered into the branch and bound to allow more choice.

5.5 Overview

The TRACS II driver scheduling system firstly relaxes a set covering model to find the optimal continuous solution, and then uses this to specify a target number of duties to search for in an integer schedule which uses a branch and bound search. TRACS II solves two objective functions in order to prioritise the requirements of a driver scheduling system, but it has been reported that time savings can be made by incorporating only one objective function with a Sherali weight to retain preference. Optimality is still limited to the quality of the previously generated duty set and the following chapter considers a column generation approach which will allow more duties to be considered.

Chapter 6

A Column Generation Model

6.1 Introduction

The major disadvantage of using most computerised scheduling systems, including TRACS II as described in the previous chapters, is that the search for optimality is limited to an heuristically reduced duty set and so for some problems the ILP fails to find a feasible solution. Large problems can be decomposed but this is usually deemed undesirable by users. Heuristics often limit the formation of duties to include only efficient duties, but since it may be the case that some inefficient duties may be required to link with them to produce the optimal solution, ideally we would wish to consider allowing all possible valid duties which could be formed to enter the set covering model. This chapter aims to introduce an alternative method which allows more duties to be available whilst using a much smaller set to solve the model.

6.2 Related Problems

Scheduling problems are a subset of problems which contain many variables. Section **2.6** describes some of the situations in which column generation has been successfully applied to problems that would otherwise have to be reduced or decomposed before being solved using a set covering or set partitioning formulation.

HASTUS [19, 20, 21, 22, 24] also contains a software module Crew-Opt [59, 60] which uses column generation techniques and has been shown to be successful in producing optimal or near-optimal driver schedules for small transit operations. Hamer and Séguin [24] report that

Crew-Opt can produce quasi-optimal solutions to problems with less than 50 drivers, and is useful for bus driver scheduling in which drivers are allocated route-by-route.

6.3 Column Generation

The constraints of a linear programming problem can be written as a set of m simultaneous linear equations in n unknowns,

$$\mathbf{A}\underline{x} = \underline{b} \tag{6.1}$$

where

A represents an m x n matrix,

$\underline{x}$ represents the variables $x_1, .., x_n$,

$\underline{b}$ represents the m constraint values.

Column generation is an approach used to solve mathematical programming problems which contain many columns; i.e. n is very large. The method uses the Revised Simplex Method to solve the problem over a subset of the columns. The process then repeatedly adds further columns to the subset which will potentially improve the solution and re-solves over the new set until optimality has been reached.

6.3.1 Theory of the Simplex Method

From Proll [82], given a set of constraints

$$\begin{aligned} \mathbf{A}\underline{x} &= \underline{b} \\ \underline{x} &\geq 0 \end{aligned} \tag{6.2}$$

the variables can be split into those which are included in a basic feasible solution and those which are not. Given that **A** is an mxn matrix we can denote

$$\begin{aligned} x_i^{\mathbf{B}} \quad & \text{for } i = 1, .., m \\ x_i^{\mathbf{N}} \quad & \text{for } i = m+1, .., n \end{aligned} \tag{6.3}$$

as the basic and non-basic variables respectively.

An mxm matrix $\mathbf{B}$ can be formed from m linearly independent columns of $\mathbf{A}$ such that equations **(6.2)** can be rewritten as :

$$\begin{aligned} & \mathbf{B}\underline{x}^{\mathbf{B}} + \mathbf{N}\underline{x}^{\mathbf{N}} = \underline{b} \\ & \\ & \underline{x}^{\mathbf{B}} \geq 0 \\ & \underline{x}^{\mathbf{N}} \geq 0. \end{aligned} \tag{6.4}$$

The matrix $\mathbf{B}$ is called the *basis matrix*.

Rewriting the first constraint of **(6.4)** in terms of $\underline{x}^{\mathbf{B}}$ gives :

$$\underline{x}^{\mathbf{B}} + \mathbf{B}^{-1}\mathbf{N}\underline{x}^{\mathbf{N}} = \mathbf{B}^{-1}\underline{b}.$$

or

$$\underline{x}^{\mathbf{B}} = \mathbf{B}^{-1}\underline{b} - \mathbf{B}^{-1}\mathbf{N}\underline{x}^{\mathbf{N}} \tag{6.5}$$

The Revised Simplex Method uses equation **(6.5)** to generate any item in the simplex tableau. For any simplex tableau which is non-optimal it is necessary to find currently non-basic variables which, when swapped with current basic variables, will improve the objective value.

The objective function can be written in the form :

$$z = \underline{c}^T \underline{x}$$

where

$\underline{c}^T$ contains unit costs for each x_i variable,

so that

$$z = \underline{c}^{\mathbf{B}^T} \underline{x}^{\mathbf{B}} + \underline{c}^{\mathbf{N}^T} \underline{x}^{\mathbf{N}}. \tag{6.6}$$

Since all non-basic variables will have the value zero, the objective function can be rewritten in terms of only the $\underline{x}^{\mathbf{B}}$ variables. Substituting **(6.5)** into **(6.6)** gives :

$$z = \underline{c}^{\mathbf{B}^T} \mathbf{B}^{-1} \underline{b} - \underline{c}^{\mathbf{B}^T} \mathbf{B}^{-1} \mathbf{N} \underline{x}^{\mathbf{N}} + \underline{c}^{\mathbf{N}^T} \underline{x}^{\mathbf{N}}$$

or

$$z + (\underline{c}^{\mathbf{B}^T} \mathbf{B}^{-1} \mathbf{N} - \underline{c}^{\mathbf{N}^T}) \underline{x}^{\mathbf{N}} = \underline{c}^{\mathbf{B}^T} \mathbf{B}^{-1} \underline{b}. \tag{6.7}$$

By now defining :

$$\underline{\pi} = \underline{c}^{\mathbf{B}^T} \mathbf{B}^{-1} \tag{6.8}$$

where π is known as the *simplex multiplier* or *pricing vector*, the objective function can be calculated as :

$$z + (\underline{\pi} \mathbf{N} - \underline{c}^{\mathbf{N}^T}) \underline{x}^{\mathbf{N}} = \underline{\pi} . \underline{b}. \tag{6.9}$$

The inclusion of a non basic variable will alter the objective cost based upon the expression :

$$\underline{\pi}\mathbf{N} - \underline{c}^{\mathbf{N}^{\mathbf{T}}} \tag{6.10}$$

So, for any non-basic variable k its *reduced cost* can be calculated as

$$\underline{\pi}k^T - c_k \tag{6.11}$$

i.e. π * original data column for variable - coefficient of variable in objective function.

Since we wish to know the alteration in the objective value z that would take place if any currently non-basic variable should be entered into the basis, then if the problem is one of maximisation, the reduced cost of a variable should be negative for z to increase. Similarly for minimisation problems the reduced cost should be positive for any objective value improvement.

The variable with the most negative (or positive) reduced cost is normally the one chosen to be the entrant variable since this will give the best objective cost improvement for a unit change in any one non-basic variable. Other methods are available, e.g. Harris [79] which also consider the changes in the basic variables for a unit change in a non-basic variable. The leaving variable is chosen by means of a ratio test.

6.3.2 Method of Solution

Assuming that all columns are available in the matrix of constraints **A**, the Revised Simplex Method begins with an initial solution and improves the objective value by swapping basic variables for non-basic variables with favourable reduced costs until no further improvement can be made.

In the case of column generation only a subset of the columns is available at the outset. The Revised Simplex Method is used to find the solution which is optimal over the subset and then generates or searches through columns not previously considered. If there are any new columns which have favourable reduced costs then some, or all, of them are added to the subset

as non-basic variables so that the current solution is no longer optimal. Using the current basic solution, the Revised Simplex Method continues to swap columns where necessary to reoptimise over the current larger subset. For any LP relaxation which is optimal over its available subset the overall optimal solution is attained when no more columns which would improve the objective value can be added to the set.

6.4 Application of Column Generation to Driver Scheduling

Since TRACS II uses heuristics to reduce the number of duties in order to enable conventional mathematical program solvers to produce a solution, and column generation has been shown to be a method which can be used to solve driver scheduling problems which often have to be decomposed, further investigation has taken place into incorporating column generation techniques within TRACS II in an attempt to overcome some of the difficulties which arise in certain situations, and also to improve upon the solution and/or speed of the process.

6.4.1 Method of Solution

A column generation approach to solving the driver scheduling problem requires strategies for resolving the following three issues :

- How to choose an initial duty set.
- How to generate further duties.
- How to produce an integer solution from the LP solution.

The driver scheduling problem is one of cost minimisation and so by defining the reduced costs for non-basic variables as in **(6.12)**, improvements in the objective value can only be made if there are any non-basic variables with negative reduced costs.

The reduced cost of duty k is :

$$c_k - \sum_{i=1}^{m} \pi_i a_{ik} \qquad \textbf{(6.12)}$$

where

$c_k =$ cost of duty k

$\pi_i =$ simplex multiplier for row i

$a_{ik} =$ coefficient of duty k in constraint i.

6.4.2 The HASTUS Crew-Opt Method

Method of Solution

Papers have been published [59, 60, 83, 84] which describe a column generation approach to driver scheduling, the modelling of the subproblem, and the branching rule used to find the integer solution.

Crew-opt first generates an initial subset of known feasible duties, and the linear programming relaxation of the resulting set covering problem is solved, using the simplex algorithm to price the variables. Once optimality has been achieved within the duty subset a subproblem is solved either to ascertain that the solution is optimal for the complete problem, or else to introduce more duties into the subset so as to improve the solution further.

The subproblem which is solved in Crew-Opt is a constrained shortest path problem, so that further duties are generated at each stage of the column generation process by constructing duties as paths through a network. Resource constraints are defined on the network to ensure that only valid duties will be considered. The feasibility definitions are :

- The maximum duration of a spell of work.
- Minimum and maximum lengths of a break.
- The maximum number of pieces of work allowed in a duty.
- The maximum spreadover of a duty.

The first two of these duration constraints are included in the graph definition constraints, thus ensuring that arcs can only be formed if they are respected. Paths through the network will then represent duties which are built from feasible spells and breaks. The final two definitions are path feasibility constraints which affect the final duty. The quantities accumulate along the path to ensure that the final duty satisfies constraints on the time and work content included

in it. More complicated constraints can also be modelled to handle a broader range of labour agreements rules and conditions such as limits on, the duration of half day duties, the number of workday types in a schedule, and overtime and spread bonuses. Crew-Opt has also been modified to deal with an increased number of breaks. For some constraints the network is amended by adjusting the cost of the arcs in order to correctly identify the shortest path as the duty with the most negative reduced cost.

Figure 6.1 shows how a feasible two-part duty would be represented by five arcs from the source node to the sink node, with each arc representing a specific task and the intermediate nodes representing relief points.

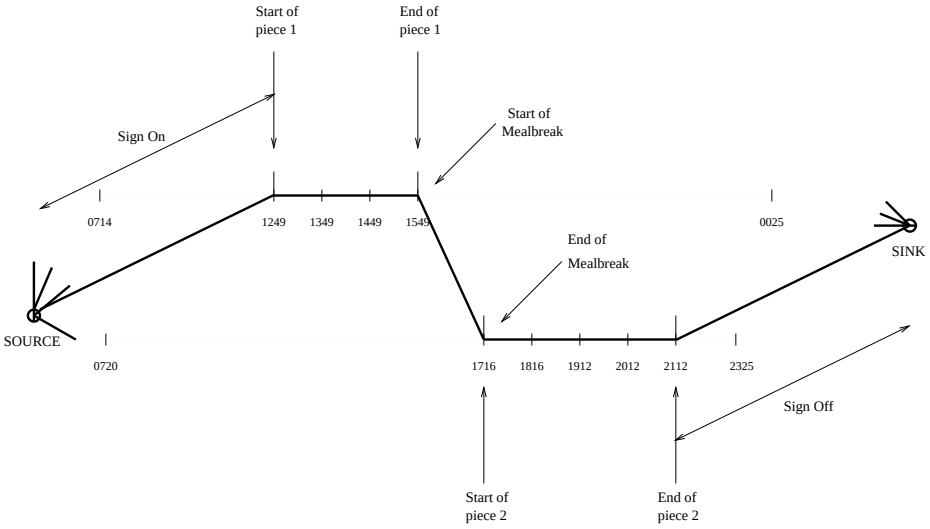

Figure 6.1: Representation of a Duty as a Path Through a Network

From **(6.12)**, finding the duty with the minimum reduced cost equates to finding the duty/duties satisfying the following :

$$\text{Minimum } c_k - \sum_{i=1}^{m} \pi_i a_{ik} \qquad \textbf{(6.13)}$$

where

$c_k =$ cost of duty k

$\pi_i =$ simplex multiplier for row i

$a_{ik} =$ coefficient of duty k in constraint i.

Assuming that the cost of the duty is calculated as :

$$\text{sign on cost} + \text{sign off cost} + (\text{spreadover} * \text{hourly wage})$$

then it is possible to calculate the reduced cost of a duty progressively. The reduced cost of a duty or part duty is the sum of the contributions to the reduced costs of its individual tasks, which include signing on and off, covering particular pieces of work, and having breaks. Since the simplex multipliers are only associated with the pieces of work contained within a duty, the contribution to the reduced costs of signing on and off and having breaks are merely their contribution to the duty cost. Hence :

- The contribution to the reduced cost of signing on and off will be given in the labour agreement.

- The contribution to the reduced cost of a break, if paid, will be *(the duration of the break * hourly wage).*

- The contribution to the reduced cost of an individual piece of work is *((the duration of the piece * hourly wage) - simplex multiplier associated with that piece of work).*

These costs can be attached to any arcs on the network, and the shortest path through the network corresponds to the duty with the smallest reduced cost. If this cost is positive then the current solution is optimal, otherwise this duty should be added to the subset and the problem reoptimised. The shortest path problem is solved using a dynamic programming algorithm and the solution of the subproblem is a constrained shortest path tree representing several duties, many of which have a negative reduced cost. Results confirmed that adding all other duties with a negative cost produced by the dynamic programming algorithm accelerated the convergence of the column generation method.

The column generation process must start with a feasible solution to the problem, and CrewOpt does this by generating an initial set of short duties which cover all the pieces of work. This solution will be costly and may violate particular given constraints, but these constraints

can be introduced later by means of penalties on the network. The solution is then optimised over the duty subset and the column generation method utilised repeatedly until the optimal LP relaxation has been found.

The integer solution is found by means of a column generation method within a branch and bound scheme. The branching strategy adopted is based upon that developed by Ryan and Foster [34, 81], and used within the ZIP programs in TRACS II [81], which considers pairs of pieces of work where one piece of work is executed immediately after the other. One branch considers the case where no duty can consecutively cover the two pieces of work and the other branch forces duties to cover both consecutively. At any node the column generation method is solved using the dynamic programming algorithm, but the network is restricted to form only the subset of duties which is relevant to the current node.

Results

Desrochers et al. published papers in 1988 [59, 84] describing the results achieved with two data sets. At this time TRACS II could solve problems with up to 600 pieces of work and produce schedules with up to 80 duties.

A) The first is an American city problem involving 25 buses and 167 pieces of work and there is a rule forbidding the formation of three-part duties. The time taken to attain the optimal LP solution was around 22 minutes, of which approximately half was spent generating 1712 new duties from 100 subproblems. The branch and bound phase then took a further 44 minutes, of which 27 minutes involved solving 207 subproblems to produce a further 1214 new duties. The branching process was altered to consider five independent pairs of variables at each level and the final schedule uses 45 duties.

B) The second problem is British and introduces the concept of half days. This problem has 20 buses and 235 pieces of work. The initial solution used was the final schedule produced by HASTUS on the same problem, and the time taken to find the optimal LP solution was 2.5 hours, solving 418 subproblems to produce 12495 new duties. The time taken to find the first integer solution was just under 3 hours and introduced 5778 more duties from 653 subproblems. The branching process considered ten pairs of tasks at each level. The final schedule uses 51 duties and is an improvement over the HASTUS solution due to its more efficient matching of half days and inclusion of more three-part duties.

Rousseau [61] describes the results obtained with Crew-Opt on four problems:

C) Crew-Opt was tested on two small French data sets which are operated on a line-by-line basis. It was not expected to improve upon the current solution and Crew-Opt produced the same number of duties satisfying all the constraints, with each line taking between 15 minutes and 3 hours of execution time on a 386 PC.

D) A Barcelona problem also builds schedules line-by-line and Crew-Opt reduces the number of drivers and the cost over a solution produced by HASTUS. On the given subset of lines the percentage of straight duties was reduced slightly so that the upper limit constraint was not satisfied, but this was not a problem since the percentage was sufficiently high on the remaining lines.

E) A number of lines of a Vienna problem were tested using Crew-Opt and a duty saving was observed. Some more complicated constraints on this problem meant that a large number of three and possibly four-part duties had to be formed.

F) The East Japan Railway problem requires one-day duties and also overnight duties to operate on over 700 suburban and inter-city trains. Break and mealbreak constraints mean that some duties are composed of up to six spells of work. HASTUS has been tested on the problem but produced some short duties which were unacceptable. In solving the problems using Crew-Opt an heuristic was used to obtain an integer solution from the LP solution. At any node the branching strategy fixed a variable to the value 1 before reoptimising with column generation. Two problems were considered. The first consisted of 77 duties with a manual solution which was improved to 65 duties using Crew-Opt. The second problem manually required 162 duties which had to be decomposed into three subsets by Crew-Opt. Once each subset was optimised the less efficient duties were reoptimised globally. The resulting schedule required 156 duties in total. The time taken to achieve these results was around 24 hours on a Sun Sparc10/31 although it is noted that an older version of Crew-Opt was used.

Analysis

Desrochers and Soumis [84] suggest a way of speeding up the optimisation process by incorporating a mixed enumeration/column generation approach which enumerates a number of good

duties used to initialise the algorithm.

Desrochers et al. [60] claim that Crew-Opt has two main advantages over existing methods of computerised solution. Firstly heuristics are not used to reduce the duty set, and secondly the LP solution produces a very good lower bound on the solution cost so that the branch and bound algorithm can be terminated at a good solution if the expected improvement does not justify the additional computation cost. The integer solution may therefore not be optimal. The method has been used successfully on smaller problems with the claim that smaller problems have fewer good alternative feasible schedules and heuristic algorithms tend to stop at worse solutions. It is expected that, with computer development, Crew-Opt will also be able to solve larger problems.

6.4.3 Proposed Column Generation Implementation within ZIP

As mentioned at the beginning of this section, in order to implement a column generation method into the TRACS II system there must be a strategy for choosing the initial duty set, a strategy for generating further duties and a strategy for producing an integer solution.

Duty Generation

Incorporating column generation techniques into the solution strategy of the set covering model potentially allows all valid duties to be considered. However, to guarantee the overall continuous optimum one of the following two methods would have to be used:

1) `All valid duties would have to be formed initially and their reduced costs calculated during the column generation process to ensure optimality.`

2) `A subproblem must be available to generate any remaining duties with negative reduced costs once optimality has been found over a duty subset.`

Crew-Opt uses the second method, with a subproblem defining a constrained shortest path technique to generate further duties as paths through a network. In the current TRACS II system the duty costing not only depends upon the duration of the duty, but may also include

a subjective weighting to deter some duties from being included in a schedule. This is approximate because many organisations seek to minimise unpopular or administratively difficult duties. The penalty costs are currently defined in a subroutine specific to each organisation and typical requirements include:

- penalties added to three-part duties
- penalties added to duties whose stretch lengths differ greatly
- penalties added to early duties which start after the earliest time for split duties.

Although these penalties can be added accumulatively as a duty is being constructed, they are not added as resource constraints and cannot be built into the network. If the shortest path through the network corresponds to a duty with undesirable features its reduced cost may not be the most negative once the penalty cost has been added to the duty cost.

Also, although Crew-Opt hopes to be able to handle larger problems, published work claims that it is restricted. Indeed, larger problems will require large networks and for each set of labour agreement rules there will be a certain amount of time required to convert them into network constraints. In order to speed up the process of column generation it is usual for methods to add more than one column at each iteration, and for larger problems it will become complex and time consuming to traverse a network to generate such duties. Decomposing a larger problem will compromise optimality.

For the reasons given above, and for further reasons explained in the next section, the technique which will be explored is that of removing the heuristics which reduced the size of the generated duty set in TRACS II, hence allowing more duties to be considered by the set covering model. A network formulation will not be used, and so duties from this larger set will be selected to enter a working subset by means of a less sophisticated enumeration method. An improvement in continuous LP optimum may be achieved over one using a smaller duty set and so for each bus company both a larger duty set and an heuristically reduced duty set will be produced. The current version of TRACS II limits the number of duties which it can accept to 22,000; in order to compare results with a column generation technique on larger duty sets this value has

been increased to 100,000. Assuming that the current system produces an LP solution then the column generation method on the same data set will not improve the cost as both will be optimal. However, the column generation method may prove to be faster and the larger data set may produce a lower LP cost than the smaller data set.

Integer Solution

If the overall continuous optimum solution has been found then in order to guarantee an optimal integer solution it would be inadequate to terminate the branch and bound phase as soon as a good solution had been found. Crew-Opt uses a column generation technique within a branch and bound structure in order to ensure that the integer solution is not limited to the duties formed in finding the optimal LP solution. Continuing through the tree in this way would guarantee an overall optimal integer schedule, although in practice Crew-Opt pauses at the first integer solution and uses the continuous optimum to analyse whether the expected improvement in integer solution cost justifies the extra computational cost of further searching. At each node the column generation method used in the branch and bound phase limits the duties which are formed to those relevant. Results given for Crew-Opt showed that a large number of new columns are added overall in the branch and bound phase even though only one path through the tree will provide the integer schedule.

In TRACS II, since the duty costs include subjective weightings, many integer solutions may be equally or possibly more preferable to the optimal schedule with respect to duty content, and so spending time searching for the optimal schedule may be wasted. Whilst the solution time of computerised systems is acceptable compared to previous manual methods, bus companies would tend to give execution time reduction a higher priority than improving a good integer solution. Faster solutions would allow a user more opportunity to analyse the effect of relaxing certain conditions or altering parameters.

Since the issue of optimality over all possible duties at the LP stage cannot be guaranteed using the proposed method, in order to achieve preliminary results the branch and bound phase will not be altered.

Initial duty set

Crew-Opt uses an initial set consisting of short duties in order to provide a feasible solution quickly. As discussed above, it is intended in the current research that a large set of potential duties be generated initially and a subset of these be used in such a way as to provide an initial subset of good duties, and hence a good feasible initial solution.

Solution Method

There are many possible ways of implementing a column generation strategy, but the method of solution which will be adopted is outlined as follows:

- **Step 0** Generate a duty superset.
- **Step 1** Create an initial solution and form an initial duty subset.
- **Step 2** Solve the LP over the current set.
- **Step 3** Add a new set of duties which will improve the solution.
- **Step 4** If no duties are added then the LP solution is optimal, otherwise go to **Step 2**.
- **Step 5** Find an integer solution using branch and bound.

The following chapter will discuss the implementation of this column generation strategy within the ZIP programs in TRACS II, leading to a comparison of results on a number of sample problems between an improved version of TRACS II and a column generation method.

Chapter 7

Implementation of Column Generation Within ZIP

7.1 Introduction

This chapter describes and analyses various strategies which have been tested in order to incorporate a column generation technique within the mathematical programming component of TRACS II. The aims of the column generation system are to derive better driver schedules, by allowing more duties to be available for any data set, and to produce results more quickly than TRACS II on the same problem. Results are given for seven problem instances, all of which have two sizes of duty set in order to determine whether any improvement occurred in a schedule which was built from a larger duty set. The smaller duty set is that which would typically be run through TRACS II. The larger set has been formed by omitting the processes which heuristically reduce the size of the generated duty set, and it contains all of the duties which are in the smaller duty set.

The first part of this chapter describes how the mathematical programming component of TRACS II has been altered in order to provide a realistic comparison with a column generation system. Throughout the development of each column generation strategy, its strengths and weaknesses were analysed through tests made on one of the small data sets. This will be described in the second part of this chapter. The results of a column generation system arising from apparently successful strategies are then given for all available data sets.

7.2 A Standard Model/Sherali TRACS II

In order to analyse the timings and results produced by a column generation model it is necessary to define a comparative model which will be known as the 'Sherali TRACS II' model. In order for the comparison to be realistic this model should implement identical routines whenever possible and appropriate. The following sections describe alterations in the ZIP routines, as described in Chapter 5, which have been made to derive a standard model.

Sherali Objective Function

In order to implement a column generation strategy within the ZIP process it is necessary to solve the relaxed mathematical model many times; this would be very time consuming if both pre-emptive objectives inherent in the TRACS II system needed to be optimised. Also, Willers [14] reported that by using a Sherali weighted objective function a significant reduction in execution times was observed over those produced by the multi-objective system. For these reasons the Sherali weighted objective function as described in section **5.3** is used as a basis for both the standard model and the column generation model.

No exclusion of duties

The Sherali strategy proposed in section **5.3.1** defines the first stage to be the exclusion of three part duties and duties with a rank greater than four, and the model is solved over the remaining duties. These duties are restored in the second stage and the model reoptimised. Since the purpose of a column generation strategy is to build up duties onto a relatively small subset, the elimination stage may prove trivial if used initially. Alternatively the elimination stage could be carried out on the complete duty set before a subset is chosen, but since many more duties are available to the model the process will take more time, and the larger resulting duty set will take longer to optimise. The elimination stage was omitted by Willers [14] in experimentation to the set covering system and so is removed from the standard model and not introduced in the column generation model.

Addition of Penalty Costs

Currently the version of TRACS II without the Sherali weighted objective function solves two objective functions, as described in section **5.2**. The second objective function requires penalty costs to be added to all duties, and this is done by repeatedly calling a routine which calculates the penalty cost for a particular duty. In a Sherali version of ZIP the penalty costs are already

included in the objective costs and so only need to be calculated at the beginning of the process. Since the column generation technique proposed generates all duties first, the penalty costs can be calculated initially rather than each time a new duty is added to the duty subset. The data file therefore includes a separate column which is the duty cost incorporating any penalty costs. This also has the advantage that both the standard method and the column generation method need no interaction with problem specific code once the bus and duty information has been read in.

Storage of Duty Information

It is necessary to access duty information for many reasons at the start of the ZIP process. Currently three separate subroutines read the bus and duty information, analyse the duties to decide which constraints should be equalities, and construct the initial solution by searching through the whole duty set for every piece of work which needs to be covered. Since a column generation system will consider many more duties it is important to make the storage and access of duties as efficient as possible. In order to improve the earlier processes, as the duties are now read in and stored they are analysed to determine which constraints will be equality constraints. The three data accessing subroutines are replaced with two and this speeds up the method by around 6%.

7.2.1 Sherali TRACS II Method

- **Step 1** Read in the ZIP parameter file which includes information as to how the process should be run and any user-defined side constraints.
- **Step 2** Read in and store the bus information file, which includes details of the number of generated duties which start or end at each relief opportunity. In this way redundant constraints can be eliminated. These correspond to pieces of work which are never used to change drivers.
- **Step 3** Read in and store the duties whilst setting up the equality constraints by looking for buses whose first pieces are only covered by duties of which they are the first piece, and buses whose last pieces are only covered by duties of which they are the last piece.
- **Step 4** Create an initial schedule by looking through the pieces of work in increasing order of the number of duties covering them. For a piece of work that is not yet covered, all duties which do cover it are considered. The duty which does not violate a constraint and minimises the nominal cost function **(5.15)** is chosen to be included in the initial

solution. It is possible that such a duty cannot be found and so uncovered work is allowed in the initial solution. Uncovered work is then banned before optimisation over the duty set so that further solutions will not contain pieces of work which are uncovered.

- **Step 5** Initialise x to be the number of duties in the initial solution. For every piece of work which remains uncovered in the initial solution, increase the value of x by one. Calculate the Sherali weight by summing the costs of the x most expensive duties, and add this weight to the cost of all duties.
- **Step 6** Solve the LP relaxation of the Sherali model over the whole duty set.
- **Step 7** If the LP solution is integral then the process stops, otherwise reduce the duty set to contain only those which use relief points appearing in the continuous solution, using the REDUCE process described in section **5.4.3**.
- **Step 8** If the optimal duty total is currently fractional, add a side constraint which rounds up the duty total to provide a target number of duties in the integer solution.
- **Step 9** Find an integer solution by branch and bound using the strategies described in section **5.4.3**.

7.2.2 Experimental Data Sets

There are seven data sets available to be tested throughout the period of the research, and each has two sizes of problem. The smaller duty set is the one which has been heuristically reduced so that it can be run through the original TRACS II system, and the larger duty set is not reduced and contains the smaller set so as to ensure that any improvement in the solution is not just caused by a different set of duties being available.

The duty sets arise from real life problems and have been acquired because they cause particular difficulties in obtaining solutions. Although a selection of these problems have been tested throughout the development of a column generation technique, the results of only one are reported as indicative of the behaviour of them all.

Results of GMB(S) Data using the Sherali TRACS II Model

The GMB(S) data set is the smallest available problem and can be solved easily using the original TRACS II system. Since results can be obtained more quickly it is best used during the

development of the column generation method to ensure that results produced are comparable both in terms of solution quality and execution time. For the purpose of such a comparison the results of using a Sherali TRACS II system on the GMB(S) data set is presented here.

No. duties	4199
No. of pieces of work	127
No. of equality constraints	24
No side constraints	
INITIAL SOLUTION	40 duties
Sherali Weight	94989
Initial Duty Cost (with penalties)	26991
LP SOLUTION	33.50 duties (3 overcovered pieces)
Iterations	377
Time Taken	38.9s
No. of duties after reduction	721
INTEGER SOLUTION	34 duties (4 overcovered pieces)
Nodes Created	2
Integer Duty Cost (with penalties)	27488
Total Time Taken	52.9s

7.3 Column Generation Solution Procedure

7.3.1 Implementation Considerations

It is necessary to alter certain aspects of the version of ZIP used within the TRACS II system before the column generation principles can be applied.

Linked List Data Structure

After each iteration of the column generation method more duties which will improve the objective cost are added to a duty subset. Since it would be very time consuming to calculate the reduced costs of all the remaining duties in the complete set, only duties covering a particular pieces of work are considered at any one time. For this reason it is preferable to store duty identifiers in linked lists according to the pieces of work that they cover. A similar method has already been coded in the C programming language as part of the program to exhaustively search for an optimal schedule (see section **3.3**), and has been incorporated into the column generation system. Duty information held within the duty file is also read into memory as in Sherali TRACS II so that information regarding a new duty entering the subset can easily be accessed. The linked list structure would also be beneficial to the current method of finding an initial solution as it avoids the need for searching through all duties when choosing a duty to cover a particular piece of work; it is not implemented in the Sherali TRACS II system as its lack of other benefits makes it time- and memory-consuming.

Matrix Inversion

The matrix manipulation routines [34] which perform the necessary calculations for the Revised Simplex Method include one to invert the basis matrix. Three of the uses of this include the initial matrix formation, an inversion whenever required according to an inverting frequency, and when the relaxed model has been optimised. Since the column generation system will optimise the relaxed model on more than one occasion, the initial inversion and setting up is not necessary for subsequent duty set increases, and calls to these subroutines are removed. Similar alterations have also been made relating to unnecessary intermediate calculations. At this point it is worth mentioning that the ZIP routines used to solve the Revised Simplex Method may not be the most appropriate in a column generation model and more recent advanced implementations of simplex methods, as reported by Nemhauser [85], are now available. The current routines have been retained because, both the Sherali TRACS II method and a column generation method would need to undergo major alterations in order to incorporate any changes or upgrades in optimisation software.

7.3.2 Implementation Strategies and Results

As described earlier, there are three considerations when implementing a column generation model to solve mathematical programming problems; these are the formation of the initial set of variables from which to derive an initial solution, the method of generating new variables, and the strategy used to produce an integer solution. The branch and bound strategy remains unchanged throughout the implementation that is described in this chapter and so the issue of the integer solution is not addressed here. Since the other two methods interact in order to optimise the relaxed model, and different approaches have been tested at various times, it is inappropriate to describe the implementations in chronological order. Instead, the development of the initial solutions are described first, and then analysis of different duty addition techniques is reported with an indication of which initial solution method is adopted.

All analysis focuses on the results of execution on the GMB(S) data set which are indicative of the behaviour of the method on other data sets. A Sherali TRACS II comparison can be found for GMB(S) in section **7.2.2**. Earlier experiments were performed on a SUN sparc 1 and then further testing was carred out on a Silicon graphics (Iris Indigo Workstation with 33MHz R3000 MIPS Processor) which also provided the results for the Sherali TRACS II model. Gradual time savings were also made throughout the successive experiments by incorporating the improvements described earlier in this section, and for these reasons the timings provided are merely representative during this section.

Initial Solution and Initial Duty Set

Strategy IS1

The simplest method of selecting an initial subset of duties is to use a parameter which defines a subset size p such that only the first p duties contained within the duty superset are considered. The initial solution can then be constrained to be found within these duties. In this case it is necessary to calculate a value or expression for p which will enable an initial solution to be constructed from it. The data set would have to be ordered in some way so that the subset contains a satisfactory spread of duties from which a feasible solution can be found. The experimentation took place by first ordering the duties by their rank (expression **(5.3)**) with the assumption that an initial solution containing only duties with ranks of less than a certain value would be more efficient. The parameter p could then be set to be the number of duties

with a rank of less than some specified value.

It is possible that a basic feasible solution cannot be found from a subset of the complete set of duties. Certainly the number of feasible solutions will be fewer than from the duty superset. For this reason the method of selecting the initial solution was altered so that, rather than finding the most efficient duty to cover any piece of work, the first duty from the duty list which covers it is chosen. In forming the initial solution the pieces of work are chosen in increasing order of the number of duties which cover them, as in the Sherali TRACS II system.

There is a potential drawback in selecting a duty subset based upon including only those duties which have a rank of less than a certain value, in that the duty subsets will have varied sizes depending on the problem and it may be difficult to form an initial solution from a smaller initial duty subset. Implementing the strategy that a subset of the GMB(S) data can only include duties with a rank of less than 11 produces a subset containing 40% of the total duty set, whereas applying the same criterion to a different data set produces an initial subset containing only 4% of the total duty set.

Observation of schedules indicates that duties with higher ranks are often used in order to achieve optimality, and so this method of selecting a subset may be very inefficient.

Strategy IS2

Since all duties have already been generated it is possible to use information about them at the outset. For instance,

- The Sherali TRACS II model uses information about the complete duty set in order to eliminate relief points which never appear as a start or end point in any duty. If more duties were to be generated at a later stage it would no longer be possible to use this information and all relief points would have to remain as possible changeover points.

- Coverage information from the complete duty set is useful. The current method uses the information regarding the number of duties covering each piece of work in finding an initial solution and is also of use to the column generation process which only adds duties

which cover particular pieces of work.

- Equality constraints are based upon information from the complete duty set so that analysis based only upon a duty subset may be inappropriate once new duties are added to the subset.

- The Sherali weight is based upon summing the duty costs of most expensive duties. By including very expensive duties after the weight has been calculated the worst case is where the weight no longer represents an upper bound on the objective cost, and it would therefore have to be recalculated after each increase in duty set size.

Combining the duty storage and equality constraint routines has reduced the time that it takes to set up the ZIP system in order to use all the information from a generated duty set. As each duty is read in it also has to be stored into a linked list structure. This method avoids the need to constantly access an external file but may be restricted in the size of duty set that it can store.

Strategy IS2 finds an initial solution from the complete duty set. The process for finding an initial solution still terminates once the first available duty has been found to cover a piece of work, rather than searching the complete duty set for the best duty. Since data sets used in a column generation method will be much larger than those currently used, it is quicker to use this method of finding an initial solution. However, this may be compromised by the fact that the initial solution will probably be worse than that used in the Sherali TRACS II system and therefore may require more column generation iterations or more duties to be added to reach optimality.

The initial duty subset then comprises only those duties which appear in the initial solution. This is a very small initial set, but it is more likely that a good feasible solution will be found within the complete set as opposed to within a ranked subset, and so fewer column generation iterations may be required to optimise the relaxed model.

Strategy IS3

Since the duties are stored in linked lists according to the pieces of work that they cover, it

is feasible to search through all of them to find the best duty to cover a piece of work when forming the initial solution. For this reason, the selection of the initial solution uses the same method as is used in the Sherali TRACS II model. This ensures that comparisons of the column generation method and the current TRACS II will be made simpler, both having the same initial solution and Sherali cost addition.

The initial duty set still comprises of only those duties which are found in the initial solution.

Strategy IS4

An initial duty set containing only the duties in the initial solution will probably require many iterations and duty additions to reach an optimal continuous solution. Also, if the size of the duty set is too small at the branch and bound stage, it is possible that an integer solution cannot be found because a crucial duty or duties may have been omitted which would be vital in forming a valid schedule which covers all pieces of work. By defining a coverage value r, such that in the initial duty set there must be at least r duties covering each piece of work, the initial duty set and initial solution can be built up simultaneously. As duties are being considered to find the most efficient one to cover a piece of work, they are added to the initial subset unless there are more than r duties already covering that piece of work. All duties which appear in the initial solution are also added to the initial set regardless of the coverage values. It is possible that some pieces of work will be covered by fewer than r duties, either because there are fewer than r available to cover them or because they never get considered in forming the initial solution. In the latter case, however, the piece of work which gets covered by a duty in the initial solution may also get covered by r duties covering adjoining pieces of work.

For the purpose of the results a coverage value of 10 has been implemented, although further experimentation on this value will be reported in section **8.3**.

Adding further duties

Once the optimal solution has been found over a subset of duties, all pieces of work have a simplex multiplier π_i attached to them. Since the greatest rate of improvement in current objective value will correspond to introducing the duty with the most negative reduced cost into the basis, from **(6.13)** it will be necessary to find

$$\text{MIN } (c_k - \sum_{i=1}^{m} \pi_i a_{ik}). \tag{7.1}$$

where

$c_k =$ cost of duty k
$\pi_i =$ simplex multiplier for row i
$a_{ik} =$ coefficient of duty k in constraint i.

The solution to **(7.1)** will occur for a duty with a low cost which covers a relatively high number of pieces of work with high simplex multipliers. Assuming the duty cost depends only on the work contained in it, the lower costs will relate to those duties which cover less work, but this is balanced out by the fact that there will be fewer simplex multiplier values to subtract from it. Therefore, since the minimum reduced cost calculation depends upon a combination of factors, the duty with the overall minimum reduced cost cannot be found without evaluating the reduced costs of all remaining duties, and so an heuristic is needed to limit the search space at each iteration. Since the simplex multipliers can vary quite significantly, especially compared to the duty costs which all have the added Sherali weight, they can be used as a simple heuristic for choosing which reduced costs to evaluate. It seems sensible that duties which cover the piece of work with the highest simplex multiplier will have relatively low reduced costs as a large value is being subtracted from the duty cost. The column generation method arranges the simplex multipliers in decreasing order after each continuous solution has been found. During each column generation iteration only duties which cover the piece of work with the highest simplex multiplier are considered. Within this set or part of this set any duties whose reduced costs price out negatively are added to the current duty subset.

Strategy CG1

The first implementation of column generation considers ONLY one simplex multiplier for each duty set increase. If there are no duties which price out negatively covering the piece of work corresponding to the highest simplex multiplier then the simplex multipliers are considered in order until there are some duties which can be added to the set. Once all the duties which price out negatively for a particular piece of work are added, the problem is re-solved over the

larger subset.

Method IS1 and CG1				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
1687 duties	44 duties	21	1994 duties	1933 seconds

Table 7.1: GMB(S): ranked subset and one simplex multiplier considered

From Table 7.1, combining this strategy with an initial subset containing only duties with a low value rank produces a final duty set which contains 47% of the available duty set. This is quite a large proportion but takes into account the large initial subset size. This method of choosing the initial solution provides a wide variety of initial subset sizes depending upon the problem considered and is considered too simplistic.

Method IS2 and CG1				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
49 duties	49 duties	58	1037 duties	2887 seconds

Table 7.2: GMB(S): subset of initial solution and one simplex multiplier considered

Table 7.2 shows the results of implementing the method of choosing an initial solution from the duty superset, but without choosing the best duty to cover a particular piece of work. This has produced a poor initial solution. From this, allowing only the initial solution to form the initial subset has provided a smaller final duty set, but at the expense of a larger number of duty additions.

Strategy CG2

Choosing only one simplex multiplier on which to base a duty addition process will be very time-consuming, especially considering that the later iterations will only be adding one or two duties at a time, repeating much of the search that has gone before. Experimentation has taken place with an increase in the number of duties added at each column generation iteration. A minimum value is defined so that at each column generation iteration the simplex multipliers are considered in decreasing order and duties added until at least a specified number of duties are added to the subset or there are no pieces of work left to consider. As in the first strategy all remaining duties are priced out for each simplex multiplier. For experimentation

the method implemented uses a minimum addition of 15 duties per column generation iteration.

Method IS2 and CG2				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
49 duties	49 duties	24	1179 duties	1929 seconds

Table 7.3: GMB(S): subset of initial solution and minimum duties added

From Table 7.3, compared to the previous duty addition strategy more duties are added to the duty subset overall, but this reflects the fact that more duties are added per column generation iteration. Fewer duty additions are also required. The initial solution is considered poor.

Method IS3 and CG2				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
40 duties	40 duties	31	1369 duties	838 seconds

Table 7.4: GMB(S): subset of better initial solution and minimum duties added

Table 7.4 shows how a better initial solution can be found using the Sherali TRACS II method, by choosing the best duty to cover a piece of work rather than the first. The time taken to ascertain an initial solution will increase but an improvement in objective value has reduced the time required to optimise the initial subset. The execution time has reduced dramatically with no other improvements implemented and so the increase in the number of duty additions and the size of the final duty set is not significant.

Strategy CG3

The Sherali TRACS II model uses a steepest edge algorithm [80] to determine the reduced cost which will actually enter the basis. It is reported by Smith [15] that, since there are many thousands of non-basic variables to consider, the steepest edge method takes much longer to calculate per iteration than using pure reduced costs, but the number of iterations is decreased significantly which reduces the overall execution times. With a column generation approach the weights have to be calculated for every new duty added to the subset. A column generation approach optimises the relaxed model on more than one occasion, and the choice of duties to be added to the duty set is based upon pure reduced costs rather than weighted reduced costs so that a steepest edge algorithm may no longer be as appropriate. Table 7.5 gives the result

of using pure reduced costs to determine the duties which enter the basis.

Method IS3 and CG3				
Initial Subset	Initial Solution	No. Subset Addition	Final Subset	Time to LP
40 duties	40 duties	32	1479 duties	3562 seconds

Table 7.5: GMB(S): subset of better initial solution and pure reduced costs calculated

It is possible that a different optimal continuous solution will be determined by using different algorithms to select the basis entrant variable, either by virtue of the fact that there are multiple optimal solutions or simply due to inaccuracies in finite precision arithmetic. For this reason the simplex multipliers may differ at the column generation stage which is why one extra column generation iteration is required and a larger duty subset is ultimately produced. The time taken to produce an optimal continuous solution has increased significantly because the column generation process chooses different duties for each subset increase, and so the steepest edge algorithm will remain in a column generation system. However, in order to reduce some of the execution time by avoiding the need to include weights in the calculations of the reduced costs of potential entrant duties, the duty with the most negative pure reduced cost per column generation iteration is chosen to be the first entrant variable.

Strategy CG4

There is a large amount of repeated calculation of duty reduced costs of potential entrant variables from the duty superset, some of which may never enter the subset. This strategy implements a very simple heuristic which detects duties from the duty superset which will never improve the objective cost and does not allow them to be considered. Banning duties which may improve the objective compromises optimality and so the heuristic provides a very weak upper bound on a duty's reduced cost. The following heuristic assumes that, at most, the highest simplex multiplier available is attached to all the pieces of work that a duty covers, and if this value is less than the cost of the duty then it cannot improve the objective cost during this column generation iteration. Results are given in Table 7.6.

$$\text{If ((No. pieces duty covers * highest simplex multiplier)} \leq \text{Duty Cost) then ban duty.} \tag{7.2}$$

Method IS3 and CG4				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
40 duties	40 duties	30	1265 duties	1721 seconds

Table 7.6: GMB(S): subset of better initial solution and eliminating certain duties

This strategy has also been tested on a selection of other data sets and it is apparent that the extra computational time spent in calculating the upper bound on the reduced cost outweighs the time saved in searching through fewer duties. This bound could be tightened by, for example, summing the x highest simplex multipliers, where x is the number of pieces of work a duty covers, but this requires further computational effort to eliminate a relatively small number of extra duties when the duties are not banned until the later stages of the process.

Strategy CG5

For some simplex multipliers there are a large number of duties added to the subset which have negative reduced costs. The disadvantage in this is that the majority of the new duties added will never be used to improve the objective value and also that the lower limit on the number of additions means that fewer pieces of work will be considered for one column generation iteration. This potentially requires more iterations to consider other simplex multipliers and also reduces the amount of variety in duties added to the set each time. This strategy limits the number of duties added per simplex multiplier and specifies a minimum value by which the duty set should increase (where possible) so that more than one simplex multiplier will be considered during any one column generation iteration. The possible drawback with this method is that by limiting the search for duties with a negative reduced cost per simplex multiplier, many more iterations may be required if a duty at the end of the list is to be used in the optimal solution.

Method IS3 and CG5				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
40 duties	40 duties	22	746 duties	1338 seconds

Table 7.7: GMB(S): subset of better initial solution and addition of more varied duties

The addition of duties per simplex multiplier in Table 7.7 is restricted to 10, and the minimum total duty addition is 30. There are still a relatively large number of subset size increases but

the initial duty subset is limited to those contained in the initial solution. This also results in a relatively small duty subset entering the branch and bound stage as fewer duties are now being added for every column generation iteration. There may be problems in constructing an integer schedule from a smaller available duty set.

Method IS4 and CG5				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
617 duties	40 duties	8	883 duties	1077 seconds

Table 7.8: GMB(S): minimum coverage in initial subset and addition of more varied duties

A limiting parameter is introduced in Table 7.8 which ensures that the initial duty subset contains a minimum number of duties covering each piece of work (where possible). Incorporating a limiting parameter of 10 increased the initial subset size to contain 15% of the total duty set and hence increased the final duty subset to contain 21%. The number of duty additions required has reduced significantly.

Experimentation then took place with a minimum value of duties per iteration of 50 duties and the results are given in Table 7.9:

Method IS4 and CG5				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
617 duties	40 duties	9	885 duties	1084 seconds

Table 7.9: GMB(S): minimum coverage in initial subset and addition of more varied duties

Increasing the duty additions per column generation iteration has not altered the results significantly, only increasing the column generation addition by 1 to add two extra duties overall. The execution time has increased slightly.

Strategy CG6

Strategy CG5 does not take into account the size of the problem being considered and so a constant duty addition limit is inappropriate. A simple calculation of a proportional limit is introduced which does not take into account the actual number of duties remaining which cover pieces of work, but it ensures that in the earlier stages more duties can be added to the subset to reduce the duty total quickly and in the later stages fewer duties need to be introduced to

optimise the objective cost. In order to automate the process of having a minimum number of duties per iteration M and a minimum number of duties per simplex multiplier I, parameters are defined respectively as follows :

$$M = \text{MAX } \{50, X\% \text{ total duties remaining}\} \tag{7.3}$$

$$I = \text{MAX } \{5, Y\% \text{ (total duties remaining/No. pieces of work). }\} \tag{7.4}$$

It is also noted that the simplex multipliers are considered in decreasing order, based upon the assumption that those with larger values are more likely to produce further duties with negative reduced costs. This suggests that since it is time consuming to consider all the simplex multipliers for each column generation iteration, a limit should be placed on the number of simplex multipliers which add further duties to the subset. This is defined as :

$$P\% \text{ (total number of pieces of work where additions are made).} \tag{7.5}$$

Experimentation was carried out with values of 20 for X, Y, and P.

Method IS4 and CG6				
Initial Subset	Initial Solution	No. Subset Additions	Final Subset	Time to LP
617 duties	40 duties	6	920	31.4 seconds

Table 7.10: GMB(S): minimum coverage in initial subset and parameters governing additions

This implementation is reported in Table 7.10 and has drastically reduced the execution time. The number of column generation iterations has also decreased whilst still retaining 22% of the total duty set to enter the branch and bound stage. The limit on the pieces of work considered is tighter than the total duty addition limit. Experimentation with the limiting parameters will be described in section **8.3**.

7.4 Results and Timings

The column generation method using strategy IS4 and CG6 successfully finds the optimal continuous schedule for GMB(S) 19% faster than that of the Sherali TRACS II system given in section **7.2.2**. At this point the method was tested on all the available data sets to compare the timings and ensure that the continuous solutions produced by both methods were equal. The STK(L) data set was run on a Sun Sparc 1 due to quota requirements, but all other data sets were run on a Silicon Graphics (Iris Indigo Workstation with 33MHz R3000 MIPS Processor) and all ran on the local hard drive at over 95% efficiency. All timings are reported in seconds.

7.4.1 Continuous Solution Timings

Table 7.11 gives an indication of the size of each data set tested and a comparison of the running times of the relaxed model on both the Sherali TRACS II system and the implemented column generation system. It is noted that the continuous solution is identical for both methods.

Data	Duties	Constraints	Sherali TRACS II	Column Generation	Improvement
AUC(S)	10529	413	770.2s	961.7s	-24%
AUC(L)	43798	480	5695.0s	1615.5s	72%
CTJ(S)	10775	429	597.7s	378.4s	37%
CTJ(L)	18307	439	1095.2s	536.4s	51%
CTR(S)	10690	443	809.7s	883.3s	-9%
CTR(L)	13792	451	1010.5s	1131.5s	-12%
GMB(S)	4199	127	38.9s	31.4s	19%
GMB(L)	6956	138	76.4s	49.4s	35%
RI2(S)	5176	167	53.5s	39.9s	25%
RI2(L)	7997	180	103.5s	58.8s	43%
STK(S)	10678	250	310.0s	232.9s	25%
STK(L)	90234	418	15039.7s	1242.3s	92%
SYD(S)	8866	302	326.2s	225.1s	31%
SYD(L)	43150	396	5019.9s	914.5s	82%

Table 7.11: Comparison of Timings to Continuous Solution

The timings were improved by using the column generation system in 11 out of the 14 data

sets. The greatest time improvements were for the largest data sets (AUC(L),STK(L),SYD(L)) which would not normally be run through the original TRACS II with its limit of 22,000 duties.

7.4.2 Overall Timings

Table 7.12 displays the timings of the branch and bound process and the overall process time on all data sets. With the column generation system there are fewer duties available at the branch and bound stage and further reduction of the data set using REDUCE (as described in section **5.4.3**) may make it more difficult to find an integer solution. All data sets have been run through Sherali TRACS II and the column generation system to find integer solutions through the branch and bound phase. Results are shown for the column generation system both with and without using the REDUCE process.

Data	Branch and Bound time			Total Time		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
AUC(S)	1361.7s	1212.3s	1942.4s	2131.9s	2174.0s	2904.1s
AUC(L)*	48311.3s	14460.0s	136414.7s	54006.3s	16075.5s	138030.2s
CTJ(S)	383.9s	204.3s	567.5s	981.6s	582.7s	945.9s
CTJ(L)	1672.3s	1118.9s	1356.5s	2767.5s	1655.3s	1892.9s
CTR(S)*	81665.0s	51411.8s	ERROR	82474.7s	52295.1s	ERROR
CTR(L)*	ERROR	ERROR	97543.6s	ERROR	ERROR	98675.1s
GMB(S)	14.0s	6.9s	14.7s	52.9s	38.3s	46.1s
GMB(L)	49.4s	10.2s	33.4s	125.8s	59.6s	82.8s
RI2(S)	79.8s	24.7s	25.9s	133.3s	64.6s	65.8s
RI2(L)	67.2s	9.2s	8.0s	170.7s	68.0s	66.8s
STK(S)	634.8s	96.7s	752.0s	944.8s	329.6s	984.9s
STK(L)*	14813.7s	573.7s	52177.4s	29853.4s	1816.0s	53419.7s
SYD(S)	21.6s	104.5s	108.3s	347.8s	329.6s	333.4s
SYD(L)	654.3s	159.3s	837.70s	5674.2s	1073.8s	1752.2s

Table 7.12: Comparison of Branch and Bound Timings and Total Timings

* represents data sets where no integer solution is found

ERROR occurs in branch and bound

It should be noted that since the branch and bound method stops when a good solution is found and all three systems take a different path through the search tree, it is difficult to compare the quality and timings of results produced by the branch and bound phase. However, it is necessary to ensure that the duty total in the integer solution is the same in all three methods and possibly even better than that produced by the smaller duty set. Also, since the execution times are improved in the continuous LP phase, it is preferable that the total execution times also improve over those of Sherali TRACS II for the same problem. The minimisation of execution time is a major consideration for most users and since larger data sets are available it is important that execution times of a column generation system, whilst preferable to another method on the same data set, are reasonable compared to those achieved from other scheduling systems.

Without reducing the data set in the column generation system there are more duties entering the branch and bound phase, and, as expected, most branch and bound execution times are slower than the column generation version which does reduce its duty set. In 8 out of the 12 data sets which execute without errors, the column generation system without REDUCE takes longer than the Sherali TRACS II system, which does use the duty reduction procedure.

Four of the data sets fail to find integer solutions using any of the methods described, either by failing to find a solution with the remaining duties or by exhausting the node limit of 500. This suggests that reducing the problem in a column generation system does not hinder the branch and bound by its limited remaining duty subset. In fact, in only 1 out of the 14 data sets does the non-reduced method execute faster, and in this case it is by a negligible margin. The column generation method using REDUCE runs slower than Sherali TRACS II in the branch and bound phase for data set SYD(S) but the total timing is actually faster. For the total timings the reduced column generation method runs slower on AUC(S) which also took longer to find the continuous solution For those data sets which did find an integer solution, all of them found solutions within 28 minutes. This compares to the slowest Sherali TRACS II run on a small data set which finds a solution in 36 minutes, and so even with larger data sets the execution time is acceptable.

7.4.3 Detailed Analysis of Results

The following tables give a more detailed analysis of the comparison of the results produced by the Sherali TRACS II system and the column generation system. The schedule cost incor-

porates the wage and penalty costs of the duties in the final solution and does not include the Sherali weighting.

AUC data

The AUC(S) data has to cover 413 pieces of work, and the AUC(L) data has to cover 480.

Feature	AUC(S)			AUC(L)		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
Total no. duties	10529	10529	10529	43798	43798	43798
Initial subset	-	2067	2067	-	2308	2308
Initial soln	115	115	115	120	120	120
No. iterations	2570	-	-	5362	-	-
Subset additions	-	8	8	-	7	7
Final subset	-	3740	3740	-	6112	6112
LP solution	86.41	86.41	86.41	85.33	85.33	85.33
Time to LP	770.2	961.7	961.7	5695.0	1615.5	1615.5
No. duties	2405	1352	3740	4346	1779	6112
nodes created	22	25	25	500(*)	500(*)	500(*)
Final Solution	87	87	87	NONE	NONE	NONE
Overcover	1	1	2	-	-	-
Final Cost	139288	142897	140309	-	-	-
total time	2131.9	2174.0	2904.1	54006.3	16075.5	138030.2

Table 7.13: AUC Data : Comparison of Results

* represents data sets where node limit is reached

For the smaller data set the column generation process starts with 20% of the total data set available and increases this to 36% after 8 additions to the duty set. The time taken to produce the optimal continuous solution is longer than for the Sherali TRACS II system. The duty reduction technique provides the branch and bound with 13% of the total duties available in the column generation method compared to 23% of the duties using Sherali TRACS II but creates 3 more nodes in finding a comparable integer solution. Where the reduction technique

is not used, the total execution time of the smaller problem is only 2% slower than for the Sherali TRACS II due to a faster execution of the branch and bound process.

The larger data set requires fewer duty additions and produces a continuous solution which is over a duty less than with the smaller data set. There is also a significant time reduction in using a column generation approach to the LP stage. All three methods fail to find an integer solution within 500 nodes and this will be investigated further in section **8.2**.

CTJ data

The CTJ(S) data has to cover 429 pieces of work, and the CTJ(L) data has to cover 439.

Feature	CTJ(S)			CTJ(L)		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
Total no. duties	10775	10775	10775	18307	18307	18307
Initial subset	-	1607	1607	-	1734	1734
Initial soln	112	112	112	112	112	112
No. iterations	1988	-	-	2175	-	-
Subset additions	-	5	5	-	7	7
Final subset	-	2586	2586	-	3494	3494
LP solution	87.50	87.50	87.50	86.42	86.42	86.42
Time to LP	597.7	378.4	378.4	1095.2	536.4	536.4
No. duties	2459	1087	2586	4050	1404	3494
nodes created	19	12	29	13	80	30
Final Solution	88	88	88	87	87	87
Overcover	27	25	36	21	23	28
Final Cost	95205	96013	96508	106895	102986	104600
total time	981.6	582.7	945.9	2767.5	1655.3	1892.9

Table 7.14: CTJ Data : Comparison of Results

The smaller data set is solved more quickly using a column generation system and significantly so if the duty set is reduced after the continuous optimum has been ascertained where only 10% of the available duties are used to find the integer solution in 7 fewer nodes. The non-reduced

version of the column generation approach uses a slightly larger duty set to enter the branch and bound phase than for the Sherali TRACS II method and takes less time overall since most time is saved in requiring only 5 additions to the duty set to find an LP optimum from 24% of the total duty set. Both column generation systems produce a schedule with a greater cost than that of Sherali TRACS II.

The larger data set produces an integer solution with one duty fewer than that found from the smaller data set although the overall cost is higher. The reduced column generation version does use many more nodes than both other methods but still finds the integer solution in 40% of the time taken by Sherali TRACS II.

CTR data

The CTR(S) data has to cover 443 pieces of work, and the CTR(L) data has to cover 451.

Feature	CTR(S)			CTR(L)		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
Total no. duties	10690	10690	10690	13792	13792	13792
Initial subset	-	1993	1993	-	2070	2070
Initial soln	124	124	124	123	123	123
No. iterations	2599	-	-	2657	-	-
Subset additions	-	7	7	-	10	10
Final subset	-	3603	3603	-	4112	4112
LP solution	86.62	86.62	86.62	86.54	86.54	86.54
Time to LP	809.7	883.3	883.3	1010.5	1131.5	1131.5
No. duties	4943	2003	3603	6379	2342	4112
nodes created	500(*)	500(*)	ERROR	ERROR	ERROR	500(*)
Final Solution	NONE	NONE	.	.	.	NONE
Overcover	-	-	.	.	.	-
Final Cost	-	-	.	.	.	-
total time	82474.7	52295.1	-	-	-	98675.1

Table 7.15: CTR Data : Comparison of Results

* represents data sets where node limit is reached

Both the smaller and larger data sets generate an LP solution with under 87 duties but fail to find an integer solution with 87 duties either within the 500 node limit or because of errors caused within the branch and bound search. The time taken by the column generation system to find the continuous solution is slower for both sizes of problems.

The non-reduced column generation system fails during the branch and bound on the smaller data set after 200 nodes have been searched. Both the Sherali TRACS II system and the reduced column generation system fail during the branch and bound on the larger data set and so the error does not only occur with a final duty set provided by a column generation system. The errors occur in the process which has not been altered throughout this research. Attempts to find an integer solution for CTR will be reported in the section **8.2**.

GMB data

The GMB(S) data has to cover 127 pieces of work, and the GMB(L) data has to cover 138.

Feature	GMB(S)			GMB(L)		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
Total no. duties	4199	4199	4199	6956	6956	6956
Initial subset	-	617	617	-	719	719
Initial soln	40	40	40	42	42	42
No. iterations	377	-	-	463	-	-
Subset additions	-	6	6	-	6	6
Final subset	-	920	920	-	1126	1126
LP solution	33.50	33.50	33.50	32.85	32.85	32.85
Time to LP	38.9	31.4	31.4	76.4	49.4	49.4
No. duties	721	346	920	2039	521	1126
nodes created	2	3	6	2	7	6
Final Solution	34	34	34	33	33	33
Overcover	4	4	3	2	1	2
Final Cost	27488	27488	28075	34600	36552	35941
total time	52.9	38.3	46.1	125.8	59.6	82.8

Table 7.16: GMB Data : Comparison of Results

The GMB data set is the smallest problem provided but the larger duty set allows a schedule to be produced with one fewer duty than previously found using Sherali TRACS II. Both the continuous phase and the branch and bound phase execute more quickly using the column generation system, even though the non-reduced version on the smaller duty set enters the branch and bound phase with more duties than Sherali TRACS II. With the reduced column generation method only 8% of the duty set enters the branch and bound process with the smaller set and 7% with the larger set, suggesting that the REDUCE technique is not detrimental in finding an integer solution from a smaller subset although it is possible that the quality is affected by removing certain duties at this stage.

RI2 data

The RI2(S) data has to cover 167 pieces of work, and the RI2(L) data has to cover 180.

Feature	RI2(S)			RI2(L)		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
Total no. duties	5176	5176	5176	7997	7997	7997
Initial subset	-	853	853	-	932	932
Initial soln	54	54	54	54	54	54
No. iterations	431	-	-	556	-	-
Subset additions	-	5	5	-	5	5
Final subset	-	1160	1160	-	1461	1461
LP solution	44.19	44.19	44.19	43.79	43.79	43.79
Time to LP	53.5	39.9	39.9	103.5	58.8	58.8
No. duties	1136	539	1160	1010	700	1461
nodes created	6	5	3	2	2	2
Final Solution	45	45	45	44	44	44
Overcover	10	9	9	8	8	8
Final Cost	40379	41380	40729	41221	41221	41221
total time	133.3	64.6	65.8	170.7	68.0	66.8

Table 7.17: RI2 Data : Comparison of Results

For this problem the larger duty set enables a schedule to be produced with one duty fewer

than with the smaller duty set. There is little difference in total execution times between the reduced and non-reduced versions of the column generation system, which in any case runs over 50% faster than the Sherali TRACS II version on both duty sets. All three methods produce the identical cost solutions from the larger duty set.

STK data

The STK(S) data has to cover 250 pieces of work, and the STK(L) data has to cover 418.

Feature	STK(S)			STK(L)		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
Total no. duties	10678	10678	10678	90234	90234	90234
Initial subset	-	1355	1355	-	1844	1844
Initial soln	76	76	76	80	80	80
No. iterations	1222	-	-	8995	-	-
Subset additions	-	7	7	-	7	7
Final subset	-	2421	2421	-	8297	8297
LP solution	60.55	60.55	60.55	57.83	57.83	57.83
Time to LP	310.0	232.9	232.9	15039.7	1242.3	1242.3
No. duties	1931	846	2421	5553	1670	8297
nodes created	35	6	40	139	14	500(*)
Final Solution	61	61	61	NONE	NONE	NONE
Overcover	1	2	1	-	-	-
Final Cost	83302	81841	84423	-	-	-
total time	944.8	329.6	984.9	29853.4	1816.0	53419.7

Table 7.18: STK Data : Comparison of Results

* represents data sets where node limit is reached

The smaller duty set provides an integer schedule with 61 duties. The column generation system using the REDUCE technique sends significantly fewer duties into the branch and bound phase and creates only 6 nodes to find an integer schedule which is cheapest even though it has an extra overcovered piece of work.

The larger duty set is actually the largest available duty set and execution time to the continuous LP stage is 92% faster than that of Sherali TRACS II which would normally be limited to 22000 duties. However, none of the three methods succeed in finding an integer solution with 58 duties. Both TRACS II and the reduced column generation system terminate before the node limit is reached to indicate that an integer solution cannot be found with the duty set available at the branch and bound stage. The non-reduced column generation system cannot find an integer solution within 500 nodes. Attempts to find an integer solution will be reported in the section **8.2**.

SYD data

The SYD(S) data has to cover 302 pieces of work, and the SYD(L) data has to cover 396.

Feature	SYD(S)			SYD(L)		
	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE	Sherali TRACS II	Col. Gen. REDUCE	Col. Gen. No REDUCE
Total no. duties	8866	8866	8866	43150	43150	43150
Initial subset	-	1058	1058	-	1411	1411
Initial soln	67	67	67	68	68	68
No. iterations	1187	-	-	3420	-	-
Subset additions	-	7	7	-	7	7
Final subset	-	1978	1978	-	4853	4853
LP solution	56.00	56.00	56.00	56.00	56.00	56.00
Time to LP	326.2	225.1	225.1	5019.9	914.5	914.5
No. duties	2895	944	1978	6966	1483	4853
nodes created	10	7	11	32	27	20
Final Solution	56	56	56	56	56	56
Overcover	50	50	39	52	60	67
Final Cost	35137	36371	35011	32860	33720	32537
total time	347.8	329.6	333.4	5674.2	1073.8	1752.2

Table 7.19: SYD Data : Comparison of Results

The first thing that should be noted for the large data set is that there is an uncovered piece of work in the initial solution. The Sherali weighting therefore sums the 69 most expensive duties

to ensure that this value will provide an upper bound on the optimal schedule cost.

The larger data set produces a comparable solution to the smaller data set but with lower costs. It does however require over three times longer to find a solution to the larger set. There is a significant time reduction using the column generation system on the larger data set over the Sherali TRACS II system but most of the time is taken by the latter to find the same continuous solution as is produced by the smaller data set.

7.4.4 Overview

It is impossible to compare directly the timings or results of the column generation method presented and that of the Hastus Crew-Opt method as their structures differ, and two different techniques are used, but it is noted that problem **A** (reviewed in section **6.4.2**) is similar in size and structure to the RI2(S) problem. It is not known how many duties were generated to initialise the column generation procedure or if any duties were removed from the duty set at any stage, but Crew-Opt required 307 shortest path calculations and generated 2926 new duties in total. From a generated set of 5176 duties the column generation system reported in section **7.4** produced the optimal LP solution (optimal given the generated duty set) from a total of 1160 duties after 40 seconds. The duty set was then reduced to 539 duties and the integer schedule was found after a further 82 seconds. Problem **F** (reviewed in section **6.4.2**) contains a large schedule which had to be decomposed using Crew-Opt. The final solution required 156 duties and to date the largest problem tested using the customised column generation system and reported in section **7.4** produces a schedule with 88 duties.

The aim of the column generation system is to provide better schedules by allowing more duties to be available and also to provide results at least as fast as the Sherali TRACS II system. Results have shown that in all cases the reduced column generation system executes faster than the non-reduced system and on all data sets tested the limited duty set entering the branch and bound phase was not a disadvantage in forming an integer solution. It is still noted however that it is possible that the duty reduction on some data sets may prove to be too extreme to be able to produce an integer schedule but results using the non-reduced version of the column generation system will no longer be reported.

In 12 out of the 14 data sets the column generation system produced an integer solution faster

than the Sherali TRACS II system on the same duty set, and 3 out of the 7 problem formed a final schedule with fewer duties by using a larger generated duty set. One of the other data sets produced a schedule with the same number of duties with a lower cost from a larger duty set, but the other 3 problems failed to find an integer solution from a larger set. For the SYD problem the larger data sets produced a schedule with the same number of duties but the cost was lower. For the remaining data sets which improved upon the number of duties in the final schedule, the final cost was higher than that produced by the smaller duty set. It is certainly possible that the minimisation of duties as the main objective may introduce more undesirable features into duties, and penalty costs added for these problems are relatively high. As an example, for a three part duty the penalty cost can be as much as 1000, which implies that the difference in cost may equate to replacing a small number of two part duties with three part duties.

Chapter 8

Refinement of the Column Generation Method Within ZIP

8.1 Introduction

This chapter describes further experimentation which has been carried out in order to improve the column generation method described in Chapter 7. Successful investigations have been implemented and the results of refined column generation system on the available duty sets are reported in section **8.4**.

8.2 Failure to Find an Integer Solution

Of the fourteen duty sets tested four failed to produce an integer solution using either Sherali TRACS II or a column generation method with and without duty reduction at the branch and bound stage. All three methods are described in Chapter 7. There are a number of reasons why a method may fail to find an integer solution:

1. The branch and bound tree is being limited to too few nodes, and an integer solution may be found after exploring a much larger number of nodes.

2. Duties which are necessary in order to form an integer solution which covers all of the pieces of work may not be available to the branch and bound process.

3. There may be no integer solution available with the specified number of duties.

The branch and bound stage terminates without finding a solution before the node limit is reached with both the Sherali TRACS II and reduced column generation systems on the STK(L) problem. Increasing the node limit in this case would have no effect on the solution. Willers [14] recommended that a node limit of 100 is high enough to avoid prematurely abandoning a branch and bound search in a commercial version of TRACS II but for more conclusive results and an opportunity to improve on an integer solution a value of 500 is used. Increasing this value would be at the expense of much higher execution times and would be much less acceptable to the users.

Duties may not be available to the branch and bound process because REDUCE, as described in section **5.4.3** removes some from the set. In the case of the column generation method a version is tested without REDUCE. For this version to fail also, suggests that the duty set selected in the column generation phase did not include some duties in forming an integer schedule. In order to test the robustness of REDUCE it would have to be omitted in the Sherali TRACS II system to allow all duties to enter the branch and bound stage. This would not be carried out in a commercial system, especially with much larger duty sets having been generated, and it has been shown from the column generation system that the reduction technique speeds up the execution times in most problems. For the non-reduced column generation system, in order to ensure that an omission of vital duties is not the cause of the difficulties in finding an integer solution, all generated duties would have to be added to the subset once the optimal continuous solution had been found. Implementing such a strategy would defeat the object of a column generation technique which works with a smaller duty set but implicitly considers many more duties.

Generally, a smaller duty set is available to the branch and bound process with a column generation system. The reduction technique may then be inappropriate as it may remove too many further duties to enable an integer solution to be found easily. However, since the results reported in section **7.4** show no disadvantage over the non-reduced version, an alternative technique would have to consider a different set of duties at the branch and bound stage. For problems for which there is evidence to suggest that a column generation technique cannot find a known integer solution (e.g, if TRACS II were tested and found a solution), a different

strategy could be adopted. One such possibility would be to use the REDUCE process on the entire duty set, rather than the final subset resulting from the column generation method.

For all three methods tested on any problem in chapter 7, each attempts to find integer solutions from a different duty set. For situations in which none of the systems can find an integer solution there is a higher probability that, in fact, none exists than if only one system were tested, even with the limitations mentioned. A potential drawback in allowing many more duties to be available may be that the continuous solution can be improved greatly but there is a limit on the improvement of an integer solution.

The four data sets which fail to produce integer schedules are CTR(S), CTR(L), AUC(L) and STK(L). In the case of CTR(S) and CTR(L) no integer solution has been found at all. Both sizes of duty set produce a continuous solution with under 87 duties and so both search for an integer solution with 87 duties. For the cases of AUC(L) and STK(L), integer solutions have been found with respectively 87 and 61 duties on the smaller data sets but the branch and bound on the larger set attempts to find integer solutions with respectively 86 and 58 duties. The AUC(L) data set reaches the node limit of 500 in all three systems but the STK(L) data set terminates before the node limit is reached on all but the non-reduced column generation system.

8.2.1 Specifying a Target Number of Duties

Since an integer solution cannot be found for data sets CTR(S),CTR(L), AUC(L) and STK(L), a constraint is added to the model to force, if possible, an integer solution requiring one more duty than previously attempted. With the AUC(L) data, a known 87 duty schedule can be found from the smaller duty set and with the STK(L) data set, the branch and bound will search for a schedule with 59 duties. For the CTR data, no solution has been found with 87 duties and so an attempt is made to ascertain if one can be found with 88 duties.

The Sherali TRACS II system has the ability to pick up an earlier continuous solution and reoptimise if any conditions have changed. It would therefore be preferable to pick up the continuous solution provided by the first execution, add the necessary constraint, and reoptimise. However, the column generation remains a one pass method at present, and so both systems add the constraint initially and begin the execution from scratch.

Table 8.1 compares the Sherali TRACS II system with the reduced column generation system on the above four data sets with an increased duty target.

Feature	AUC(L)		STK(L)		CTR(S)		CTR(L)	
	Sherali TRACS II	Col Gen	Sherali TRACS II	Col Gen	Sherali TRACS II	Col Gen	Sherali TRACS II	Col Gen
Total duties	43798	43798	90234	90234	10690	10690	13792	13792
Initial subset	-	2308	-	1844	-	1993	-	2070
Initial soln	120	120	80	80	124	124	123	123
No. iterations	7173	-	64027	-	2753	-	3079	-
Set additions	-	6	-	6	-	8	-	9
Final subset	-	5417	-	7773	-	3524	-	4073
LP solution	87	87	59.00	59.00	88.00	88.00	88.00	88.00
Time to LP	8678.4	1246.2	128385.7	1335.7	1024.7	1098.2	1337.8	1266.7
No. Duties	3072	1299	2820	1100	3744	1637	4765	2040
Nodes Created	61	22	15	26	500(*)	500(*)	500(*)	56
Final Solution	87	87	59	59	NONE	NONE	NONE	88
Overcover	0	3	0	3	-	-	-	10
Final Cost	125455	128996	65361	64367	-	-	-	125414
Total Time	9981.3	1416.8	128634.1	1752.7	24214.0	20979.9	40564.7	3130.1

Table 8.1: Specifying a Target Number of Duties for Certain Problems

The most notable result is that in the case of STK(L) a schedule with the target number of duties has been found, and so the solution has two fewer duties and a lower cost than the schedule formed from the smaller duty set. The addition of a constraint has also produced an integer solution for AUC(L) which has a lower cost than the solution previously produced. The cost of the solution is higher than that produced by Sherali TRACS II but this is not significant and could be caused by the fewer available duties at the branch and bound stage, the cost of the overcover in the solution, or the search for a better solution only within a small tolerance. No integer solution still can be found from the small data set for CTR, but the column generation method has found a schedule with 88 duties from the large data set, where Sherali TRACS II terminates at the node limit.

The execution time of the relaxed Sherali TRACS II model has increased, particularly in the

case of STK(L) where the timing is over 8 times that of the model without the duty target constraint added. The total timings cannot be compared as previously no integer solution could be found. The column generation timings remain acceptable compared to that of other data sets whereas the Sherali TRACS II model has more difficulty in producing solutions from large data sets with an added constraint.

8.2.2 Summary

There is a potential problem in that a column generation model may produce continuous solutions which provide too low a duty total from which to form an integer schedule. The addition of a constraint which specifies a higher duty total may enable an integer schedule to be produced without a large increase in the total execution times and the solution would still be no more costly in terms of duties over a solution produced by a smaller generated set of duties.

8.3 Altering the Initial Subset Size and Duty Increase Per Iteration

Since the column generation method on the AUC(S) problem provided the worst increase in execution timing to form the continuous solution it will be used as a basis for further investigation into altering the number of duties which are selected for addition to the subset at each column generation iteration.

Many duties will not be used once they have been added to the duty subset, but if fewer duties are selected per simplex multipier then it may require more iterations to select a duty vital to the optimal continuous solution, and also it may prove more difficult to find an integer solution from a smaller total set. Selecting too many more duties per iteration, however, may increase the execution time per column generation iteration considering that many of these additional duties will be unusable. This may also cause more duties to be searched within the branch and bound phase, with a resulting increase in its execution time.

Some experimentation has taken place with the parameters which limit the addition of duties to the subset. The total number of duties added per column generation iteration is limited by

the percentage of pieces of work considered and the percentage of duties remaining. The first places a limit on the number of pieces of work which can produce further duties to add to the subset. The second provides a duty addition target which, once it has been met, terminates the column addition iteration once all necessary duties covering the current piece of work have been added. The percentage of duties per piece of work limit the number of duties which can be added to the subset for any simplex multiplier under consideration.

It is noted that tables 8.2, 8.3 and 8.4 do not exhaustively test all possible combinations of parameters since they all interact, but value choice is based upon the intention to tighten or relax the limits which have the most effect on the number of duties chosen.

Analysis of Initial Coverage Parameter

The initial coverage parameter determines a lower limit (where possible) on the number of duties in the initial subset which cover each piece of work. As a duty is being considered for inclusion in the initial solution it is added to the subset if there are fewer than the specified number of duties covering ANY of the pieces of work that the duty covers. This means that there may be many more duties covering some pieces of work than the specified limit. It is also possible that some pieces of work cannot be covered by the specified number of duties either because there are fewer duties in total covering them or because a particular piece of work does not get explicitly considered in the initial solution.

- If the parameter is set too low it may be necessary to perform more duty additions in order to reach an optimal continuous solution. Further, fewer duties at the branch and bound stage may make it more difficult to find an integer solution.

- If the parameter is set too high the initial subset will be relatively large so making calculations more time consuming. There will be a larger proportion of unproductive duties in the set which defeats the object of using a column generation procedure to include more useful duties later.

Table 8.2 presents the results of implementing different coverage values, indicated by the first column. The smallest value tested is 5. This produces a continuous solution very quickly, even

Initial Cover	Initial Set	Pieces of work(P%)	Remaining Duties(X%)	Duties Per Piece (Y%)	Subset Additions	Final Subset	Time To LP	Total Time
5	1209	20	20	20	10	3233	845.6s	No soln
5	1209	30	20	20	10	3384	948.8	2413.5
5	1209	20	20	50	8	3487	821.0	No soln
10	2067	20	20	20	8	3740	961.7s	2471.0s
10	2067	30	20	20	8	3954	961.2	4231.3s
10	2067	20	20	30	9	3686	1102.7	2545.6
10	2067	30	20	30	8	3912	1040.5	1890.5
10	2067	20	20	50	7	3935	836.5	3015.1
10	2067	10	10	60	10	3824	1117.6	4425.6
20	3361	20	20	20	7	4489	982.7	2285.3s
20	3361	30	20	20	6	4624	935.4	No soln
20	3361	20	20	30	7	4489	973.4	2256.3
20	3361	30	20	30	6	4624	987.5	3352.0
20	3361	20	20	50	6	4530	862.4	3188.5
20	3361	10	10	60	8	4565	1175.0	2172.6
30	4502	20	20	20	5	5266	1006.1s	1843.8s

Table 8.2: Altering the Initial Coverage Parameter

though many duty additions are required, but fails to find an integer solution in two out of the three experiments. The experiments attempt to overcome the problem of having too few duties at the branch and bound stage, but the final duty set remains relatively small. Evidence from other experiments suggests that any further increase in duty additions by altering the parameters will increase the execution time without the guarantee that sufficiently more duties would be added to enable an integer solution to be formed.

A coverage value of 30 produces an initial duty subset consisting of more duties than are necessary to produce an integer solution. The duties added to the initial subset are selected by the order of the original duty set and so there is no guarantee of the quality of these duties. This suggests that although the number of duty additions decrease as the coverage parameter increases, this is outweighed by the increase in computation time of the relaxed model.

Experiments with a coverage value of either 10 or 20 produce the best results in terms of execution time and availability of an integer solution. Increasing the coverage parameter tends to increase the overall execution time even though the number of column generation iterations decreases and so a coverage value of 10 will remain as the implemented value.

Total duties added per Column Generation Iteration

Without limit, the column generation process would search through all the pieces of work in decreasing order of the value of their corresponding simplex multiplier and would add every duty with a favourable reduced cost to the duty subset. Since it is time consuming to consider all simplex multipliers at every column generation iteration, and they have been ordered so that the later multipliers are less likely to introduce duties which would improve the objective value, there is an upper limit on the percentage of pieces of work which are allowed to add further duties during each column generation iteration. The upper limit is:

$$P\% \text{ (total number of pieces of work where additions are made).} \tag{8.1}$$

There is also a parameter limiting the total number of duties which can be added to the subset per iteration. This limit is defined to be :

$$M = \text{MAX } \{50, X\% \text{ total duties remaining.}\} \tag{8.2}$$

The value of 50 is used if the percentage of duties remaining has a lower value, so that, towards the end of the process, the column generation system does not add so few duties to the subset that more column generation iterations are required. It is unlikely that this value will be used since the column generation system generally deals with a relatively small working subset of a much larger duty set and so the percentage of remaining duties will still be quite high. During any iteration the addition process will terminate as soon as either limiting condition is reached.

- If the limits are too low more column generation iterations may be required to add certain vital duties, and there is a lack of variety of duties entering the subset which may cause problems in trying to form an integer solution.

Initial Cover	Initial Set	Pieces of work(P%)	Remaining Duties(X%)	Duties Per Piece (Y%)	Subset Additions	Final Subset	Time To LP	Total Time
10	2067	10	10	60	10	3824	1117.6	4425.6
10	2067	20	20	20	8	3740	961.7s	2471.0s
10	2067	20	20	30	9	3686	1102.7	2545.6
10	2067	20	20	40	7	3785	867.7	1597.8
10	2067	20	20	50	7	3935	836.5	3015.1
10	2067	20	20	60	7	4014	896.2	2078.3
10	2067	20	10	60	7	4014	896.2	2078.3
10	2067	30	20	20	8	3954	961.2	4231.3s
10	2067	30	20	30	8	3912	1040.5	1890.5

Table 8.3: Altering the Number of Additions Per Iteration

- If the limits are too high it is time consuming to search through the later duties, especially when they may already have been covered by duties added earlier and it is considered that they are less likely to produce as significant an objective improvement

In all problems tested the percentage of pieces of work considered provides a tighter limit on the duty addition, and the absolute duty limit is never reached. Since the number of duties in the superset can, in some problems, be very large, taking a percentage of duties to add to the set may be an unrealistic limit, although it does have the advantage of reducing as the process continues.

Remaining Duties

Setting the upper limit on the number of duties which could be added to be 20% of the remaining duties is too high to restrict the generation of further duties, even when the limit on the percentage of pieces of work which can be considered is lowered. Lowering the value to 10% also has no effect, but will remain implemented in the column generation system for smaller duty sets.

Percentage of Pieces of Work

Increasing the parameter to consider 30% of the pieces of work each time bears no significant difference in execution times over those tested with 20% and so the extra duties added make little difference to the solution. However the increase in duty numbers at the branch and bound

stage make it more difficult to find an integer solution. A value of 10% was tested, which creates more iterations as fewer duties are added for each iteration, and the execution time to the continuous solution increases. A value of 20% will remain implemented in the refined column generation system.

Total duties added Per Piece of Work

For the simplex multiplier under consideration it would be possible to calculate the reduced costs of all duties covering its corresponding piece of work. However, some pieces of work will be covered by many duties and there would be a large amount of repeated calculation. A limit I is placed on the number which can be added. This is defined to be :

$$I = \text{MAX } \{5, Y\% \text{ total duties remaining/total no. pieces of work.}\} \tag{8.3}$$

The value of 5 ensures that where the percentage has a very low value, a reasonable number of duties can still be added, thus avoiding an increase in the number of iterations.

- If the value of I is set too low many more iterations may be required to consider vital duties which appear at the end of the list of duties covering a piece of work.

- If the value of I is set too high, unprofitable duties may be added and an unbalanced number of duties covering certain pieces of work will reduce the variety of duties available at the branch and bound stage.

The estimated value of the remaining duties per piece of work is very crude and could vary drastically between problems depending on the number of duties in the superset. A more sophisticated limit could be developed based upon knowledge of the number of duties covering specific pieces of work, but the limit shown should merely provide a limit which will reduce as the subset size increases to reflect the rate of improvement of the objective function. The calculations take into account the number of duties which have already been added in the iteration which cover other pieces of work.

Initial Cover	Initial Set	Pieces of work(P%)	Remaining Duties(X%)	Duties Per Piece (Y%)	Subset Additions	Final Subset	Time To LP	Total Time
10	2067	20	20	20	8	3740	961.7s	2471.0s
10	2067	20	20	30	9	3686	1102.7	2545.6
10	2067	20	20	40	7	3785	867.7	1597.8
10	2067	20	20	50	7	3935	836.5	3015.1
10	2067	20	20	60	7	4014	896.2	2078.3
10	2067	20	10	60	7	4014	896.2	2078.3

Table 8.4: Altering the Number of Duty Additions Per Piece of Work

Considering 20% of the estimated remaining duties per piece of work resulted in a lower value than the minimum limit of 5 in this problem, and so this only proved binding in the early column generation iterations. As the percentages increased, the execution times decreased, as did the number of duty additions. However, at 60% the number of duties at the branch and bound stage was quite high and the timings started to increase again. Although the total execution time is quite high for a 50% addition, the branch and bound timing cannot be used to influence the choice as it depends on the route through the search tree. A value of 50% will be implemented in the refined column generation system.

8.3.1 Proposed Parameters

The results suggest that the following equations, which limit the number of duties added to the subset on each iteration, improve the continuous execution timings on the AUC(S) data set by the greatest margin without adversely affecting the branch and bound timing.

The equations respectively limit the duties added per piece of work, the total number of duties, and the number of pieces of work from which further duties can be selected.

$$I = \text{MAX } \{5, 50\% \text{ (total duties remaining/No. pieces of work) } \}$$
$$M = \text{MAX } \{50, 10\% \text{ (total duties remaining)}\}$$
$$P = 20\% \text{ (total number of pieces of work where additions are made)}$$

(8.4)

8.4 Implementation and Results

Table 8.5 shows the results of implementing the proposed parameters on all duty sets, except CTR(S) which did not produce a solution. Data sets AUC(L), CTR(L) and STK(L) have a constraint appended to specify a target number of duties to ensure that an integer solution can be found. The execution times of both the previous column generation system and the Sherali TRACS II system are also displayed.

	Subset Additions		Time to LP			Total Time		
Data	Prev	Current	Prev	Current	TRACS II	Prev	Current	TRACS II
AUC(S)	8	7	961.7	836.5	770.2s	2174.0	3015.1	2131.9
AUC(L)	6	4	1246.2	1241.8	8678.4	1416.8	5472.3	9981.3
CTJ(S)	5	4	378.4	361.9	597.7	582.7	1186.8	981.6
CTJ(L)	7	5	536.4	541.4	1095.2	1655.3	1118.0	2767.5
CTR(L)	9	7	1098.2	1066.8	1024.7	3130.1	3228.5	-
GMB(S)	6	4	31.4	27.4	38.9	38.3	33.5	52.9
GMB(L)	6	4	49.4	43.4	76.4	59.6	53.8	125.8
RI2(S)	5	5	39.9	45.6	53.5	64.6	71.8	133.3
RI2(L)	5	4	58.8	57.6	103.5	68.0	72.2	170.7
STK(S)	7	6	232.9	224.9	310.0	329.6	468.9	944.8
STK(L)	6	6	1335.7	1230.7	128385.7	1752.7	1454.5	128634.1
SYD(S)	7	6	225.1	241.9	326.2	329.6	260.7	347.8
SYD(L)	7	5	914.5	897.6	5019.9	1073.8	1051.2	5674.2

Table 8.5: Comparison of Timings With Improved Column Generation System

There is an improvement in execution times of 5% up to the continuous solution over the previous column generation method in 10 out of 13 data sets, leaving only the AUC(S) data set which takes longer than the Sherali TRACS II system to reach the optimal continuous solution.

The total timings increased over the previous column generation method for 7 out of the 13 data sets. The worst case being AUC(L) which now takes 2.5 times longer to find an integer solution. Although there is an overall increase in total execution times by 10%, preference is given to the reduction of execution times up to the continuous solution because the branch and bound timings can vary depending on the route through the search tree. Overall solution times are still better by 41% than the Sherali TRACS II system, with only data sets AUC(S) and

CTJ(S) taking longer to find an integer schedule using a column generation method.

8.5 Sub-Optimal Solution

It is possible to enter the branch and bound phase with the current duty subset once the optimal continuous solution has been found. Terminating the column generation process before optimality over the duty superset has been reached would have the advantage that time would be saved in searching for further duties to add to the set. It is noted that the later additions to the duty set often add very few duties and the final column generation iteration which ensures optimality takes a relatively large amount of execution time. For the later duty additions to the subset the objective value is decreasing slowly, so that the tendency is for the duty total to change fractionally but the duty combinations are improved to reduce the overall duty cost. Entering the branch and bound phase when the integer part of the duty total cannot decrease further will not affect the quality of integer solution because the branch and bound process will still search for a schedule with the next highest duty total. There will, however, be fewer duties and if they use different relief points than the superset optimum, the reduction procedure will allow a different set of duties to enter the branch and bound.

With the column generation method tested in Chapter 7 the maximum number of duty additions reported is 10, but most problems only require up to 7, and so in many cases the number of column generation iterations cannot be greatly reduced. With the new column generation system shown in table 8.5, the number of duty additions had decreased and it is now less likely that terminating at a sub-optimal solution will be worthwhile. A very simple heuristic has been tested which terminates the column generation process once a continuous solution has been found in which the integer part of the duty total is unlikely to decrease:

Define S_i to be the subset formed at the end of column generation iteration i
Define $T_i.f_i$ to be the duty total over S_i $(0 \leq f_i \leq 1)$

Then the column generation process should be terminated immediately after iteration i if the condition **(8.3)** holds :

$$\begin{aligned} &\text{IF} \quad T_i = T_{i-1} = T_{i-2} \\ &\text{AND} \quad f_i \geq 0.2 \end{aligned} \tag{8.5}$$

This heuristic will terminate the column generation process if the integer part of the duty total has not decreased in two iterations and its fractional part is sufficiently high to assume that it is unlikely to decrease by a full duty. Note that this heuristic will be of little use with a duty set for which a target number of duties has been specified because the associated side constraint is likely always to be binding.

Duty Set	Initial subset	Initial solution	Subset Additions	Final Subset	LP Solution	Time to LP	Total Time
AUC(S)	2067	115	4	3801	86.73	526.7	2472.5
CTJ(L)	1734	112	4	4086	86.43	440.2	1177.5

Table 8.6: Results of Terminating at a Sub Optimal Solution

Testing this simple heuristic on all problems which produced solutions has no effect on all but two duty sets, producing the identical solutions to those reported in Table 8.5. Table 8.6 reports the effect of an early termination on duty sets AUC(S) and CTJ(L).

The number of duty additions decreased by 3 on the AUC(S) data and by 2 on the CTJ(L) data, which reduced the execution time of the continuous solution. However, overall execution time increased with many more nodes explored when reoptimising with the added constraint which attempts to find a solution at the next highest integer. Given the reduction in number of column generation iterations provided by the alteration of parameters in section **8.3**, it is not beneficial at this stage to continue with further research on terminating at a sub-optimal solution.

8.6 Conclusion

The evidence suggests that where integer solution cannot be found, it is not caused by the column generation method. It is therefore preferable and often beneficial to append a constraint which increases the duty total by one, rather than altering the solution strategy where a column

generation approach may have provided a continuous solution which is too low for an integer solution to be found.

The alteration of some of the parameters governing the addition of duties to a subset has provided a further overall decrease in execution times of the continuous solution and although the total execution time has tended to increase, in most cases the solutions are still faster than with Sherali TRACS II.

Chapter 9

Summary and Conclusions

9.1 Introduction

This chapter summarises the implementation within a driver scheduling system of a column generation method which produces good driver schedules from a given set of potential duties. Suggested topics for possible further investigation are then introduced, and finally a suitable implementation of this system is reported.

9.1.1 A Column Generation Technique

The driver scheduling problem can be modelled as a set covering problem, where each piece of work must be covered by at least one driver. As there are potentially very many valid duties for any problem, most driver scheduling systems limit the number of duties which are generated and hence entered into the set covering model. Column generation models first optimise the relaxed model over an initial subset of valid duties, and then generate or select further duties to add to the subset before reoptimisation. Simplex multipliers formed by the simplex method are used to calculate the reduced costs of new duties, and any number of duties with negative reduced costs can be added to the subset each time. When no more duties can be added to the subset the continuous solution is optimal and a method will be adopted to find an integer schedule.

It was decided that the specialised technique to be implemented would first generate a larger set of potential duties so that the column generation method then selects duties from this superset. The current branch and bound process will be used to form an integer schedule.

9.1.2 An Algorithm for the Column Generation Procedure

- **Step 1** Read in the ZIP parameter file which includes information as to how the process should be run and any user-defined side constraints.

- **Step 2** Read in and store the bus information, which includes details of the number of generated duties which start or end at each relief opportunity. In this way redundant constraints can be eliminated. These correspond to pieces of work which are never used to change drivers.

- **Step 3** Read in and store the duties, and for each piece of work form lists of those duties which cover it. As the duties are being read in the equality constraints are determined by looking for buses whose first pieces are only covered by duties of which they are the first piece, and buses whose last pieces are only covered by duties of which they are the last piece.

- **Step 4** Create an initial solution by looking through the pieces of work in increasing order of the number of duties covering them. For a piece of work that is not yet covered, all duties which do cover it are considered. A notional cost is calculated using the function:

$$\text{Min} \quad \frac{D_j}{NU_j}$$

where:

$$\begin{aligned} D_j &= \text{Sherali cost of duty } x_j \\ NU_j &= \text{number of currently uncovered workpieces covered by } x_j. \end{aligned}$$

The least cost duty which does not violate a constraint is chosen to be included in the initial solution. Uncovered work is allowed in the initial solution to ensure that it is feasible, and then banned before optimisation. An initial duty set is chosen as the initial solution is being formed, by adding the duty being considered to the subset if it covers a piece of work not already covered by at least a specified number of duties within the duty subset. The suggested lower limit is 10.

- **Step 5** Initialise x to be the number of duties in the initial solution. For every piece of work which remains uncovered in the initial solution, increase the value of x by one. Calculate the Sherali weight by summing the costs of the x most expensive duties, and

add this weight to the cost of all duties.

- **Step 6** Solve the LP relaxation over the current subset of duties.
- **Step 7** Consider the values of the simplex multipliers in descending order. Duties which cover the corresponding piece of work and have a negative reduced cost are added to the subset up to a limit defined by I.

$$I = \text{MAX } \{5, 50\% \text{ (total duties remaining/No. pieces of work) } \}$$

If the piece of work has been covered by duties added earlier in the same iteration then this is taken into account and the limit is lowered accordingly.

The addition of duties to the subset in any iteration is bounded by M or P, which respectively limit the total number of duties which can be added and the number of simplex multipliers which are used to select further duties.

$$M = \text{MAX } \{50, 10\% \text{ (total duties remaining)}\}$$
$$P = 20\%\text{(total number of pieces of work where additions are made)}$$

If no duties are added the solution is optimal, otherwise **go to Step 6**.

- **Step 8** If the LP solution is integral then the process stops, otherwise reduce the duty set using the REDUCE process which is described in section **5.4.3**.
- **Step 9** If the optimal duty total is currently fractional, add a side constraint which rounds up the duty total to provide a target number of duties in the integer solution.
- **Step 10** Find an integer solution using the branch and bound method described in section **5.4.3**.

9.2 Conclusions

The column generation procedure described has been implemented and tested on seven problem instances, each of which has two generated duty set sizes. The smaller duty set is a subset of

the larger set and has been heuristically reduced to be run through TRACS II [15, 16, 17, 18] which is a set covering system developed at the University of Leeds. Although the small duty sets in a column generation method would be superseded by the larger duty sets, their size may be representative of a different problem and so the results are still significant. Results have been compared with those produced by a specialised version of TRACS II, called the Sherali TRACS II system. This system incorporates a Sherali weighted objective function and other features which make it more comparable to the implemented column generation system.

The purpose of the research was to investigate whether better solutions could be found by allowing more potential duties to be generated. This would justify the use of a column generation system which considers many more duties but works with a much smaller subset. Also, the column generation system should produce results in no more time than Sherali TRACS II on the same duty set, making it feasible for a scheduler to use such a system.

For three problems the continuous solution produced by the larger data set has proved to be too low to produce an integer schedule. However, the addition of a constraint which specifies a duty total at the next highest integer has enabled a schedule to be found using column generation in all cases, and these schedules are better than those produced from the smaller duty set. For the Sherali TRACS II system with such a constraint only two problems succeed in finding an integer schedule and their execution times are much less acceptable than those found using column generation.

Certainly, results obtained from the larger duty sets either produce schedules with fewer duties or a schedule with the same number of duties but with a lower overall cost. Results also confirm that the column generation system produces an overall decrease in execution times of 41% over the Sherali TRACS II system on problems which do not require the addition of a target number of duties. This concludes that a column generation solution procedure can produce better solutions more quickly than a successful and widely implemented driver scheduling system.

9.3 Further Work

The following sections discuss some of the issues which could be investigated with regard to possible improvements in the column generation system.

9.3.1 General Improvements

More work could be done on improving parameters I,M and P which limit the number of duties added to the subset per simplex multiplier and overall. More testing is also needed to test the interaction between these parameters to provide a varied subset of duties which is both large enough to provide a reasonable set at the branch and bound phase but does not contain too many unproductive duties.

Willers [14] suggested various improvements in the execution of the Sherali TRACS II system which may be usefully implemented in a column generation system. Willers tested several initial solution strategies and concluded that a maximum duration strategy provides the highest number of faster solutions. This method uses a nominal cost function which selects duties to be included in the initial solution based upon maximising the duration of uncovered work rather than the number of uncovered pieces. An improvement in the initial solution may reduce the number of column generation iterations required to reach the optimal continuous solution. Willers also suggested that the branching strategy could be altered, in particular terminating the search as soon as an integer solution has been found. This is based upon the observation that in most cases the branch and bound process produces only one integer solution and yet searches for better ones within a specified tolerance. This is a potential improvement to any version of the TRACS II system, irrespective of whether it incorporates column generation techniques.

It may also be beneficial to update the optimisation methods. Section **7.3.1** describes how methods developed more recently are more efficient and may be more suited to a column generation approach which optimises a model on more than one occasion.

9.3.2 Alternative Pricing Strategies

Willers [14] reported that the implemented steepest-edge pricing strategy may no longer be beneficial as much larger models are now solved compared to those with up to 5000 duties reported in [15]. Although the column generation system is intended for use on much larger data sets, the working subset of duties is relatively small and so the steepest-edge pricing

strategy may remain the most appropriate. A set of new columns to enter the subset is selected based upon negative pure reduced costs and apart from the first entrant variable which is chosen to be the duty with the most negative pure reduced cost, the duties are then priced according to the steepest-edge strategy. Indeed Bixby et al. [86] observed that steepest-edge pricing produced much better results than using pure reduced costs, but also proposed an alternative pricing strategy *lambda price* which scales the reduced costs by dividing the duty cost by the sum of the simplex multipliers. Bixby orders the duties with negative reduced costs by this price in order to select only a subset of them. This is not applicable in the column generation application described, which limits the selection by parameters rather than duty quality, but it may be beneficial in the optimisation strategy. Partial pricing also may prove useful as it reduces the calculations necessary for each iteration and so may be powerful in column generation systems where the model is optimised on more than one occasion.

9.3.3 Branch and Bound

No work has been carried out in order to customise the branch and bound strategy to better apply to a column generation method. The most obvious investigation would be to introduce a column generation technique within the search tree so that it no longer relies on the available duties to find an integer solution. A possible method would be to introduce more duties at an infeasible node and to alter the branching strategy to bias the search to those nodes which consider smaller duty sets. In this way the solution method does not have to consider the larger problem if a solution can be found from the original duty set.

9.3.4 Column Removal

It may be beneficial to introduce a mechanism whereby duties can be removed from the subset if they are unlikely or certain not to be used in improving the objective cost. This would have the benefit of keeping the subset size lower, especially where more column generation iterations were needed and added many duties, or if column generation were to be used in the branch and bound strategy which will introduce further duties later. The danger of removing duties before the continuous solution is that there may not be sufficient duties remaining from which to find an integer solution. If a column generation technique were to be introduced into the branch and bound phase, however, the strategy proposed by Willers [14] of removing duties at some nodes in the tree may be useful in ensuring that the duty size at each node remained more constant.

9.4 Possible Application of System

The bus duty generation procedure used within TRACS II only considers duties containing up to three spells of work. Where manual schedules have been formed which include duties with more spells it is often due to difficulties in completing the systematic scheduling approach over a large bus schedule, and computerised systems often produce better schedules without the need for duties with more than three parts. There is a version of TRACS II which has been developed to schedule train driver duties [48] in which the duty generation procedure has been altered to generate four-part duties. Currently it is not necessary to generate duties with five or more parts but manual train driver schedules may have such duties, particularly overnight, or to allocate other tasks related, e.g. vehicle preparation. Allowing four-part duties creates a much larger potential set of valid duties which have to be reduced using heuristics in order to use TRACS II. Train driver scheduling therefore may particularly benefit from a column generation approach in which more duties can be considered when forming a final schedule.

9.5 Summary of Achievements

For the problem of chaining mealbreaks to produce more efficient driver schedules, a mathematical model has been proposed which optimises such chains over a specified time period. This formulation produced an optimal solution for a simple example, although it is noted that the complexity of the constraints is dependent upon the labour agreement rules.

A guaranteed optimal schedule for a given set of generated valid duties can be found by exhaustively searching through all duty combinations which cover the bus work. A search tree method has been tested which includes heuristics to limit searching on branches where it is known that the optimal solution cannot be found. On a small example this method produces the optimal solution quickly but it is inadequate for solving real problems. The method does, however, incorporate a technique for storing duty information differently, which is usefully implemented within the column generation system described.

The proposed column generation procedure has led to significantly less (up to 99% less) computer time in 10 out of 12 cases. For problems where TRACS II is used on a smaller duty set and the proposed column generation procedure is used on a larger duty set, in 1 case it has produced a solution where none were previously obtained and in 4 cases it has given solutions requiring fewer duties than previously.

In addition, it has been shown that column generation allows much larger problems to be solved in a reasonable amount of computer time. This implies that better solutions can be obtained in some cases by considering significantly larger sets of potential duties. Larger problems can be tackled through column generation which previously had to be subdivided.

Bibliography

[1] M. O. Ball, L. D. Bodin, and R. Dial. Experimentation With a Computerized System for Scheduling Mass Transit Vehicles and Crews. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 313–334. North-Holland, 1981.

[2] M. O. Ball and L. D. Bodin. A Matching Based Heuristic for Scheduling Mass Transit Vehicles and Crews. *Transportation Science*, 17:4–31, 1983.

[3] T. Hartley. A Glossary of Terms in Bus and Crew Scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 353–359. North-Holland, 1981.

[4] J. F. Turner. *Timetable and Duty Schedule Compilation (Road Passenger Transport).* Sir Isaac Pitman and Sons Ltd., 1946.

[5] *Preprints of the Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services*, Chicago, 1975.

[6] A. Wren, editor. *Computer Scheduling of Public Transport.* Proceedings of the Second International Workshop on Computer-Aided Scheduling of Public Transport. North-Holland, 1981.

[7] J.-M. Rousseau, editor. *Computer Scheduling of Public Transport 2.* Proceedings of the Third International Workshop on Computer-Aided Scheduling of Public Transport. North-Holland, 1985.

[8] J. R. Daduna and A. Wren, editors. *Computer-Aided Transit Scheduling.* Proceedings of the Fourth International Workshop on Computer-Aided Scheduling of Public Transport. Springer-Verlag, 1988.

[9] M. Desrochers and J.-M. Rousseau, editors. *Computer-Aided Transit Scheduling.* Proceedings of the Fifth International Workshop on Computer-Aided Scheduling of Public Transport. Springer-Verlag, 1992.

[10] J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors. *Computer-Aided Transit Scheduling.* Proceedings of the Sixth International Workshop on Computer-Aided Scheduling of Public Transport. Springer-Verlag, 1995.

[11] A. Wren and J.-M. Rousseau. Bus Driver Scheduling - An Overview. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 173–187. Springer-Verlag, 1995.

[12] S. E. G. Elias. The Use of Digital Computers in the Economic Scheduling for Both Man and Machine in Public Transport. Technical Report 49, Kansas State University Bulletin, 1964.

[13] S. E. G. Elias. A Mathematical Model for Optimizing the Assignment of Man and Machine in Public Transit "Run-Cutting". Technical Report 81, West Virginia University Engineering Experiment Station, 1966.

[14] W. P. Willers. *Improved Algorithms for Bus Crew Scheduling.* PhD thesis, University of Leeds, 1995.

[15] B. M. Smith. *Bus Crew Scheduling using Mathematical Programming.* PhD thesis, University of Leeds, 1986.

[16] B. M. Smith and A. Wren. A Bus Crew Scheduling System Using a Set Covering Formulation. *Transportation Research*, 22A:97 – 108, 1988.

[17] A. Wren and B. M. Smith. Experiences with a Crew Scheduling System Based on Set Covering. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 104–118. Springer-Verlag, 1988.

[18] B. M. Smith and M. E. Parker. *University of Leeds IMPACS Crew-Scheduling System User Manual (Amdahl Version).* University of Leeds, October 1993.

[19] R. Lessard, J.-M. Rousseau, and D. Dupuis. HASTUS I: A Mathematical Programming Approach to the Bus Driver Scheduling Problem. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 255–267. North-Holland, 1981.

[20] J.-M. Rousseau and J.-Y. Blais. HASTUS: An Interactive System for Buses and Crew Scheduling. In J.-M. Rousseau, editor, *Computer Scheduling of Public Transport 2*, pages 45–60. North-Holland, 1985.

[21] J.-M. Rousseau, R. Lessard, and J.-Y. Blais. Enhancements to the HASTUS Crew Scheduling Algorithm. In J.-M. Rousseau, editor, *Computer Scheduling of Public Transport 2*, pages 295–310. North-Holland, 1985.

[22] J.-Y. Blais and J.-M. Rousseau. Overview of HASTUS Current and Future Versions. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 175–187. Springer-Verlag, 1988.

[23] J.-Y. Blais, J. Lamont, and J.-M. Rousseau. The HASTUS Vehicle and Manpower Scheduling System at the Societé dé de Transport de la Communauté Urbaine de Montréal. *Interfaces*, 20:26–42, 1990.

[24] N. Hamer and L. Séguin. The HASTUS System: New Algorithms and Modules for the 90s. In M. Desrochers and J.-M. Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 17–29. Springer-Verlag, 1992.

[25] J. Hoffstadt. Computerized Vehicle and Driver Scheduling for the Hamburger Hochbahn Aktiengesellschaft. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 35–52. North-Holland, 1981.

[26] J. R. Daduna and M. Mojsilovic. Computer-Aided Vehicle and Duty Scheduling Using the HOT Programme System. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 133–146. Springer-Verlag, 1988.

[27] M. Völker and P. Schütze. Recent Developments of the HOT System. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 334–348. Springer-Verlag, 1995.

[28] J. Ross. BERTA - Computer-Aided System for the Scheduling and Evaluation of Public Transport Operations on its way to Realization. In *Sixth International Workshop on Computer-Aided Scheduling of Public Transport*, Lisbon, August 1993.

[29] I. Patrikalakis and D. Xerocostas. A New Decomposition Scheme of the Urban Public Transport Scheduling Problem. In M. Desrochers and J.-M. Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 407–425. Springer-Verlag, 1992.

[30] P. Carraresi, G. Gallo, N. Ciaramella, and L. Lucchesi. BDS: A System for the Bus Drivers' Scheduling Problem Integrating Combinatorial Optimization and Logic Programming. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 68–82. Springer-Verlag, 1988.

[31] E. Tosini and C. Vercellis. An Interactive System for Extra-Urban Vehicle and Crew Scheduling Problems. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 41–53. Springer-Verlag, 1988.

[32] A. Wren and M. Chamberlain. The Development of Micro-BUSMAN: Scheduling on Micro-Computers. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 160–174. Springer-Verlag, 1988.

[33] J. C. Falkner and D. M. Ryan. EXPRESS: Set Partitioning for Bus Crew Scheduling in Christchurch. In M. Desrochers and J.-M. Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 359–378. Springer-Verlag, 1992.

[34] D. M. Ryan. ZIP - a Zero One Integer Programming Package for Scheduling. Technical Report CSS-85, AERE, Computer Science and Systems Division, Harwell, Oxfordshire, 1980.

[35] P Mott and H Frische. INTERPLAN - An Interactive Program System for Crew Scheduling and Rostering of Public Transport. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 200–211. Springer-Verlag, 1988.

[36] L. K. Luedtke. RUCUS II: A Review of System Capabilities. In J.-M. Rousseau, editor, *Computer Scheduling of Public Transport 2*, pages 61–116. North-Holland, 1985.

[37] M. O. Ball, L. D. Bodin, and J. Greenberg. Enhancements to the RUCUS II Crew Scheduling System. In J.-M. Rousseau, editor, *Computer Aided Scheduling of Public Transport 2*, pages 279–294. North-Holland, 1985.

[38] G. Bennington and K. Rebibio. Overview of RUCUS Vehicle Scheduling Program(BLOCKS). In D. Bergmann and L. Bodin, editors, *Preprints: Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services*, 1975.

[39] R. S. K. Kwan, A. Wren, and B. M. Smith. An Expert System for Bus Crew Scheduling. Technical Report 88.14, School of Computer Studies, University of Leeds, July 1988.

[40] R. Kwan, A. Wren, and L. Zhao. Driver Scheduling Using Intelligent Estimation Techniques with Heuristic Searches. In M. Desrochers and J.-M. Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 379–394. Springer-Verlag, 1992.

[41] L. Zhao, A. Wren, and R. S. K. Kwan. Enriching Rules in a Driver Duty Estimator. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 236–247. Springer-Verlag, 1995.

[42] L. Zhao. *A Knowledge Based Driver Duty Estimator*. PhD thesis, University of Leeds, 1993.

[43] A. Wren and D. O. Wren. A Genetic Algorithm for Public Transport Driver Scheduling. *Computers and Operations Research*, pages 101–110, 1994.

[44] R. Clement and A. Wren. Greedy Genetic Algorithms, Optimizing Mutations, and Bus Driver Scheduling. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 213–235. Springer-Verlag, 1995.

[45] R. S. K. Kwan and A. Wren. Hybrid Genetic Algorithms for Bus Driver Scheduling. In *Advanced Methods in Transportation Analysis*. Springer-Verlag, 1996. to appear.

[46] A. Wren, R. S. K. Kwan, and M. E. Parker. Scheduling of Rail Driver Duties. In *Computers in Railways IV*, volume 2: Railway Operations, pages 81–89. Computational Mechanics Publications, 1994.

[47] M. E. Parker, A. Wren, and R. S. K. Kwan. Modelling the Scheduling of Train Drivers. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 359–370. Springer-Verlag, 1995.

[48] A. S. K. Kwan, R. S. K. Kwan, M. E. Parker, and A. Wren. Producing Train Driver Shifts by Computer. To be Presented at the 5th International Conference on Computer-Aided Design, Manufacture and Operations in the Railway and Other Mass Transit Systems (Berlin), 1996.

[49] R. E. Marsten and F. Shepardson. Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model. *Networks*, 11:165–177, 1981.

[50] G. W. Graves, R. D. McBride, I. Gershkoff, D. Anderson, and D. Mahidhara. Flight Crew Scheduling. *Management Science*, 39:736–745, 1993.

[51] K. L. Hoffman and M. Padberg. Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science*, 39:657–682, 1993.

[52] L. R. Ford Jr. and D. R. Fulkerson. A Suggested Computation for Maximal Multi-Commodity Network Flows. *Management Science*, 5:97–101, 1958.

[53] G. B. Dantzig and P. Wolfe. Decomposition Principle for Linear Programs. *Operations Research*, 8:101–111, 1960.

[54] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, 9:849–859, 1961.

[55] P. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Solving Binary Cutting Stock Problems by Column Generation and Branch-and-Bound. *Computational Optimization and Applications*, 3:111–130, 1994.

[56] A. Mehrotra and M. A. Trick. A column generation approach for graph coloring. WWW - http://mat.gsia.cmu.edu/color.ps, April 1995.

[57] C. C. Riberio, M. Minoux, and M. C. Penna. An Optimal Column-Generation-With-Ranking Algorithm for Very Large Scale Set Partitioning Problems in Traffic Assignment. *European Journal of Operations Research*, 41:232–239, 1989.

[58] P. Carraresi, M. Nonato, and L. Girardi. Network Models, Lagrangean Relaxation and Subgradients Bundle Approach in Crew Scheduling Problems. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 188–212. Springer-Verlag, 1995.

[59] M. Desrochers and F. Soumis. CREW-OPT: Crew Scheduling by Column Generation. In J. R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, pages 83–90. Springer-Verlag, 1988.

[60] M. Desrochers, J. Gilbert, M. Sauvé, and F. Soumis. CREW-OPT: Subproblem Modelling in a Column Generation Approach to Urban Crew Scheduling. In M. Desrochers and J.-M. Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 395–406. Springer-Verlag, 1992.

[61] J. M. Rousseau. Results Obtained with Crew-Opt, a Column Generation Method for Transit Crew Scheduling. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 349–358. Springer-Verlag, 1995.

[62] M. Minoux. Column Generation Techniques in Combinatorial Optimization: A new Application to Crew Pairing Problems. In *Proceedings of the 24th AGIFORS Annual Symposium*, pages 18–29, September 1984.

[63] S. Lavoie, M. Minoux, and E. Odier. A New Approach for Crew Pairing Problems by Column Generation with an Application to Air Transportation. *European Journal of Operations Research*, 35:45–58, 1988.

[64] P. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Airline Crew Scheduling: A New Formulation and Decomposition Algorithm. *To appear in Operations Research*, 1994.

[65] C. Barnhart, L. Hatay, and E. L. Johnson. Deadhead Selection for the Long-Haul Crew Pairing Problem. *Operations Research*, 43:491–499, 1995.

[66] M. J. Brusco, L. W. Jacobs, R. J. Bongiorno, D. V. Lyons, and B. Tang. Improving Personnel Scheduling at Airline Stations. *Operations Research*, 43:741–751, 1995.

[67] C. C. Riberio and F. Soumis. A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem. Technical Report G-91-32, GERAD, Montréal, January 1993.

[68] L. Bianco, A. Mingozzi, and S. Riciardelli. A Set Partitioning Approach to the Multiple Depot Vehicle Scheduling Problem. Technical Report R. 350, IASI-CNR, Rome, December 1992.

[69] L. Bianco, A. Mingozzi, M. Navoni, and S. Riciardelli. An Exact Algorithm for Combining Vehicle Trips. In J. R. Daduna, I. Branco, and J. M. P. Paixáo, editors, *Computer-Aided Transit Scheduling*, pages 145–172. Springer-Verlag, 1995.

[70] K. Reddington. *Scheduling of Multiple Vehicle Types: The Allocation of Locomotives to Trains.* PhD thesis, London School of Economics, 1992.

[71] J. L. Kennington and R. V. Helgason. *Algorithms for Network Programming.* John Wiley and Sons, 1980.

[72] Dash. *XPRESS-MP Reference Manual.* Dash Associates, 1991.

[73] M. Chamberlain and A. Wren. Developments and Recent Experienxce with the BUSMAN and BUSMAN II Systems. In M. Desrochers and J.-M. Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 1–15. Springer-Verlag, 1990.

[74] G. Mitra and K. Darby-Dowman. A Computer Based Bus Crew Scheduling System Using Integer Programming. In J.-M. Rousseau, editor, *Computer Scheduling of Public Transport 2*, pages 223–232. North-Holland, 1985.

[75] P. D. Manington. *Mathematical and Heuristic Approaches to Road Transport Scheduling.* PhD thesis, University of Leeds, 1977.

[76] H. D. Sherali. Equivalent Weights for Lexicographic Multi-Objective Programs: Characterizations and Computations. *European Journal of Operations Research*, 18:57–61, 1982.

[77] W. P. Willers, L. G. Proll, and A. Wren. A Dual Strategy for Solving the Linear Programming Relaxation of a Driver Scheduling System. *Annals of Operations Research*, 58:519–531, 1995.

[78] J. K. Reid. Fortran Subroutines for Handling Sparse Linear Programming Bases. Technical Report R-8269, AERE, Harwell, Oxfordshire, 1976.

[79] P. M. J. Harris. Pivot Selection Methods of the Devex LP Code. *Mathematical Programming Study*, 4:30–57, 1975.

[80] D. Goldfarb and J. K. Reid. A Practicable Steepest -Edge Simplex Algorithm. *Mathematical Programming*, 12:361–371, 1977.

[81] D. M. Ryan and B. A. Foster. An Integer Programming Approach to Scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 269–280. North-Holland, 1981.

[82] L. G. Proll. R36: Mathematical Programming Notes. University of Leeds, 1991.

[83] M. Desrochers. *La Fabrication D'Horaires de Travail pour les Conducteurs D'Autobus par une Méthode de Génération de Colonnes.* PhD thesis, Centre de Recherche Sur Les Transports, Université de Montréal, 1986.

[84] M. Desrochers and F. Soumis. A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science*, 23:1 – 13, 1989.

[85] G. L. Nemhauser. The Age of Optimisation: Solving Larger-Scale Real-World Problems. *Operations Research*, 42:5–13, 1994.

[86] R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno. Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods. *Operations Research*, 40:885–897, 1992.